13.
Distribution Logistics – Vehicle Routing

What is the Vehicle Routing Problem?

Given a set of customers,

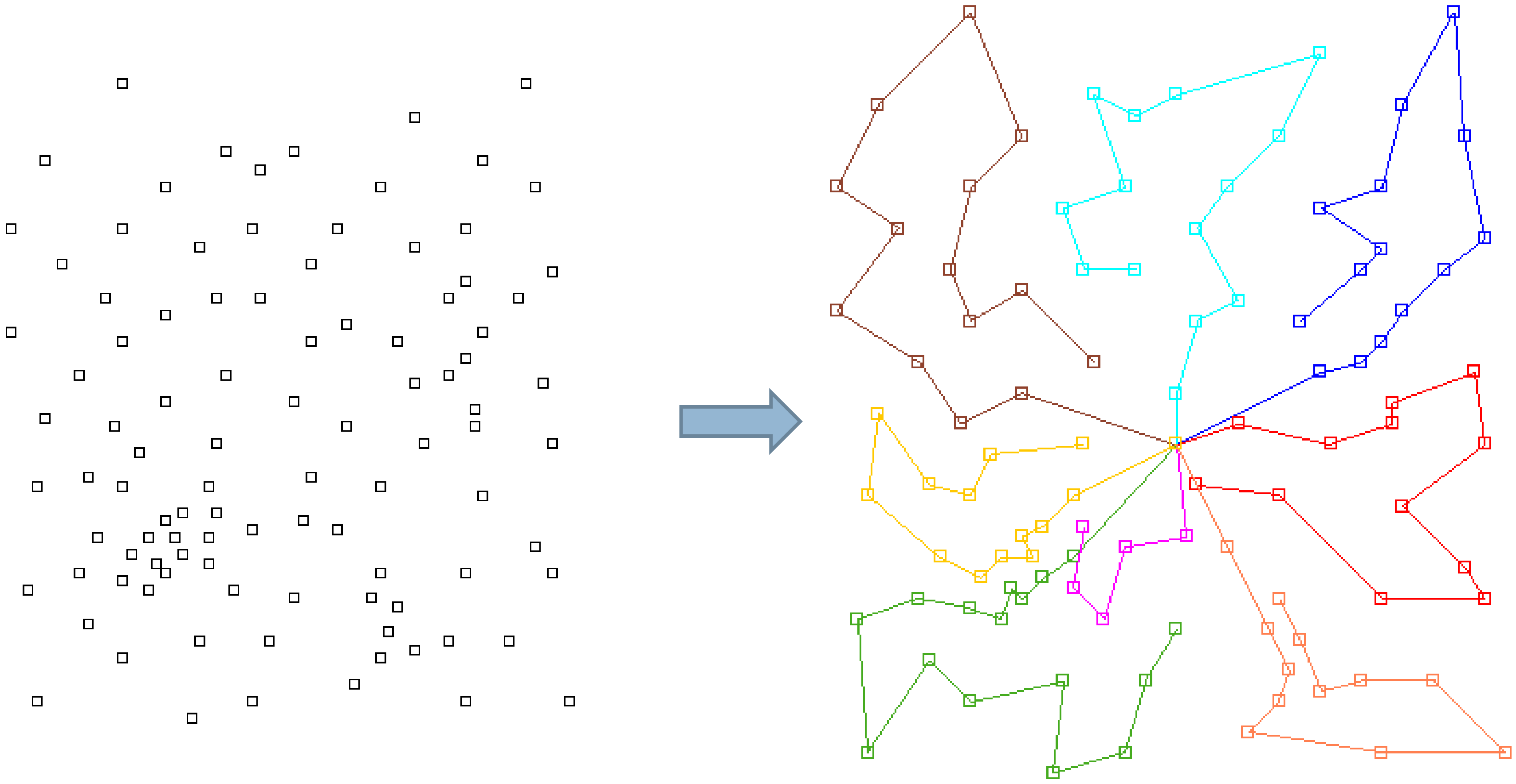
and

a fleet of vehicles to make deliveries,

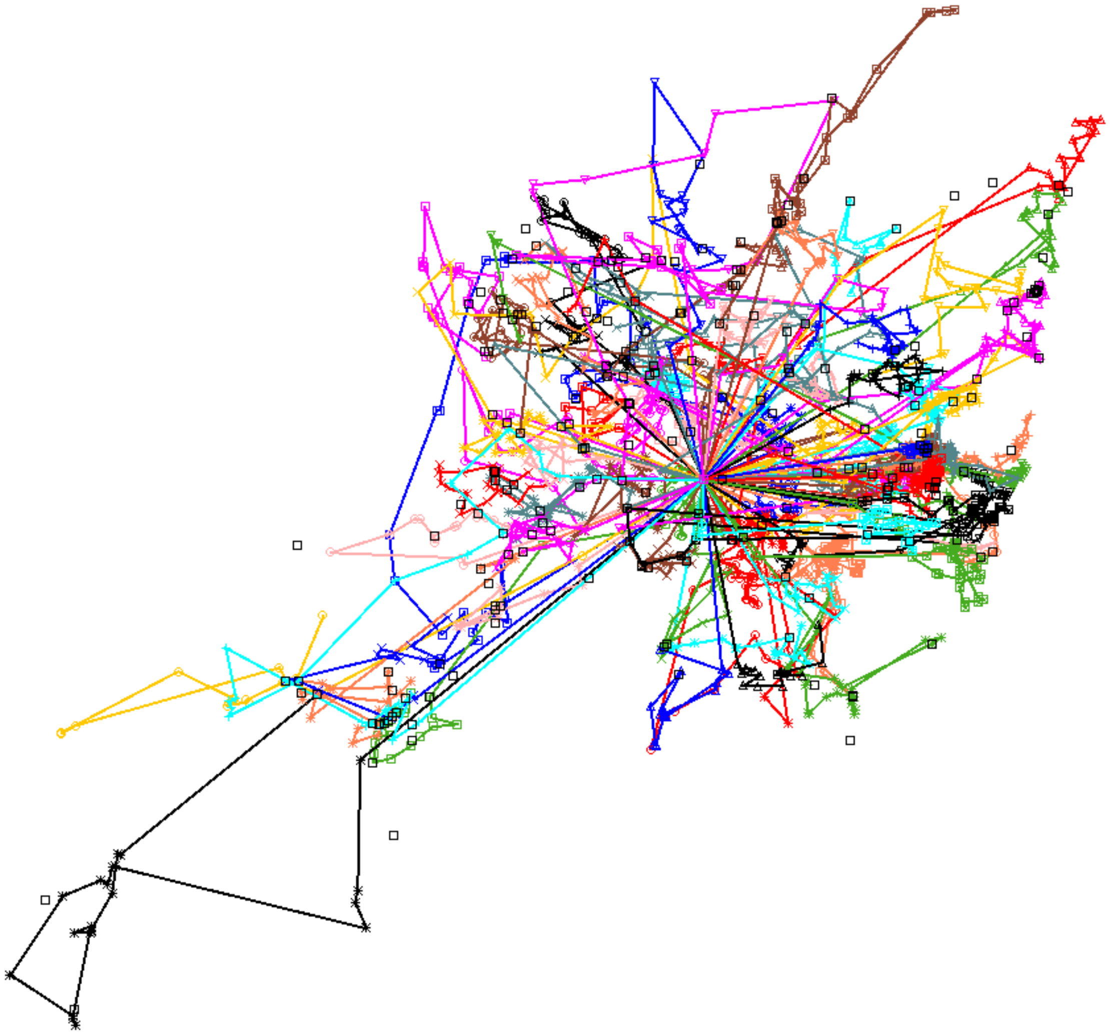
find

a set of routes that services all customers at minimum cost

What is the Vehicle Routing Problem?



What is the Vehicle Routing Problem?

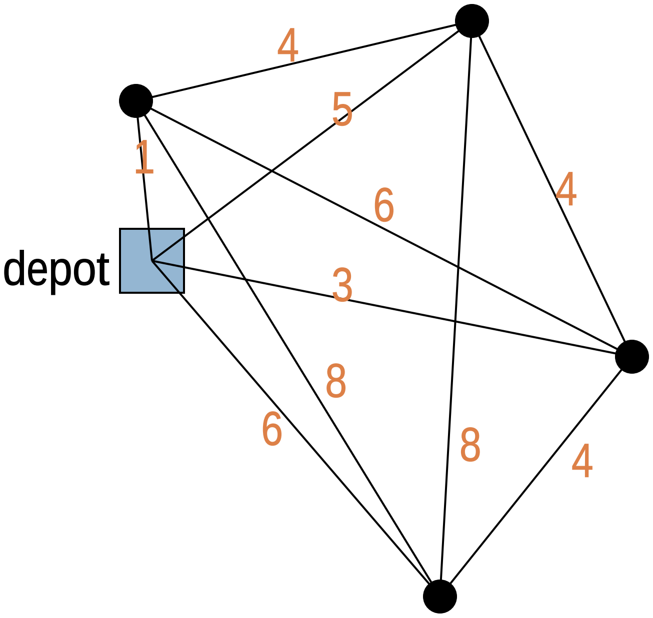


Basic VRP structure

- Find the best vehicle route(s) to serve a set of geographically scattered orders from customers.
- Best route may be
  - minimum cost,
  - minimum distance, or
  - minimum travel time.
- Orders may be
  - Delivery from depot to customer
  - Pickup at customer and return to depot
  - Pickup at one place and deliver to another place

Basic VRP structure

- Nodes: physical locations
  - Depot.
  - Customers
- Arcs or Links
  - Transportation links
- Number on each arc represents cost, distance, or travel time.



Basic VRP structure

- For each customer, we know
  - Quantity required
  - The cost to travel to every other customer
- For the vehicle fleet, we know
  - The number of vehicles
  - The capacity (weight and/or volume)
- We must determine which customers each vehicle serves, and in what order, to minimise cost

Basic VRP structure

Objective function

- In academic studies, usually a combination:
  - First, minimise number of routes
  - Then minimise total distance or total time

- In real world
  - A combination of time and distance
  - Must include vehicle- and staff-dependent costs
  - Usually vehicle numbers are fixed

MIP formulation

Data:

c_{ij}: Cost of travel from i to j

q_i: Demand at i

Decision variables:

x_{ijk}: Travel direct from i to j on vehicle k

$$\text{minimise}: \sum_{i,j} \mathrm{c}_{\mathrm{ij}} \sum_k x_{ijk}$$

subject to

$$\begin{aligned}
\sum_{\mathrm{i}} \sum_{\mathrm{k}} \mathrm{x}_{\mathrm{ijk}} &= 1 \quad \forall j \\
\sum_{\mathrm{j}} \sum_{\mathrm{k}} \mathrm{x}_{\mathrm{ijk}} &= 1 \quad \forall i \\
\sum_{\mathrm{j}} \sum_{\mathrm{k}} \mathrm{x}_{\mathrm{ihk}} - \sum_{\mathrm{j}} \sum_{\mathrm{k}} \mathrm{x}_{\mathrm{hjk}} &= 0 \quad \forall k, h \\
\sum_{i} \mathrm{q}_{\mathrm{i}} \sum_{\mathrm{j}} \mathrm{x}_{\mathrm{ijk}} &\leq Q_k \quad \forall k \\
\{\mathrm{x}_{\mathrm{ijk}}\} &\subseteq S \\
x_{ijk} &\in \{0,1\}
\end{aligned}$$

Travelling Salesman Problem

History: Travelling Salesman Problem - TSP

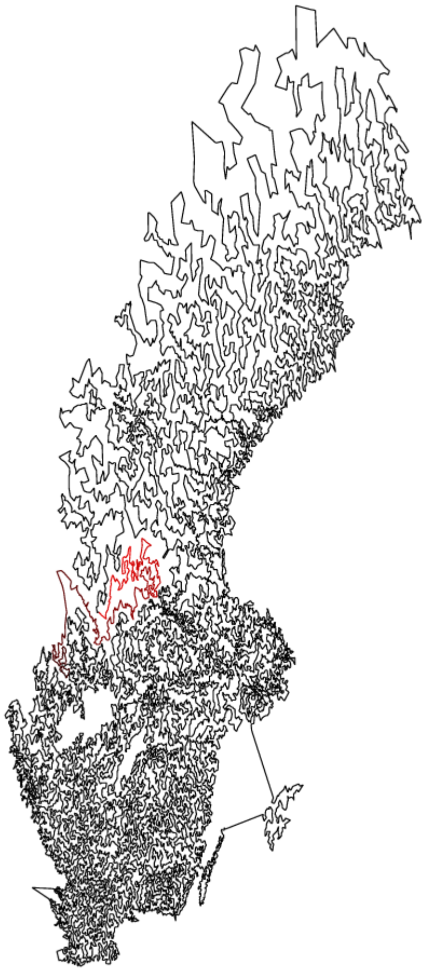
- A travelling salesman has to visit a number of cities. He knows the cost of travel between each pair. *What order does he visit the cities to minimise cost?*

  - A sub-problem in many others
  - Used in chip fabrication and many other real-world problems
  - TSP = VRP with 1 vehicle of infinite capacity
  - In vehicle routing - having decided which vehicle will visit which customers, each vehicle route is a travelling salesman problem

Travelling Salesman Problem

- Exact solution are found regularly for problems with 200-300 cities, and occasionally for problems with 1000 nodes.
- Some larger problems solved

  (24,798 cities, towns and villages in Sweden)
- But – no constraints on the solution
- Even one constraint, and the whole method is unusable



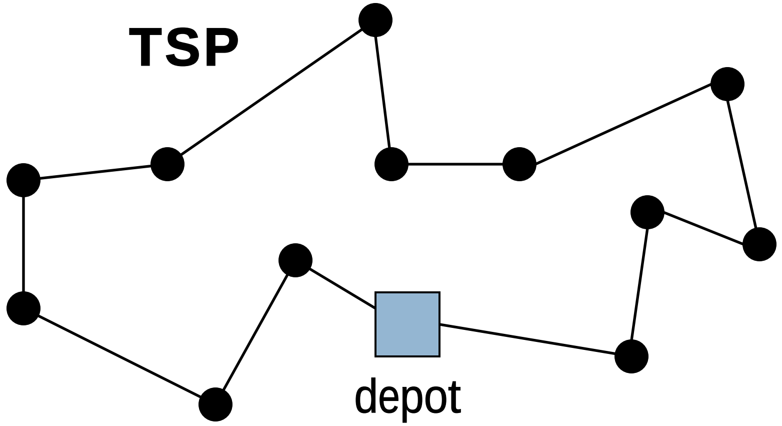
TSP Solutions

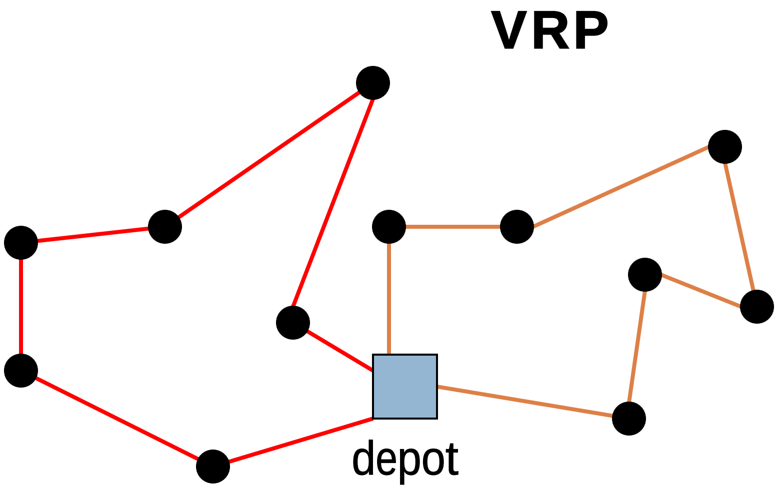
- Heuristics
  - **Construction**: build a feasible route.
  - **Improvement**: improve a feasible route.
    - Not necessarily optimal, but fast.
    - Performance depends on problem.
    - Worst case performance may be very poor.

- Exact algorithms
  - Integer programming.
  - Branch and bound.
    - Optimal, but usually slow.
    - Difficult to include complications

TSP & VRP

- **TSP: Travelling Salesman Problem**
  - One route can serve all orders.
- **VRP: Vehicle Routing Problem**
  - More than one route is required to serve all orders.





Simplest Model: TSP

- Given a depot and a set of n customers, find a route (or “tour”) starting and ending at the depot, that visits each customer once and is of minimum length.
  - One vehicle.
  - No capacities.
  - Minimize distance.
  - No time windows.
  - No compatibility constraints.
  - No DOT rules.

Symmetric and Asymmetric

Let c_{ij} be the cost (distance or time) to travel from i to j.

If $c_{ij} = c_{ji}$ for all customers, then the problem is *symmetric*.

- Direction does not affect cost.

If $c_{ij} \neq c_{ji}$ for some pair of customers, then the problem is *asymmetric*.

- Direction does affect cost.

TSP Construction Heuristics

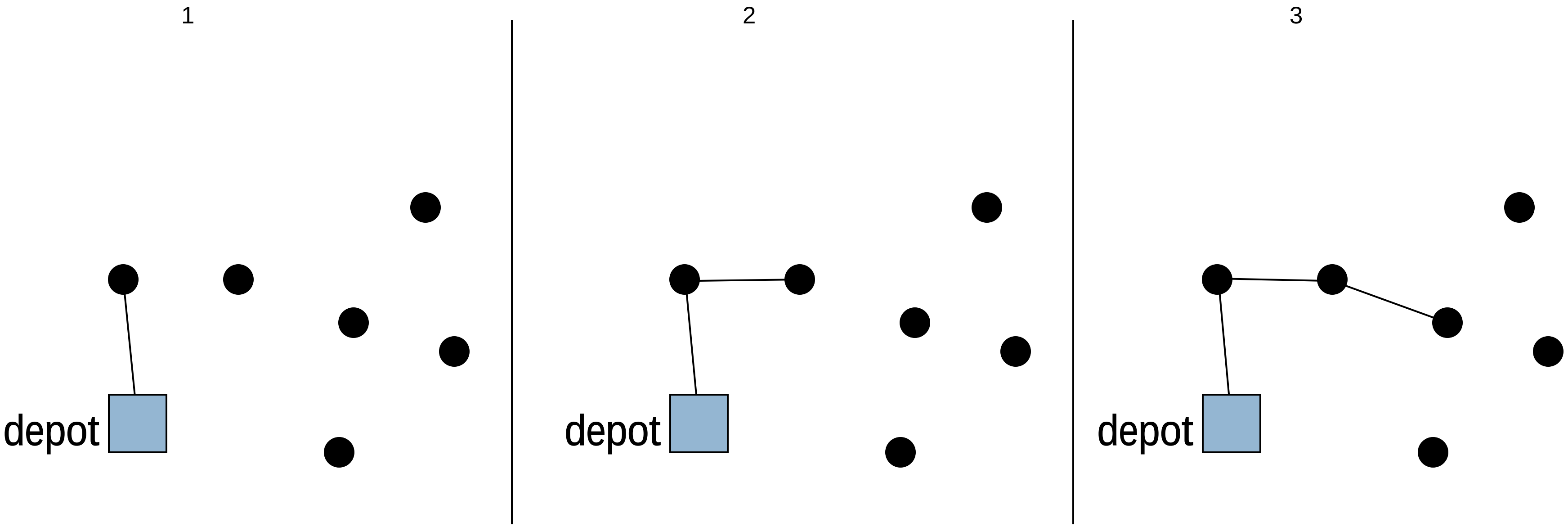
- Nearest neighbor.
  - Add nearest customer to end of the route.

- Nearest insertion.
  - Go to nearest customer and return.
  - Insert customer closest to the route in the best sequence.

- Savings method.
  - Add customer that saves the most to the route

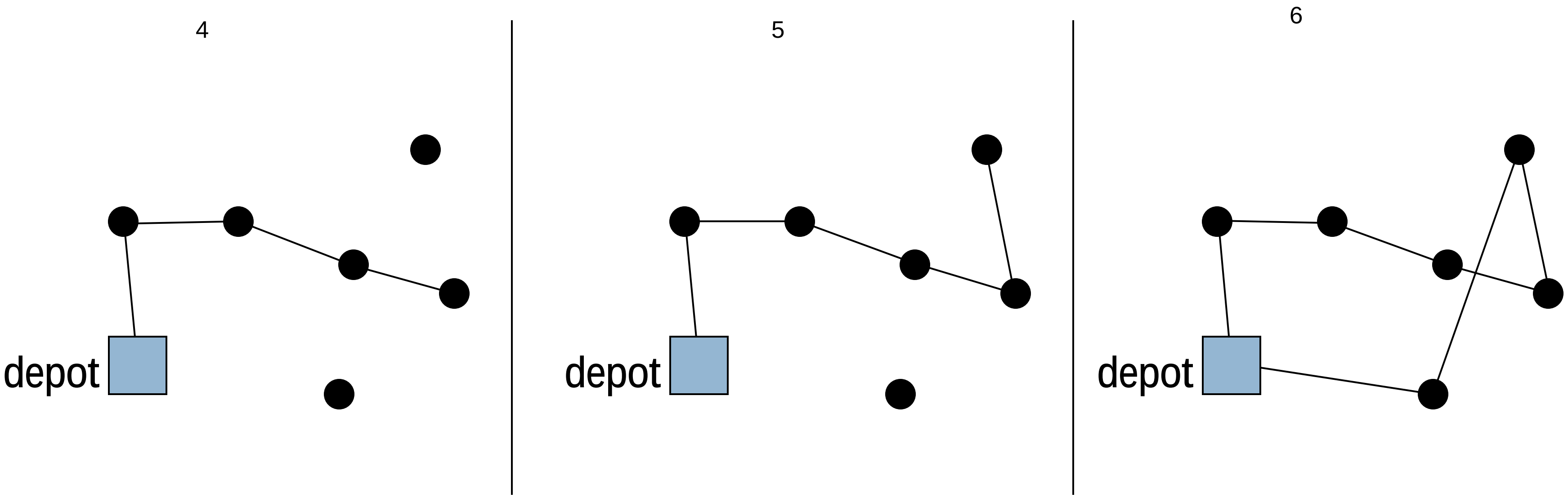
Nearest Neighbor

- Add nearest customer to end of the route.



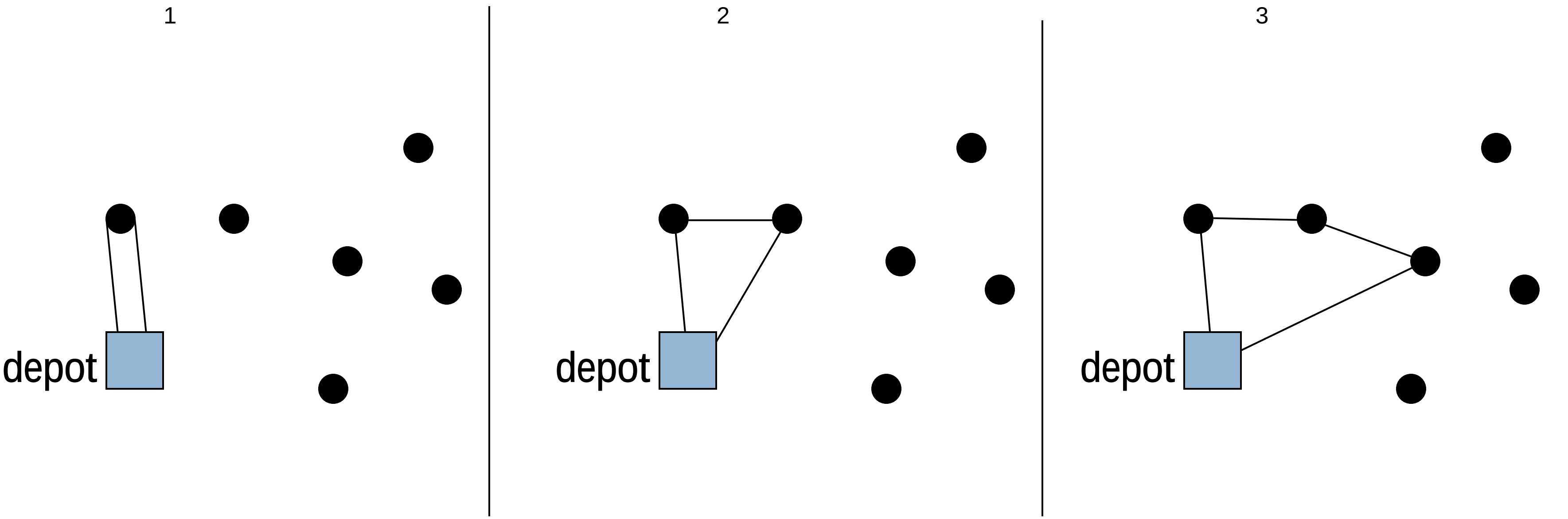
Nearest Neighbor

- Add nearest customer to end of the route.



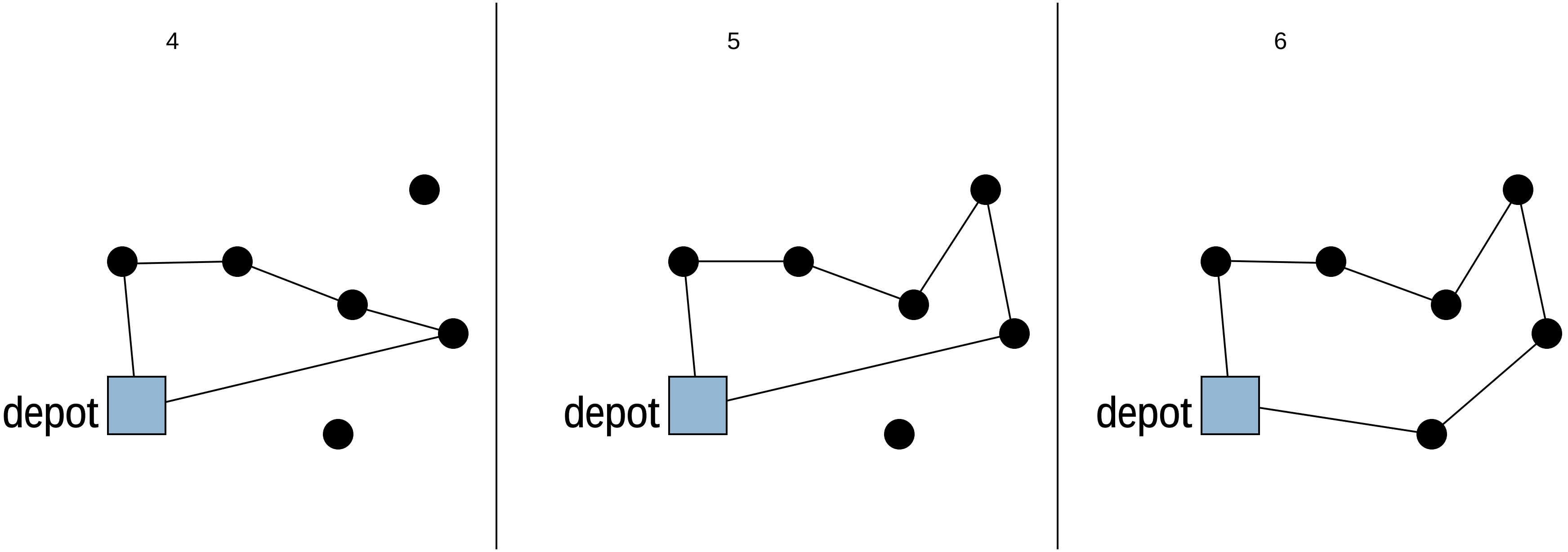
Nearest Insertion

- Insert customer closest to the route in the best sequence.



Nearest Insertion

- Insert customer closest to the route in the best sequence.



Savings Method

1. Select any city as the “depot” and call it city “0”.

   - Start with separate one stop routes from depot to each customer.

2. Calculate all *savings* for joining two customers and eliminating a trip back to the depot.

   $S_{ij} = C_{i0} + C_{0j} - C_{ij}$

3. Order savings from largest to smallest.

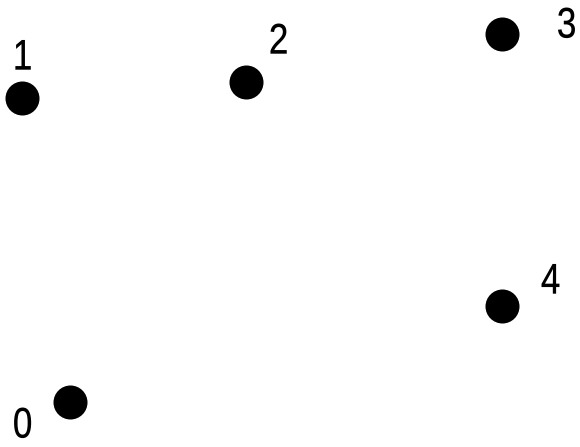
4. Form route by linking customers according to savings.

   - Do not break any links formed earlier.

   - Stop when all customers are on the route.

Savings Method Example

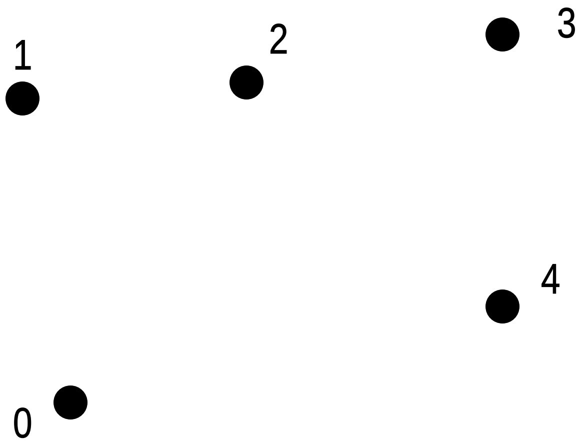
Given 5 customers and the costs (distances) between them.



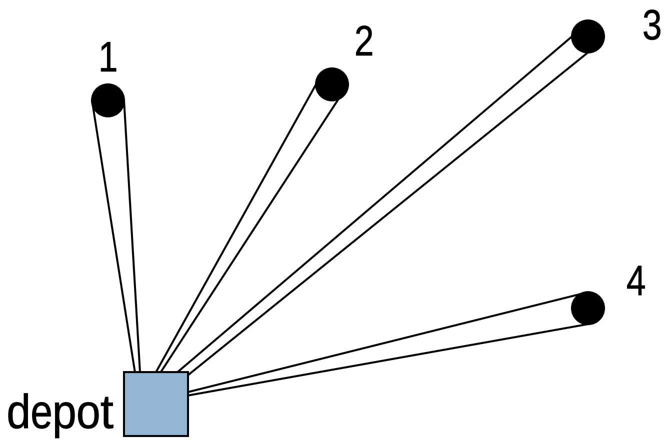
		j				
	c_{ij}	0	1	2	3	4
	0	-	8	9	13	10
	1	8	-	4	11	13
i	2	9	4	-	5	8
	3	13	11	5	-	7
	4	10	13	8	7	-

Savings Method Example

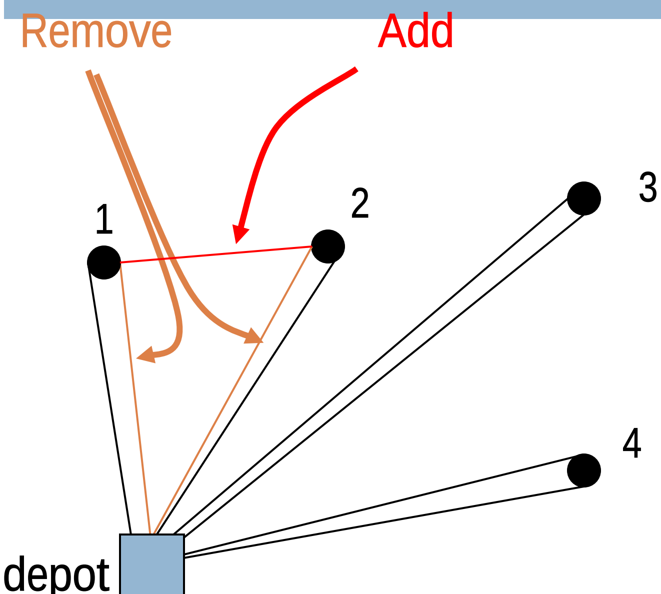
Given 5 customers, select the lower left as the depot.



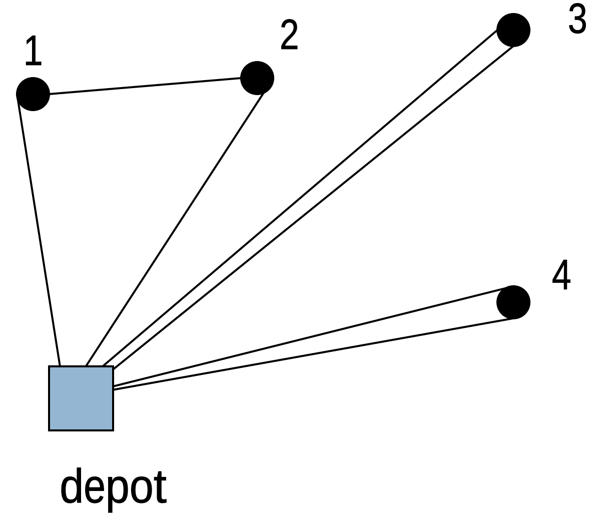
Conceptually form routes from the depot to each customer.



Savings Method: S_{12}



$$\textit{Savings} = S_{12} = C_{10} + C_{02} - C_{12}$$
$$= 8 + 9 - 4 = 13$$



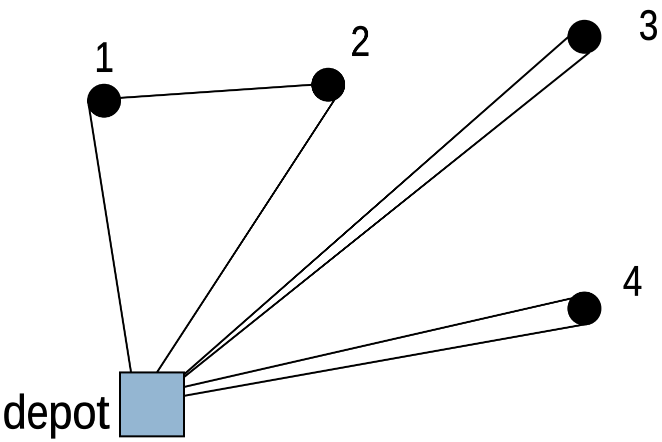
		j				
	c_{ij}	0	1	2	3	4
	0	-	8	9	13	10
	1	8	-	4	11	13
i	2	9	4	-	5	8
	3	13	11	5	-	7
	4	10	13	8	7	-

Savings Method

$$S_{12} = C_{10} + C_{02} - C_{12}$$

Note: $S_{21} = C_{20} + C_{01} - C_{21}$

so $S_{12} = S_{21}$



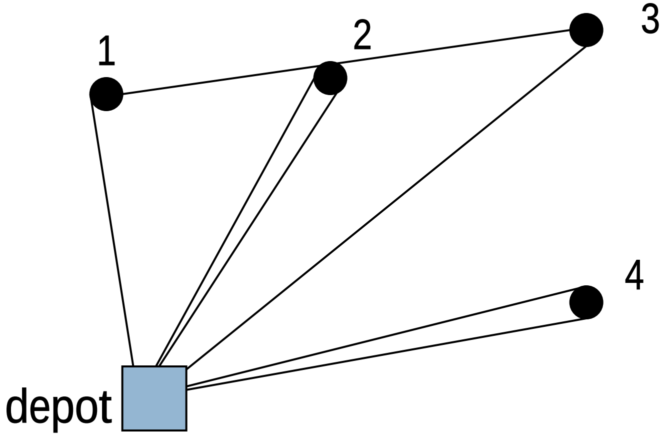
If problem is symmetric, then $s_{ij} = s_{ji}$, so $s_{21} = s_{12}$, $s_{32} = s_{23}$, etc. There are (n-1)(n-2)/2 savings to calculate.

If problem is asymmetric, then all s_{ij}'s must be calculated. There are (n-1)(n-2) savings to calculate.

Savings Method: S_{13}

$S_{13} = C_{10} + C_{03} - C_{13}$

$= 8 + 13 - 11 = 10$

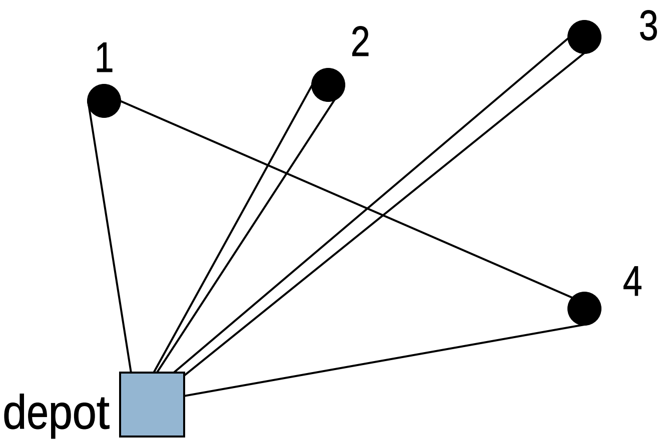


	c_{ij}	j: 0	1	2	3	4
	0	-	8	9	13	10
	1	8	-	4	11	13
i	2	9	4	-	5	8
	3	13	11	5	-	7
	4	10	13	8	7	-

Savings Method: S_{14}

$S_{14} = C_{10} + C_{04} - C_{14}$

$= 8 + 10 - 13 = 5$

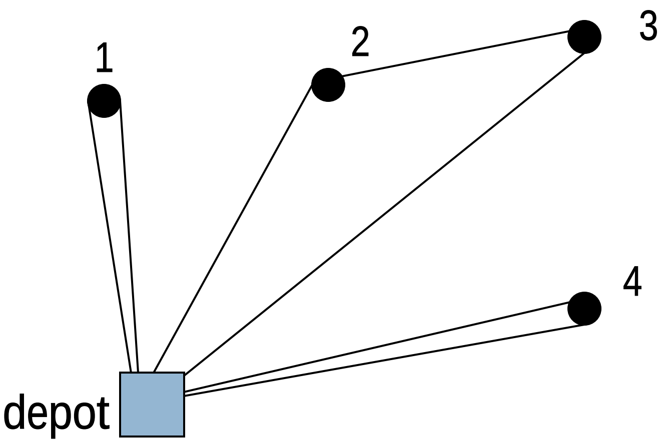


	c_{ij}	0	1	2	3	4
	0	-	8	9	13	10
	1	8	-	4	11	13
i	2	9	4	-	5	8
	3	13	11	5	-	7
	4	10	13	8	7	-

Savings Method: S_{23}

$S_{23} = C_{20} + C_{03} - C_{23}$

$= 9 + 13 - 5 = 17$

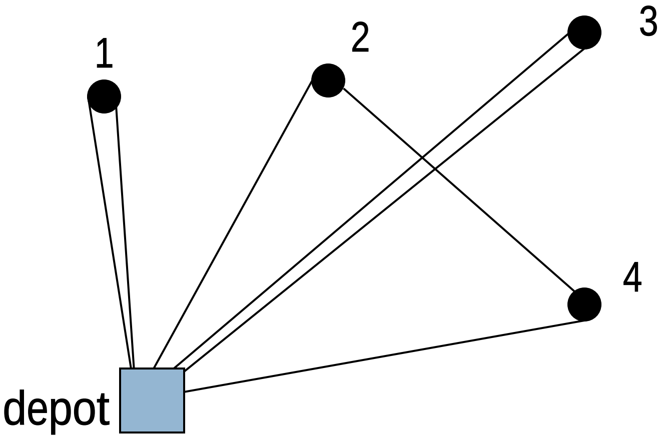


		j				
	c_{ij}	0	1	2	3	4
	0	-	8	9	13	10
	1	8	-	4	11	13
i	2	9	4	-	5	8
	3	13	11	5	-	7
	4	10	13	8	7	-

Savings Method: S_{24}

$S_{24} = C_{20} + C_{04} - C_{24}$

$= 9 + 10 - 8 = 11$

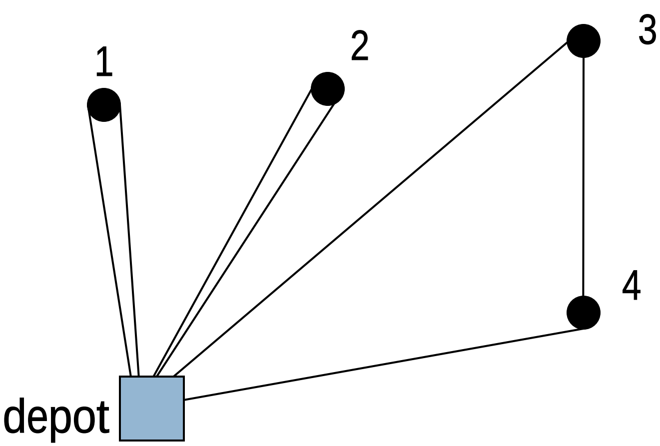


				j		
	c_{ij}	0	1	2	3	4
	0	-	8	9	13	10
	1	8	-	4	11	13
i	2	9	4	-	5	8
	3	13	11	5	-	7
	4	10	13	8	7	-

Savings Method: S_{34}

$S_{14} = C_{30} + C_{04} - C_{34}$

$= 13 + 10 - 7 = 16$



		j				
	c_{ij}	0	1	2	3	4
	0	-	8	9	13	10
	1	8	-	4	11	13
i	2	9	4	-	5	8
	3	13	11	5	-	7
	4	10	13	8	7	-

Savings Method

- Order savings from largest to smallest.

S_{23} (= S_{23}) = 17

S_{34} (= S_{43}) = 16

S_{12} (= S_{21}) = 13

S_{24} (= S_{42}) = 11

S_{13} (= S_{31}) = 10

S_{14} (= S_{41}) = 5

Savings Method

- Form route by linking customers according to savings.

S_{23}

S_{34}

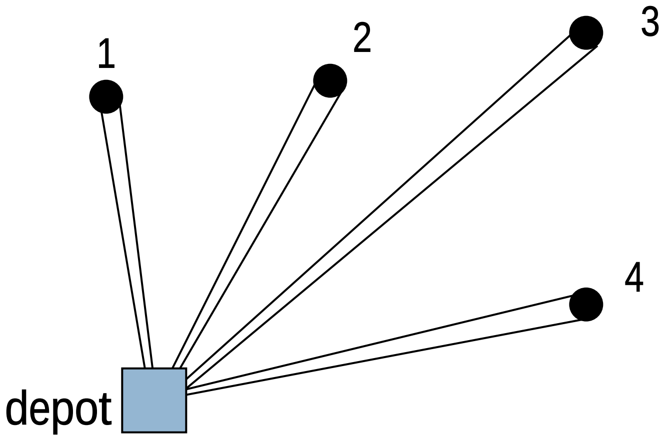
S_{12}

S_{24}

S_{13}

S_{14}

Link 2 and 3.



Savings Method

- Form route by linking customers according to savings.

S_{23} 0-2-3-0

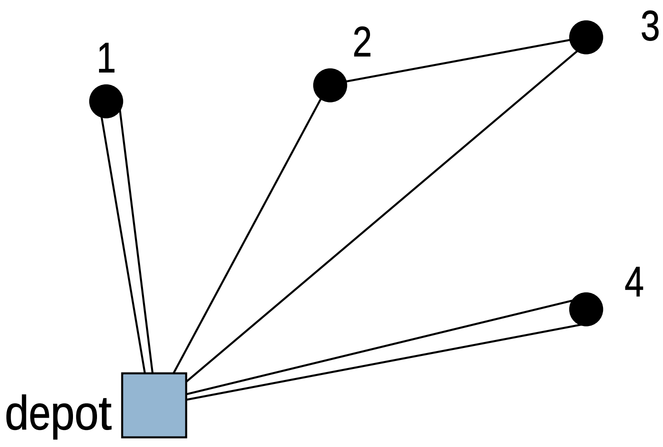
S_{34}

S_{12}

S_{24}

S_{13}

S_{14}



Savings Method

- Form route by linking customers according to savings.

S_{23} 0-2-3-0

S_{34}

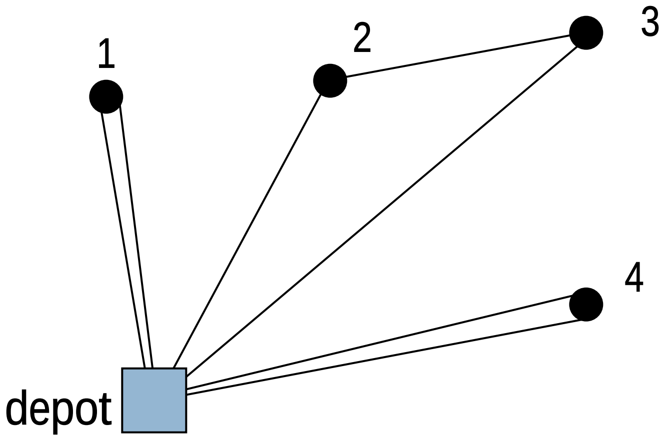
S_{12}

S_{24}

S_{13}

S_{14}

Link 3 and 4.
Do not break earlier links.



Savings Method

- Form route by linking customers according to savings.

S_{23} 0-2-3-0

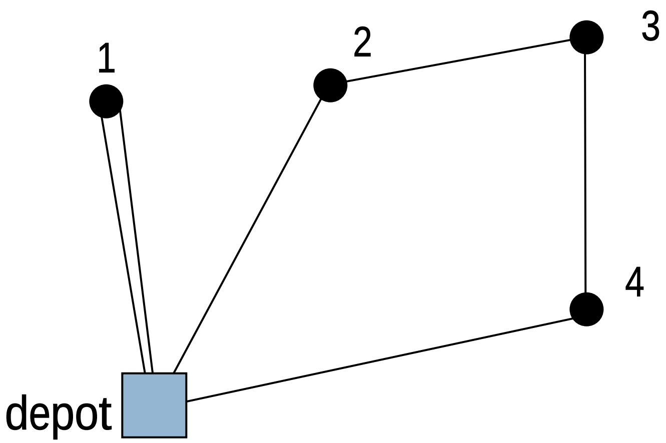
S_{34} 0-2-3-4-0

S_{12}

S_{24}

S_{13}

S_{14}



Savings Method

- Form route by linking customers according to savings.

S_{23} 0-2-3-0

S_{34} 0-2-3-4-0

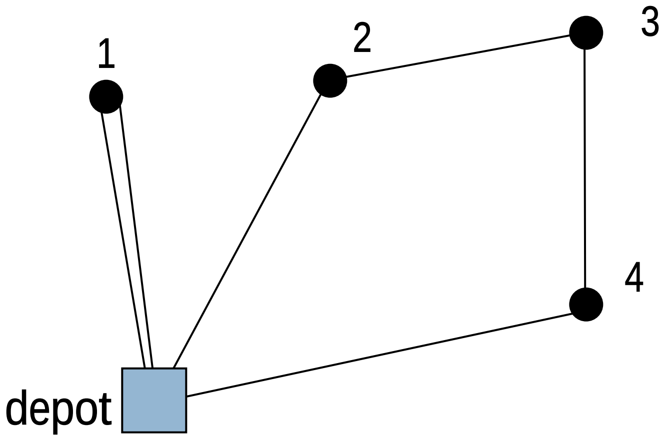
S_{12}

S_{24}

S_{13}

S_{14}

Link 1 and 2.
Do not break earlier links.



Savings Method

- Form route by linking customers according to savings.

S_{23} 0-2-3-0

S_{34} 0-2-3-4-0

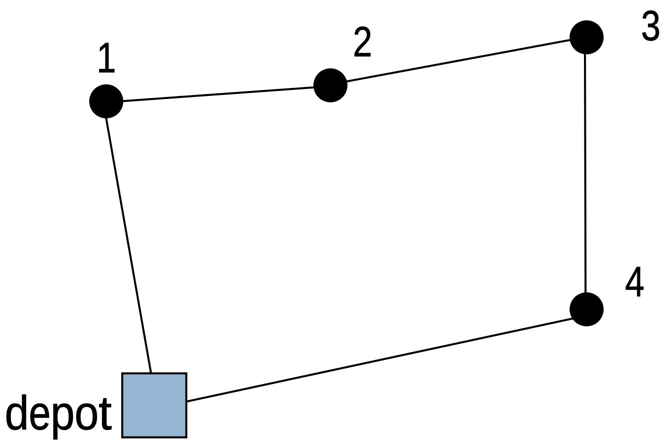
S_{12} 0-1-2-3-4-0

S_{24}

S_{13}

S_{14}

Done!



Route Improvement Heuristics

- Start with a feasible route.

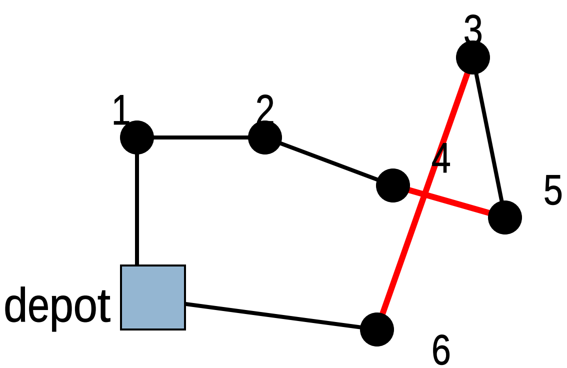
- Make changes to improve route.
  - Exchange heuristics.
    - Switch position of one customer in the route.
    - Switch 2 arcs in a route.
    - Switch 3 arcs in a route.

  - Local search methods.
    - Simulated Annealing.
    - Tabu Search.
    - Genetic Algorithms.

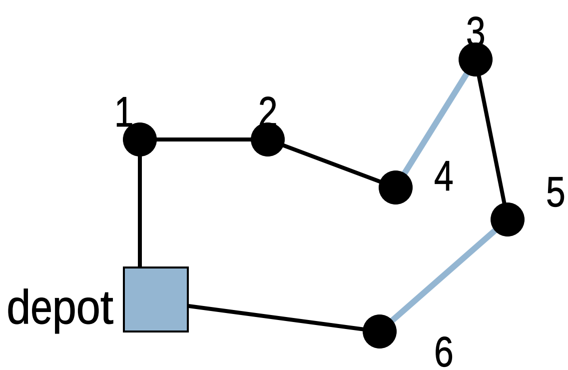
K-opt Exchange

- Replace k arcs in a given TSP tour by k new arcs, so the result is still a TSP tour.
- 2-opt: Replace 4-5 and 3-6 by 4-3 and 5-6.

Original TSP tour



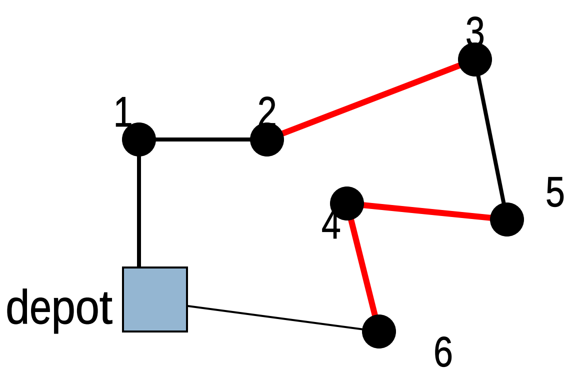
Improved TSP tour



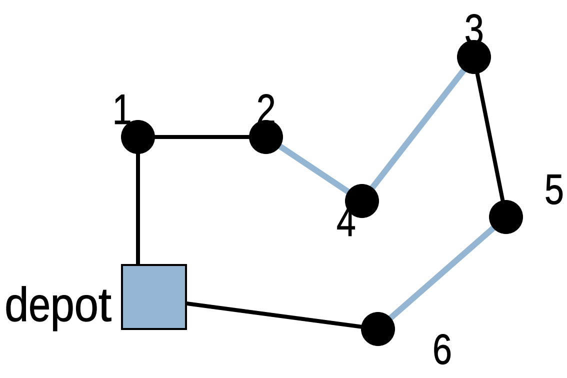
3-opt Exchange

- 3-opt: Replace 2-3, 5-4 and 4-6 by 2-4, 4-3 and 5-6.

Original TSP tour



Improved TSP tour



Capacitated VRP

From TSP to VRP

- TSP = VRP with 1 vehicle of infinite capacity
- The VRP extends the TSP for multiple vehicles.

Capacitated VRP

- Given a depot and a set of customers, find a set of minimum cost depot returning vehicle routes to service all customers (each customer must be served only once by exactly one vehicle).
  - Multiple capacitated vehicles
  - Minimize traveling distance

Solving VRPs

- VRP is a very hard problem to solve
  - *NP Hard* in the strong sense
  - Exact solutions only for small problems (20-50 customers)

  - Most solution methods are heuristic
  - Most operate as:
    - Construct
    - Improve

Math Programming Approaches

$$\text{minimise}: \sum_{i,j} c_{ij} \sum_k x_{ijk}$$

subject to

$$\sum_i \sum_k x_{ijk} = 1 \quad \forall j$$

$$\sum_j \sum_k x_{ijk} = 1 \quad \forall i$$

$$\sum_j \sum_k x_{ihk} - \sum_j \sum_k x_{hjk} = 0 \quad \forall k, h$$

$$\sum_i q_i \sum_j x_{ijk} \leq Q_k \quad \forall k$$

$$\{x_{ijk}\} \subseteq S$$

$$x_{ijk} \in \{0,1\}$$

- Advantages
  - Can find the optimal solution
- Disadvantages
  - Only works for small problems
  - One extra constraint → back to the drawing board

Heuristic Column Generation

Columns represent routes

Column/route cost c_k

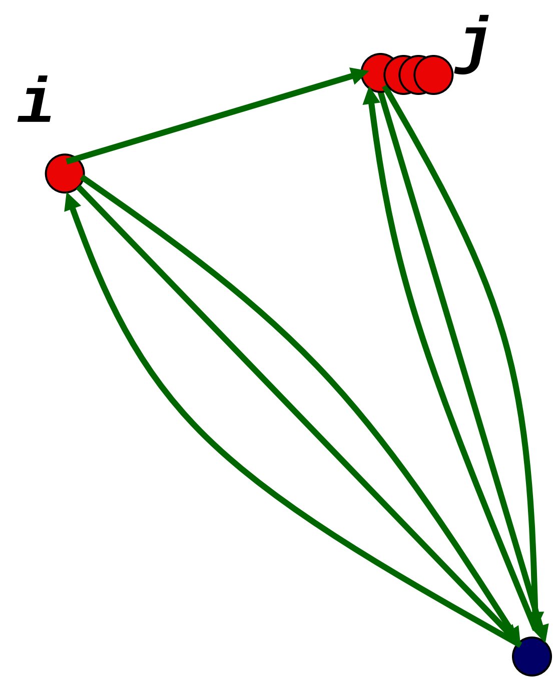
Rows represent customers

Array entry $a_{ik} = 1$ iff customer i is covered by route k

	89	76	99	45	32	
1	1	1	1	0	0	…
2	0	1	1	0	1	…
3	0	0	0	0	0	…
4	1	0	1	1	0	…
5	1	0	0	0	0	…

Heuristic methods - Construction

- Savings method (Clarke & Wright 1964)
  - Calculate S_{ij} for all i, j
  - Consider cheapest S_{ij}
  - If j can be appended to i
    - merge them to new i
    - update all S_{ij}
  - else
    - delete S_{ij}
  - Repeat

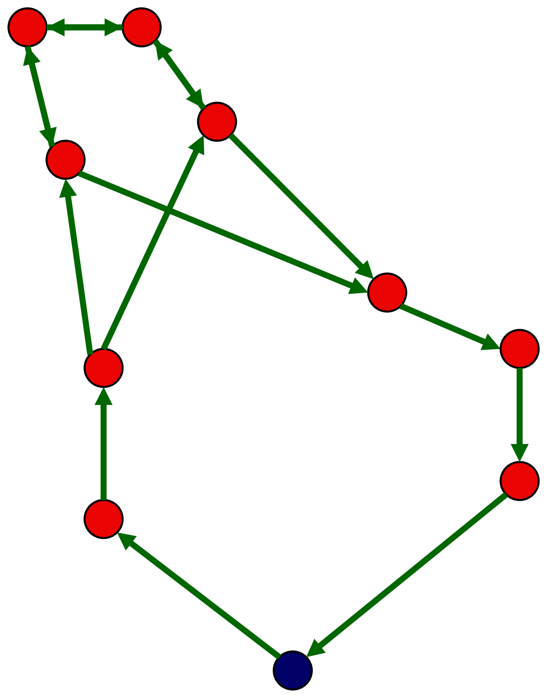


Improvement Methods

Edge-exchange Intra- and Inter-route Neighbourhood Structures

For example, **2-opt (3-opt, 4-opt…):**

- Remove 2 arcs
- Replace with 2 others

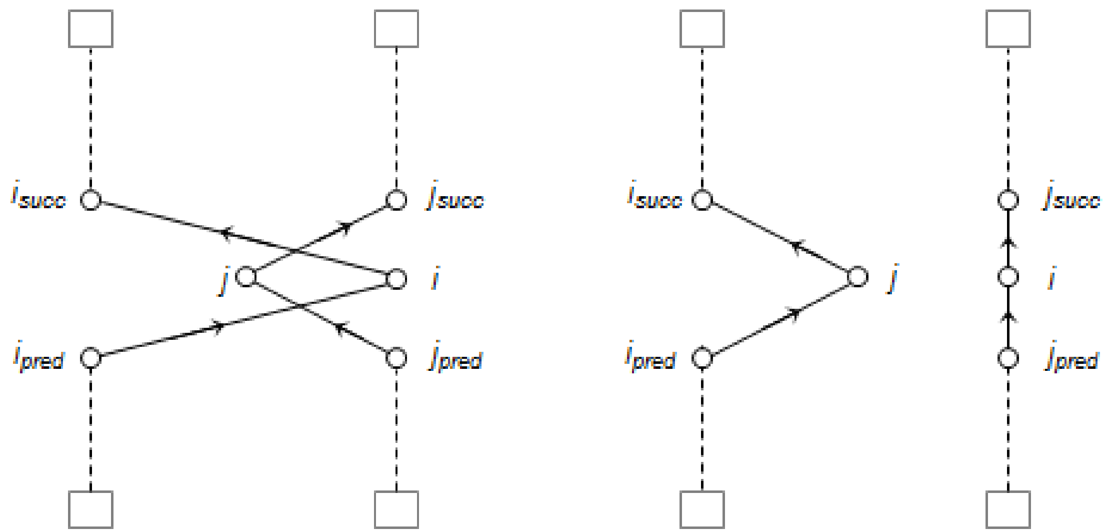


Improvement Methods

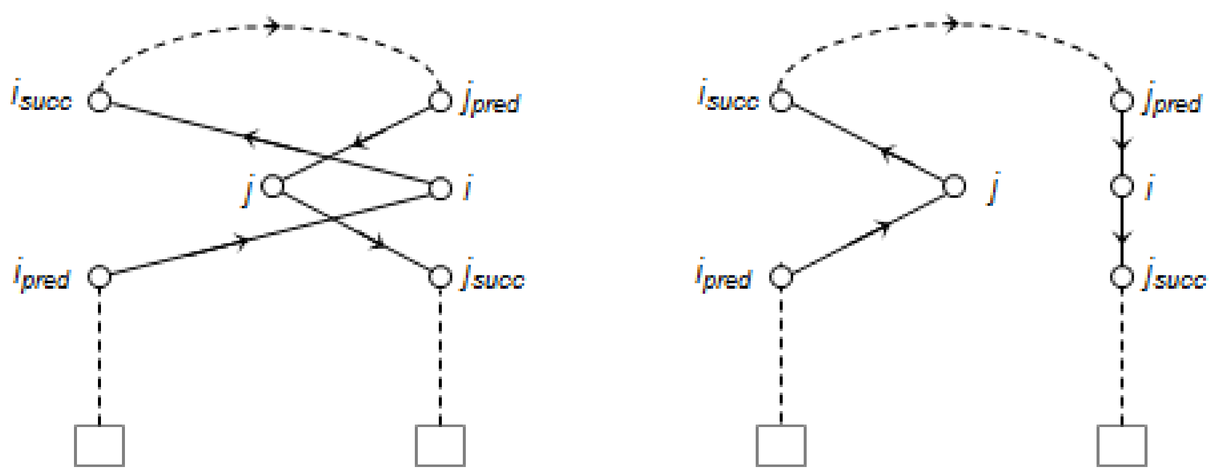
1-1 Exchange

(swap)

Inter-route



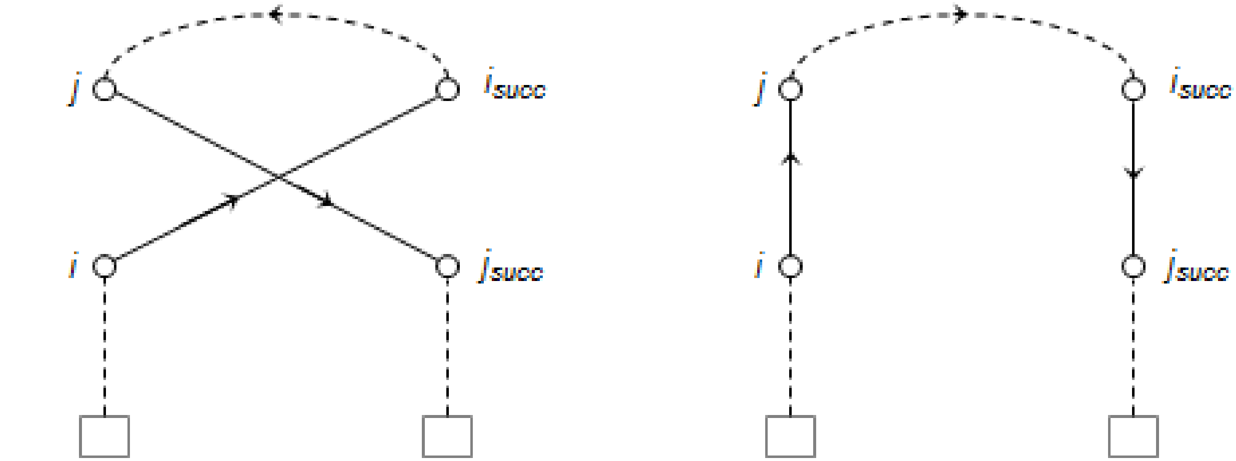
Intra-route



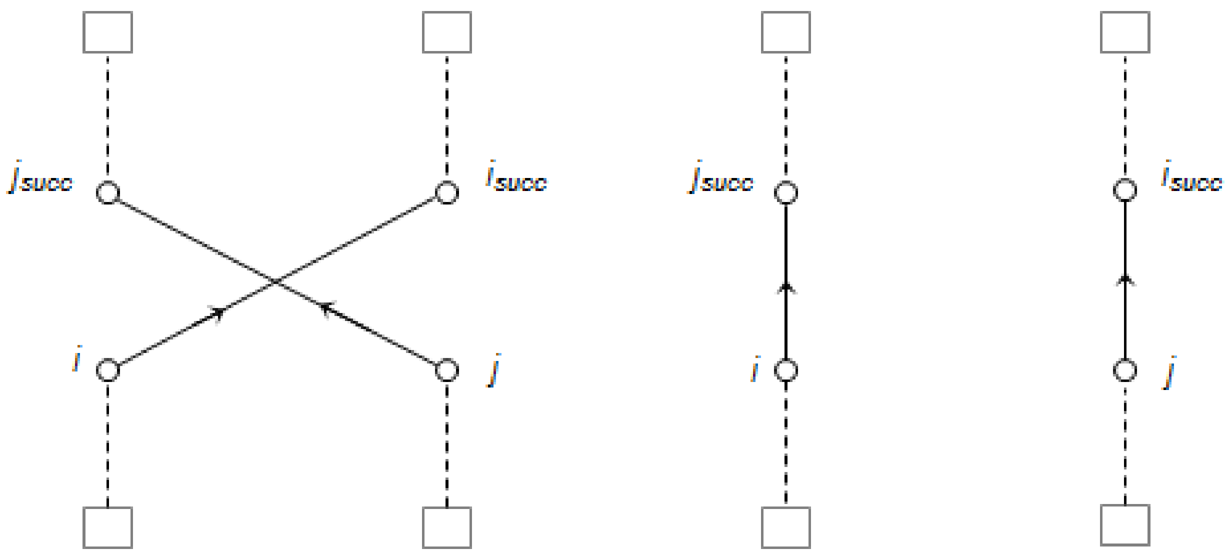
Improvement Methods

2-OPT

Intra-route



Inter-Route

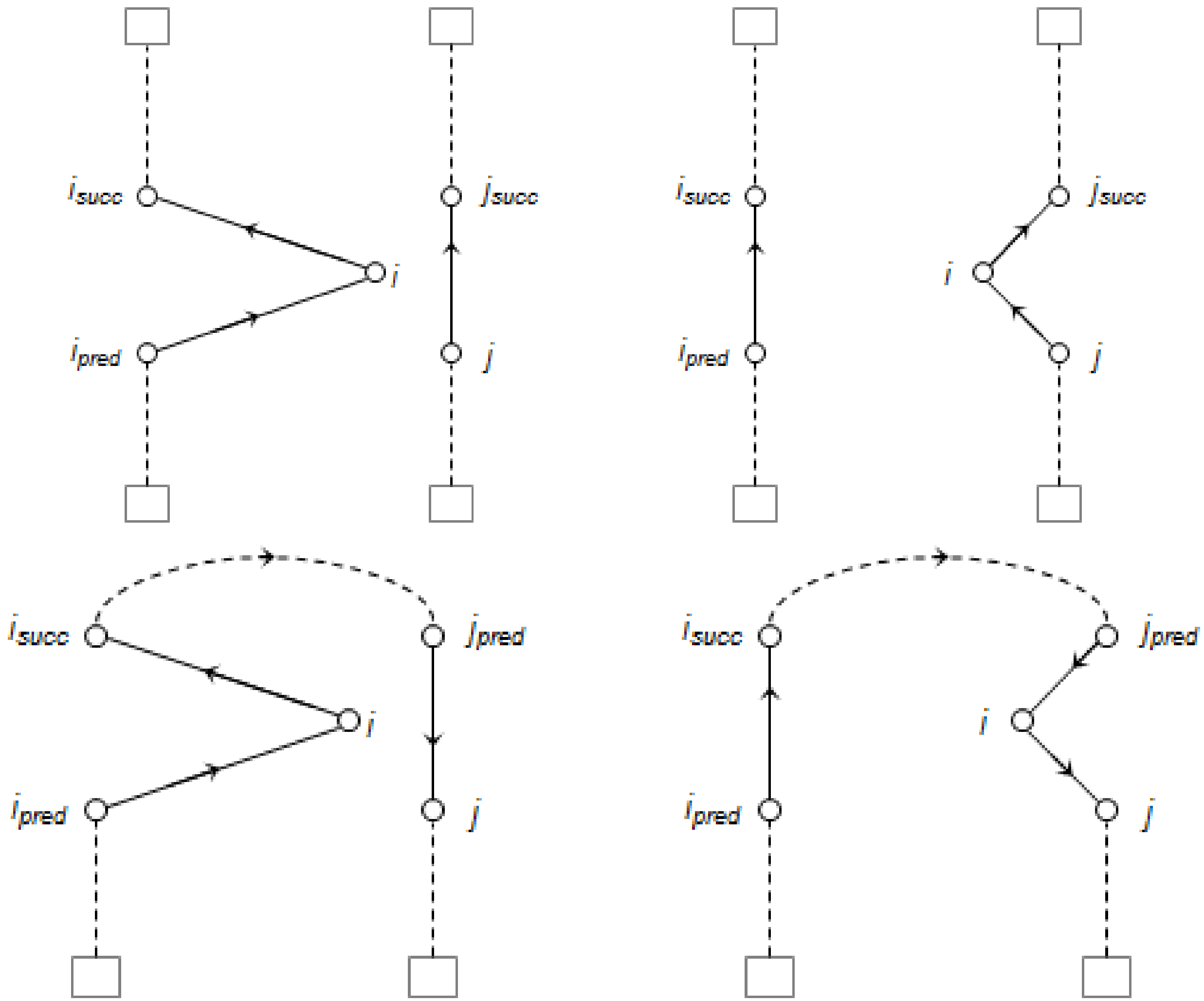


Improvement Methods

0-1 Relocate

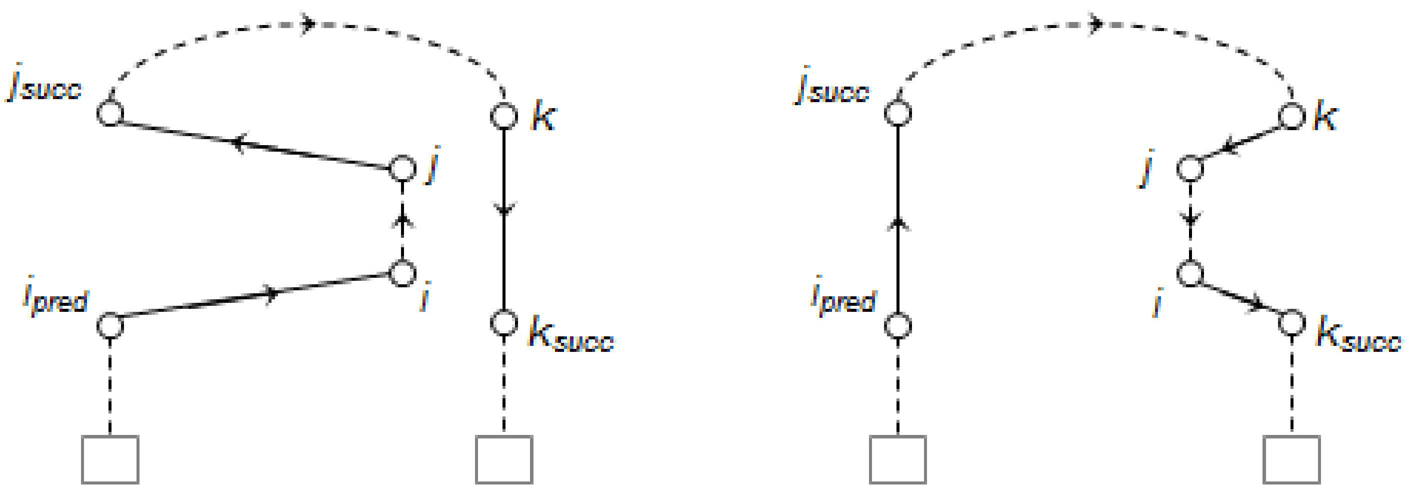
Inter-route

Intra-route



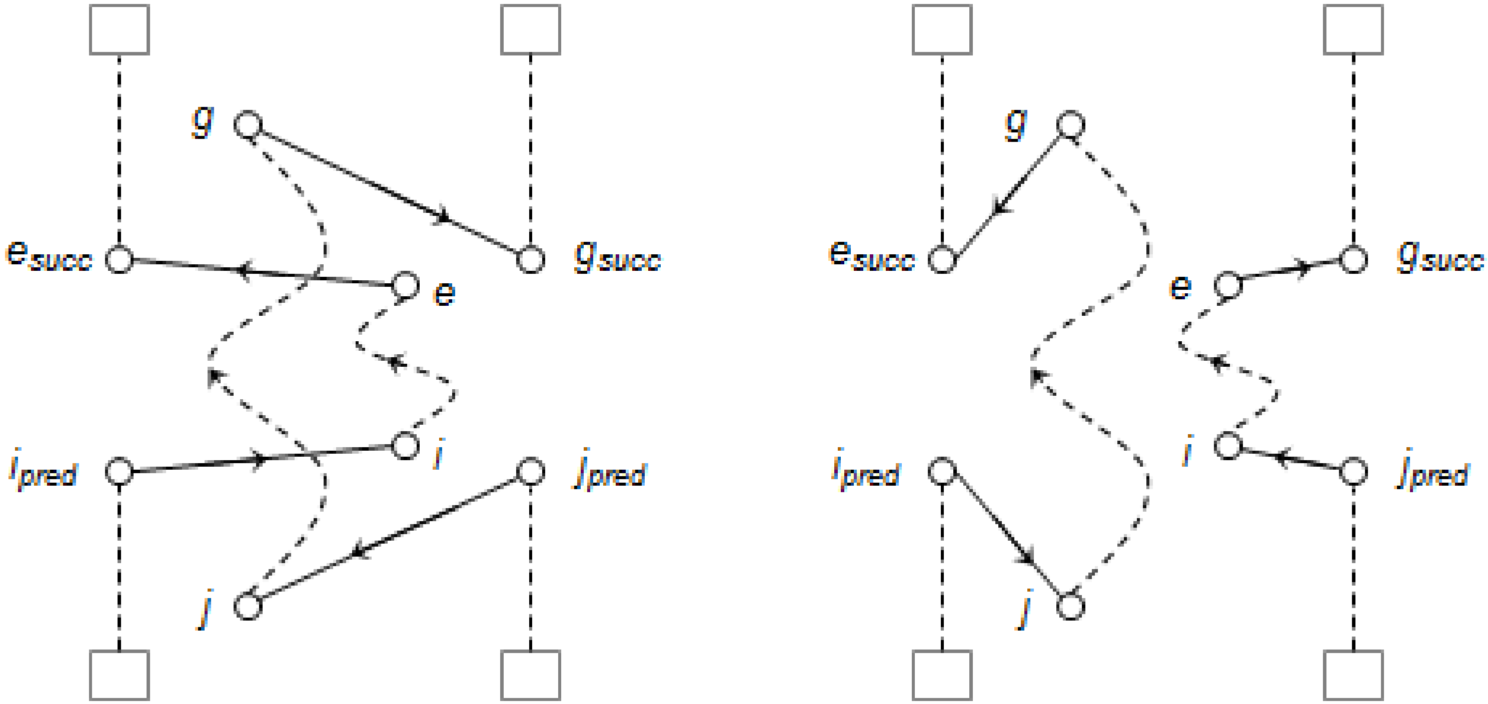
Improvement Methods

I-OPT



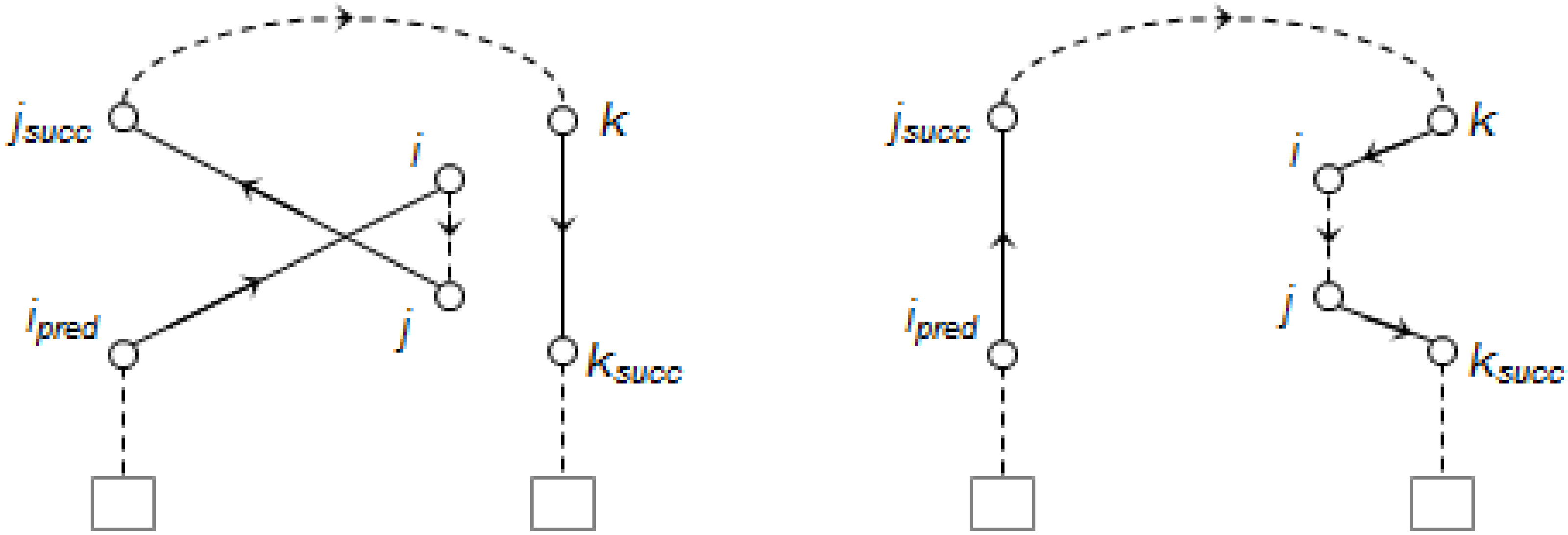
Improvement Methods

CROSS



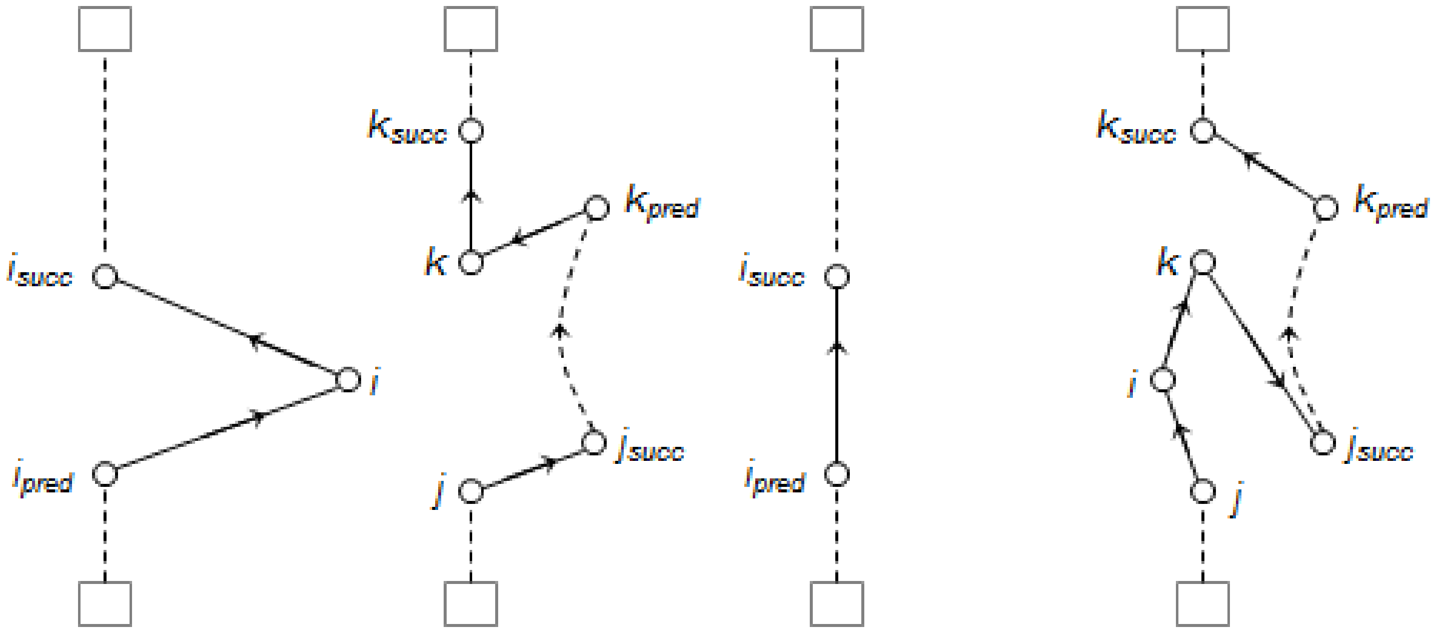
Improvement Methods

OR-OPT



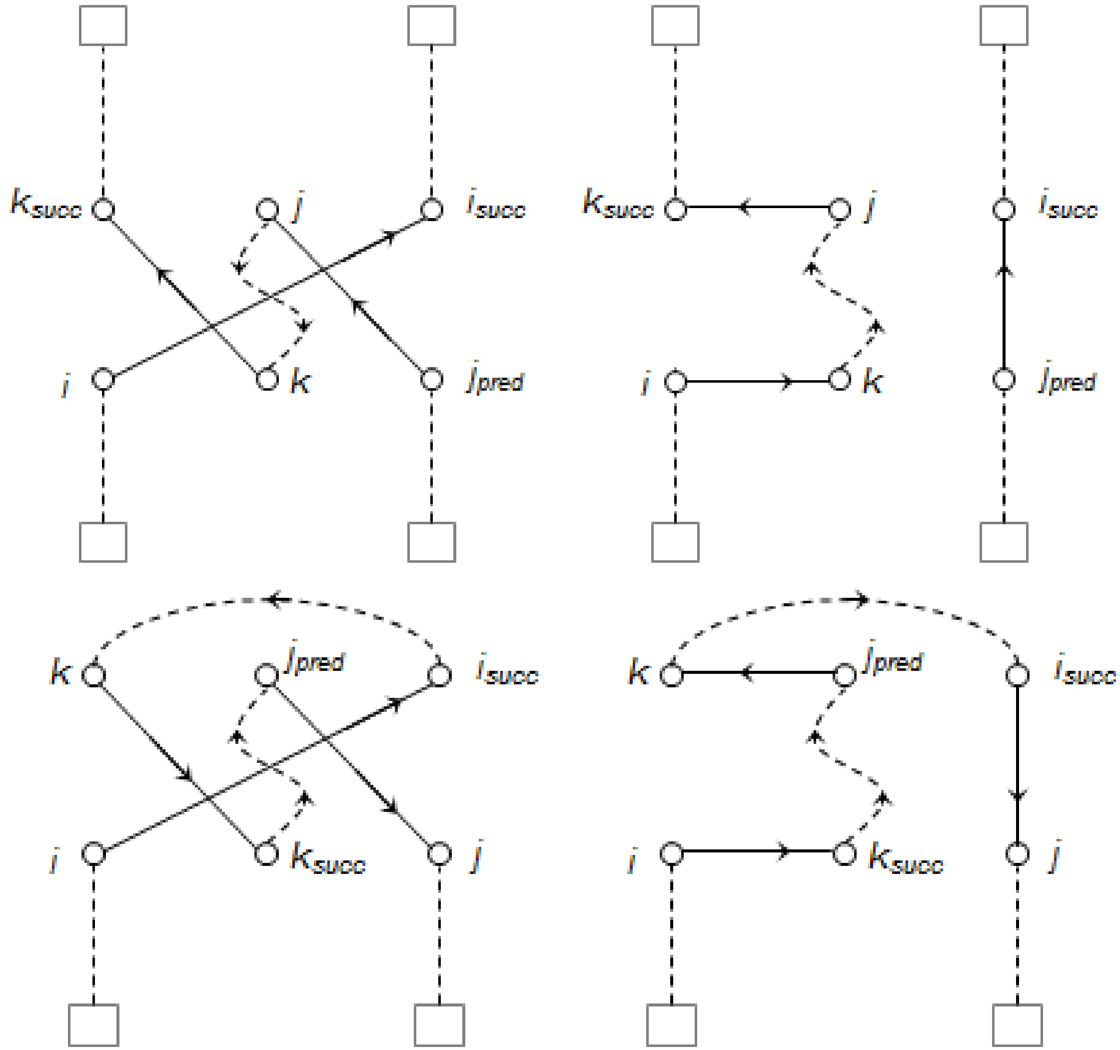
Improvement Methods

GENI



Improvement Methods

CROSSOVER-EXCHANGE



Improvement methods

Large Neighbourhood Search

= Destroy & Re-create

- Destroy part of the solution
  - Remove visits from the solution
- Re-create solution
  - Use insert method to re-insert customers
  - Different insert methods lead to new (better?) solutions
- If the solution is better, keep it
- Repeat

Improvement methods

Variable Neighborhood Search

- Consider multiple neighborhoods
- Find local minimum in smallest neighborhood
- Advance to next-largest neighborhood
- Search current neighborhood
  - If a change is found, return to smallest neighborhood
  - Otherwise, advance to next-largest

Improvement methods

Path Relinking

- Applies where-ever a population of solutions is available
- Take one (good) solution A
- Take another (good) reference solution B
- Gradually transform solution A into solution B
  - pass through new solutions “between” A and B
  - new solutions contain traits of both A and B
  - should be good!

Improvement methods

Genetic Programming

- Simulate the Natural Selection

Evolutionary Algorithms

- Generate a population of solutions (construct methods)
- Evaluate fitness (objective)
- Create next generation:
  - Choose two solutions from population
  - Recombination - Combine them
  - (Mutate)
  - Produce offspring (calculate fitness)
  - (Improve)
  - Repeat until population doubles
- Apply selection:
  - Bottom half "dies"
- Repeat

Scaling

- **Solving problems with tens of thousands of nodes**
- Decompose problem
  - Split into smaller problems
- Limit search
  - Only consider inserting next to nearby nodes
  - Only consdier inserting into nearby routes

Solution Methods

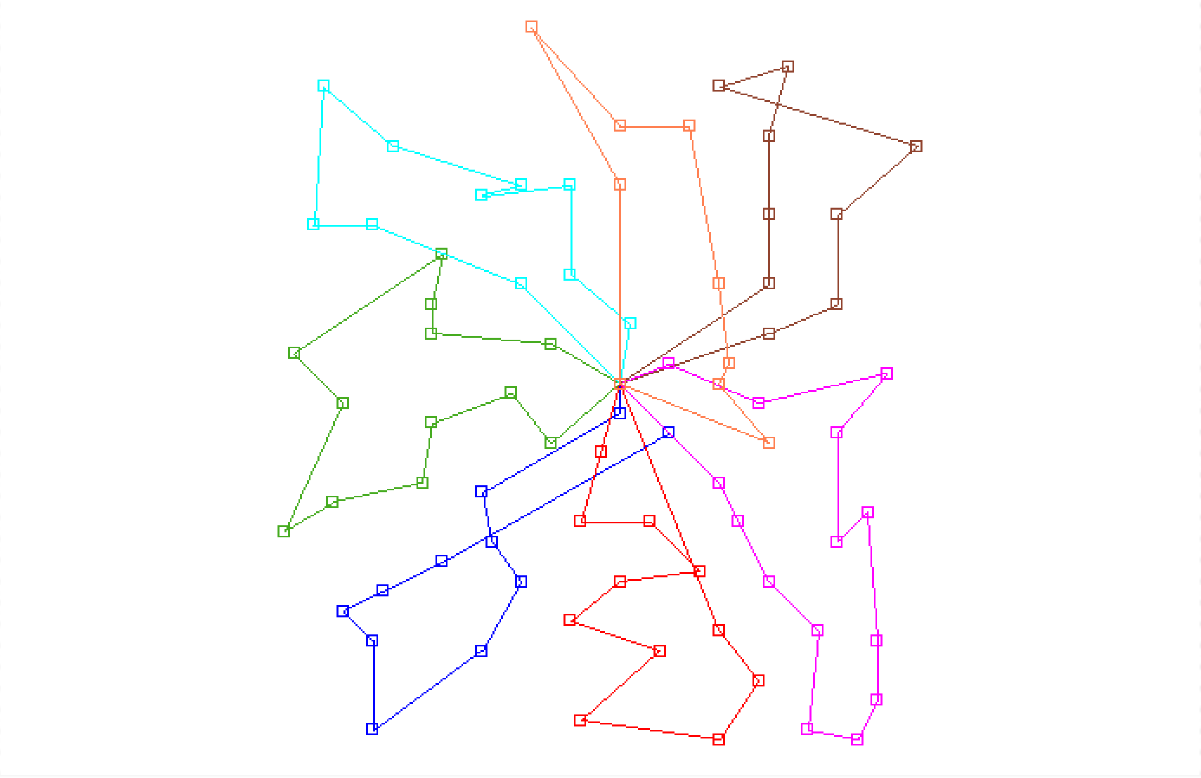
- **.. and the whole bag of tricks**

Other Metaheuristic Algorithms:

- Tabu Search
- Simulated Annealing
- Ants
- Bees
- ….

Hybrid Exact & Metaheuristic Algorithms

Rich VRP Variants



Capacitated VRP

- Homogeneous vehicles.
- One capacity (weight or volume).
- Minimize distance.
- No time windows or one time window per customer.
- No compatibility constraints.
- No DOT rules.

Rich VRPs

- Multiple vehicle types
  - Different fixed and variable traveling costs
- Multiple vehicle capacities
  - Weight, Cubic feet, Floor space, Value.
- Many different types of Costs:
  - Fixed charge
  - Variable costs per loaded mile & per empty mile
  - Waiting time; Layover time
  - Cost per stop (handling)
  - Loading and unloading cost
- Priorities for customers or orders

Rich VRPs

- Time windows for pickup and delivery.
  - Hard vs. soft
- Compatibility
  - Vehicles and customers.
  - Vehicles and orders.
  - Order types.
  - Drivers and vehicles.
- Driver rules (DOT)
  - Max drive duration = 10 hrs. before 8 hr. break.
  - Max work duration = 15 hrs. before 8 hr break.
  - Max trip duration = 144 hrs.

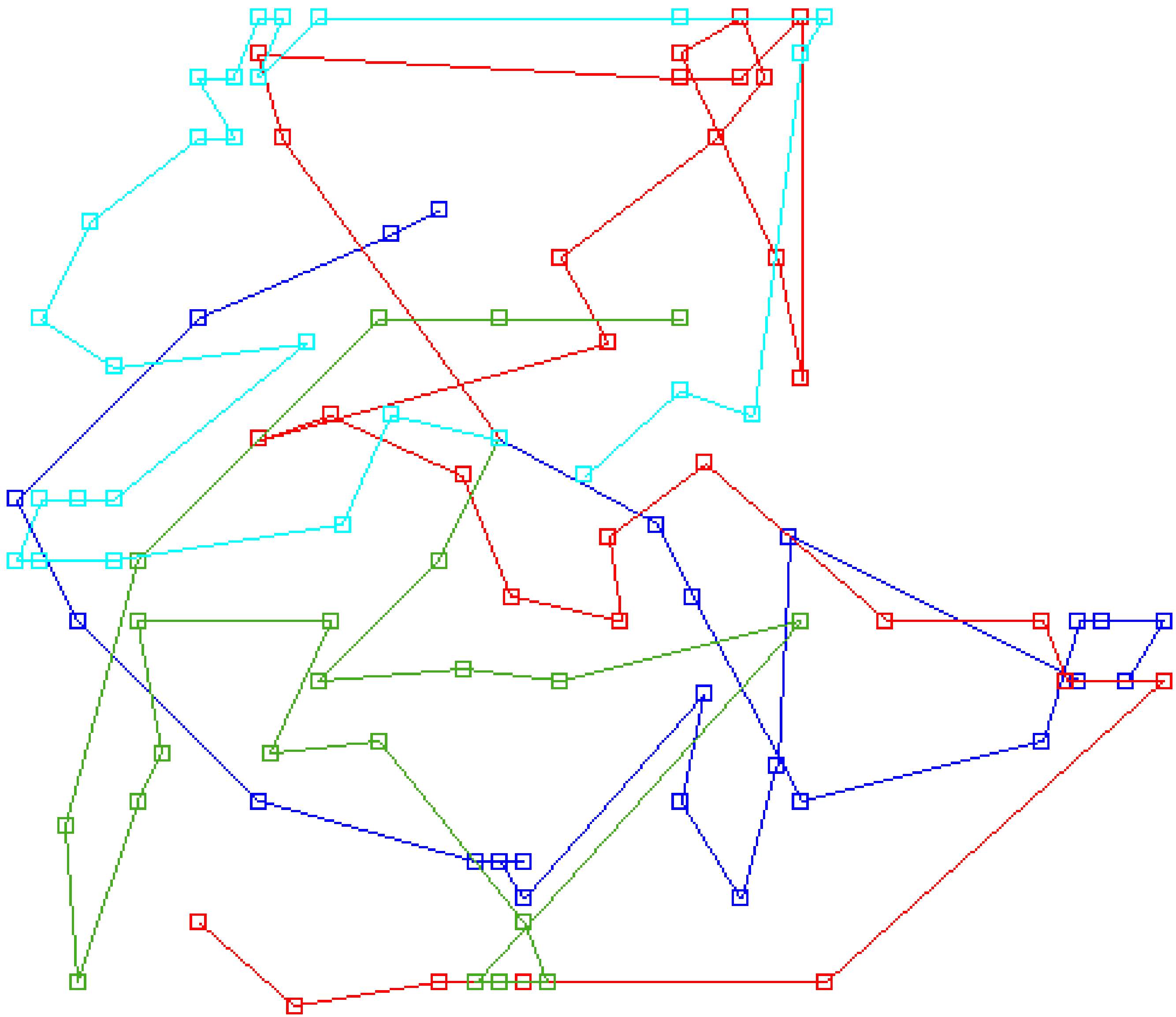
Time window constraints

- VRP with Time Window constraints
  - A window during which service can start (if the vehicle arrives earlier it waits at customer location)
  - E.g. only accept delivery 7:30am to 11:00am

  - Additional input data required
    - Duration of each customer visit
    - Time between *each pair* of customers

    - (Travel time can be vehicle-dependent or time-dependent)
  - Makes the route harder to visualise

Time Window constraints

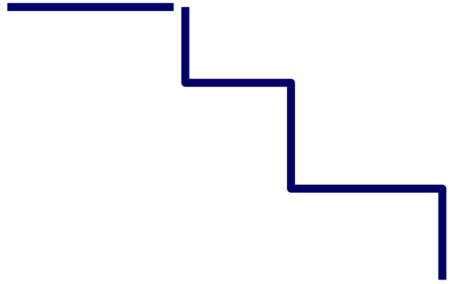


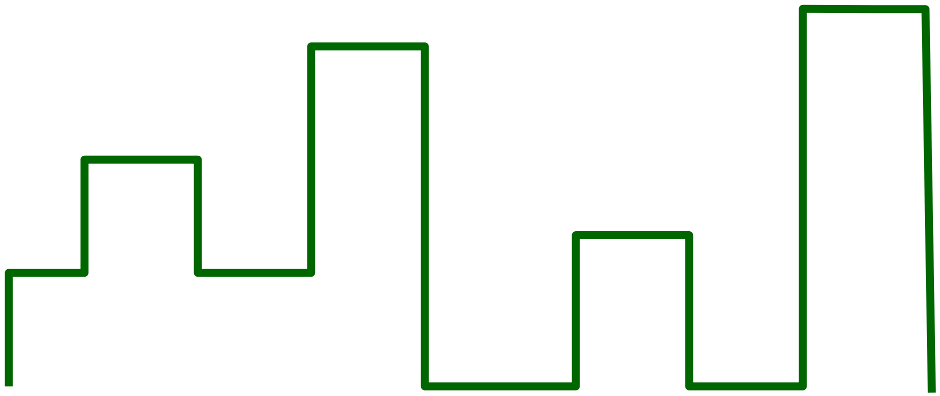
Pickup and Delivery problems

- Most routing considers delivery to/from a depot (depots)
- Pickup and Delivery problems consider FedEx style problem:

pickup at location A, deliver to location B

Load profiles





Pickup and Delivery problems

- PDPs have two implied constraints:
  - pickup is before delivery
  - pickup and delivery are on the same vehicle

- Usually, completely different methods used to solve this sort of problem
- Can be quite difficult
- Standard VRP is in effect a PDP with all stuff picked up at (delivered to) a depot. Not usually solved that way

Pickup and Delivery problem

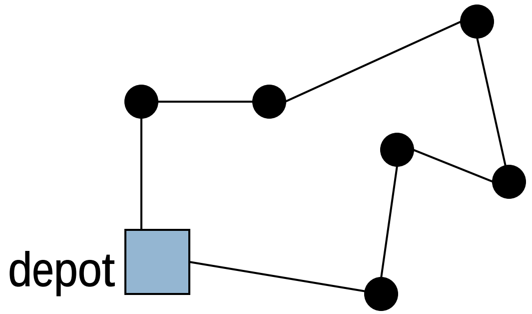
- Interesting variants
- Dial-a-ride problem:
  - Passenger transport
  - Like a multi-hire taxi
  - Pickup passenger A, pickup passenger B, drop off B, pickup up C, drop off A, ….
  - Ride-time constraints (e.g. max 1.4 x direct travel time)
- PDP can be used to model cross-docking
  - Pick up at Factory, Deliver to DC;
    Pickup at DC, Deliver to customer
  - Constraint: "Deliver to DC" before "Pickup at DC"

Pure Pickup or Delivery

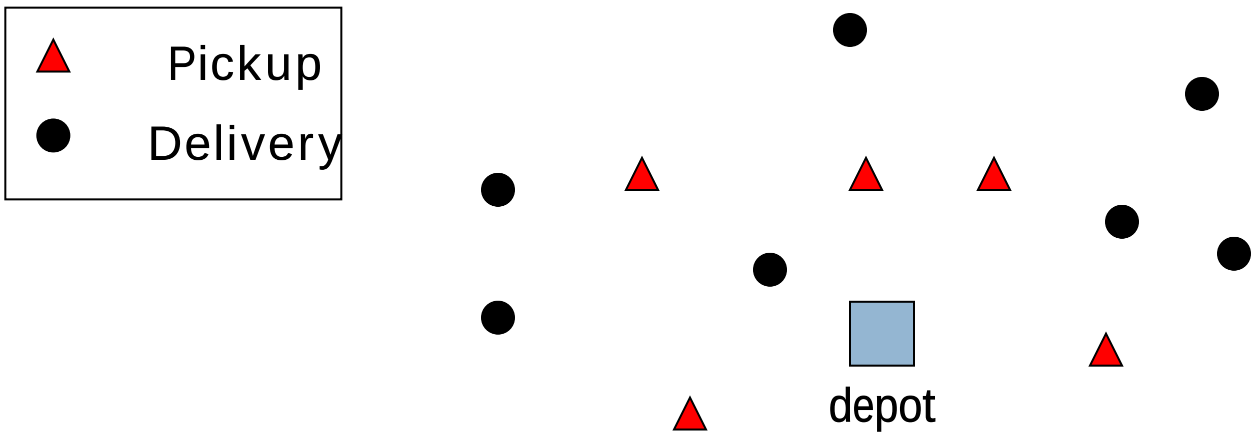
- **Delivery**: Load vehicle at depot. Design route to deliver to many customers (destinations).

- **Pickup**: Design route to pickup orders from many customers and deliver to depot.

- Examples:
  - UPS, FedEx, etc.
  - Manufacturers & carriers.
  - Carpools, school buses, etc.

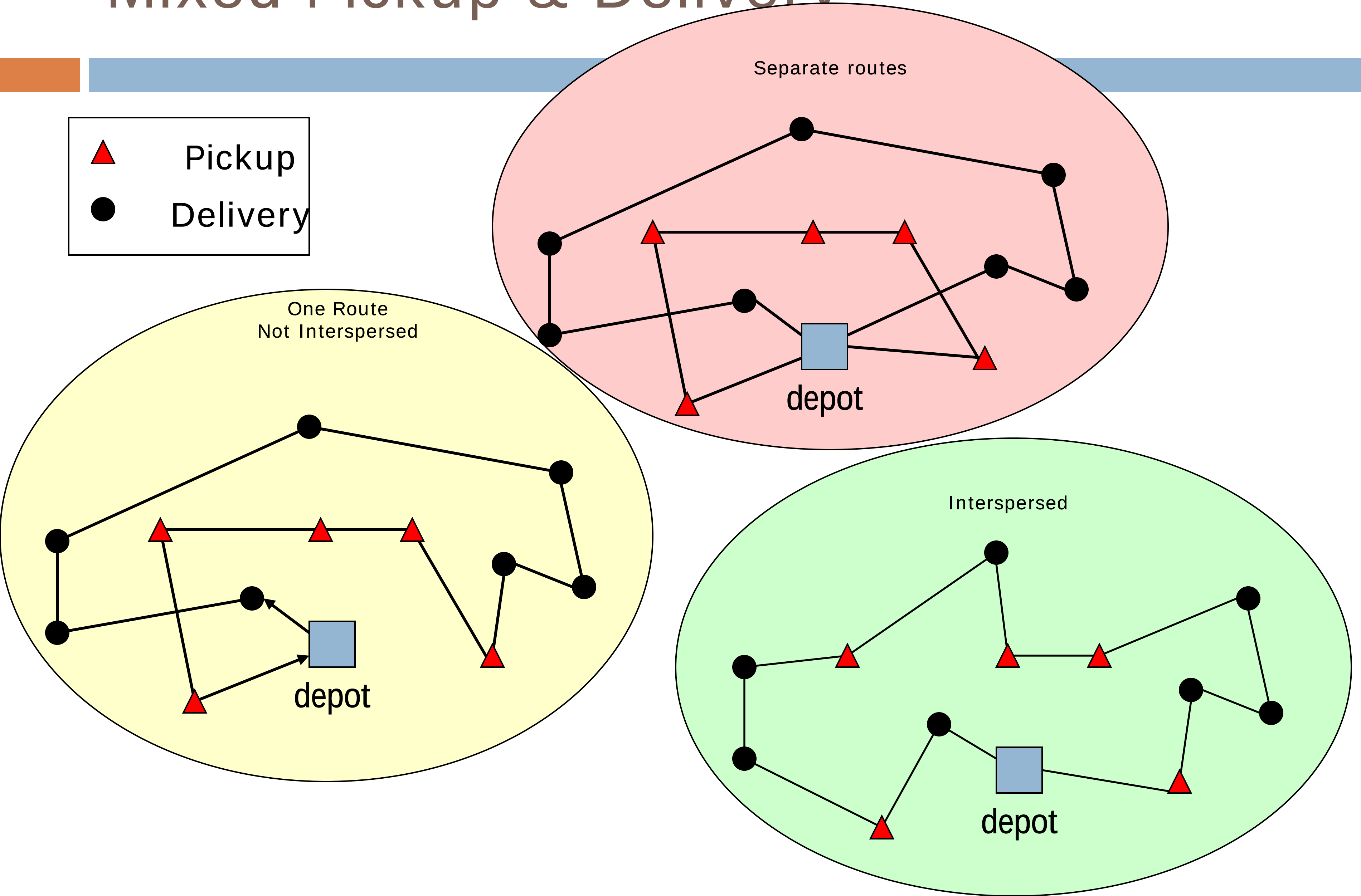


Mixed Pickup & Delivery



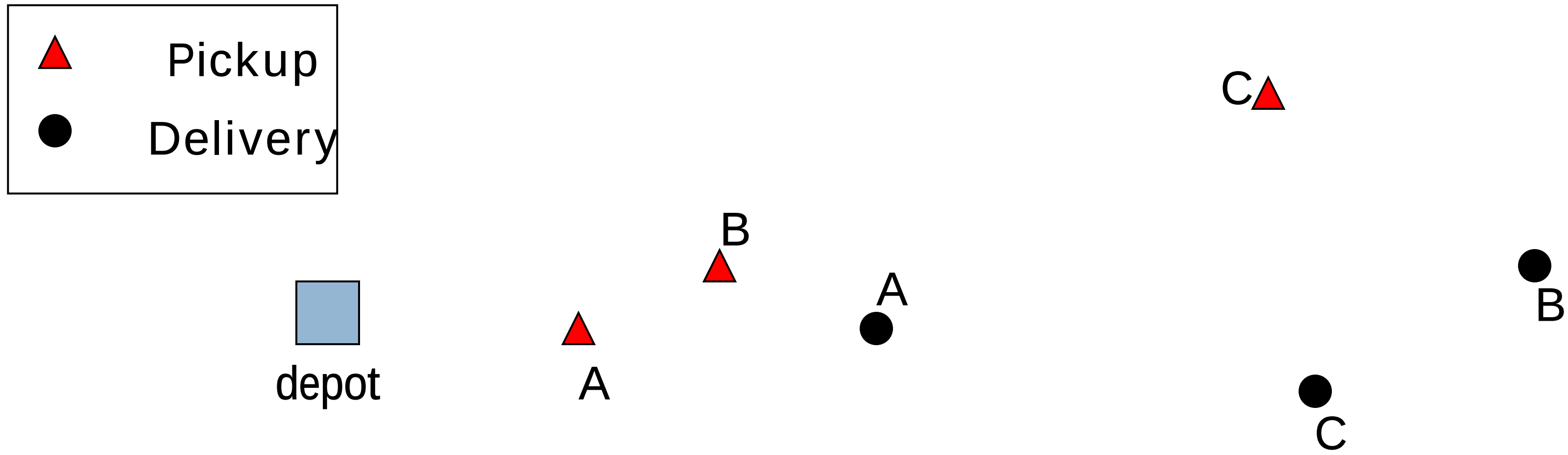
- Can pickups and deliveries be made on same trip?
- Can they be interspersed?

Mixed Pickup & Delivery



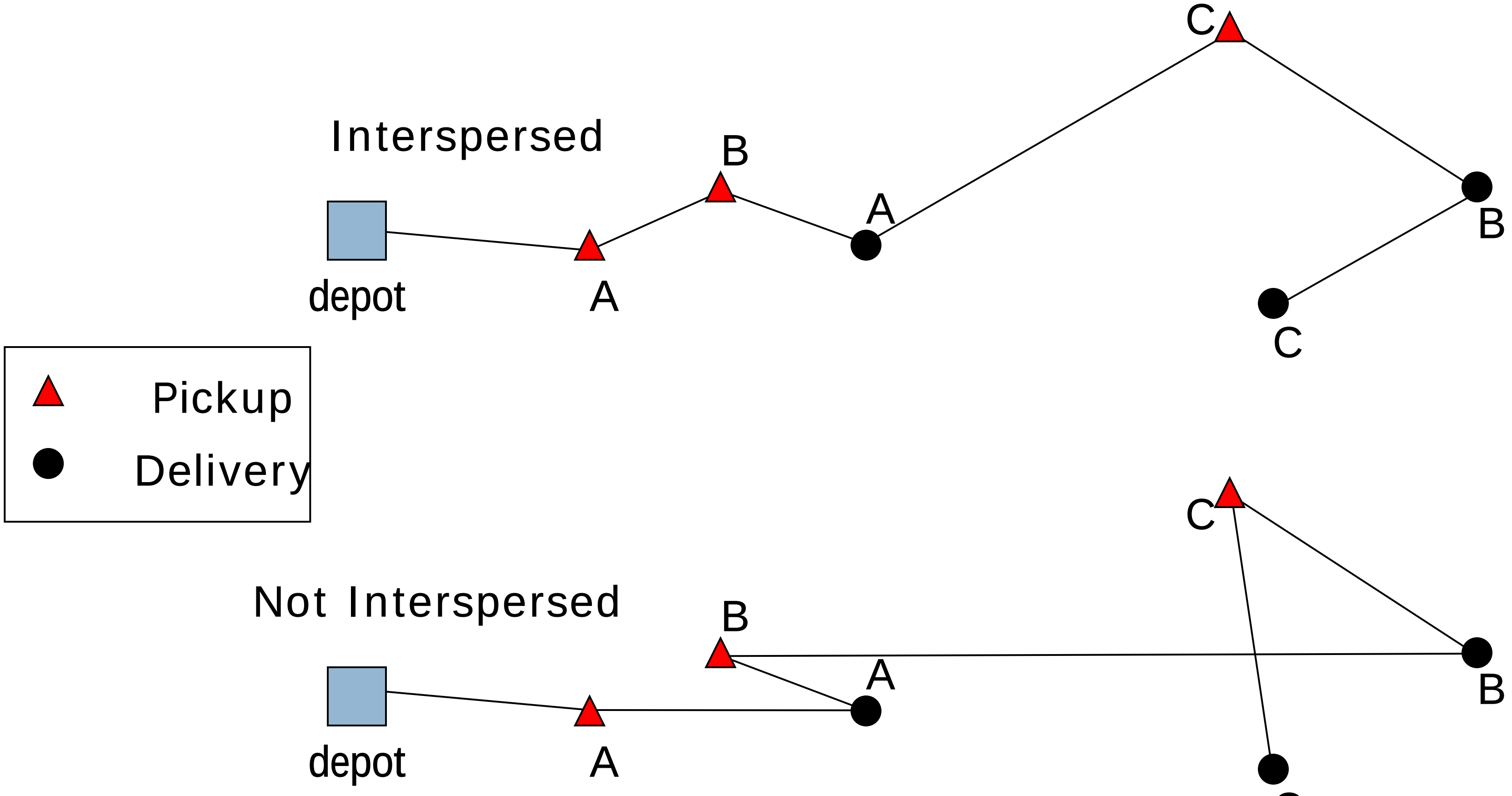
Pickup-Delivery Problems

- Pickup at one or more origin and delivery to one or more destinations.
- Often long haul trips.



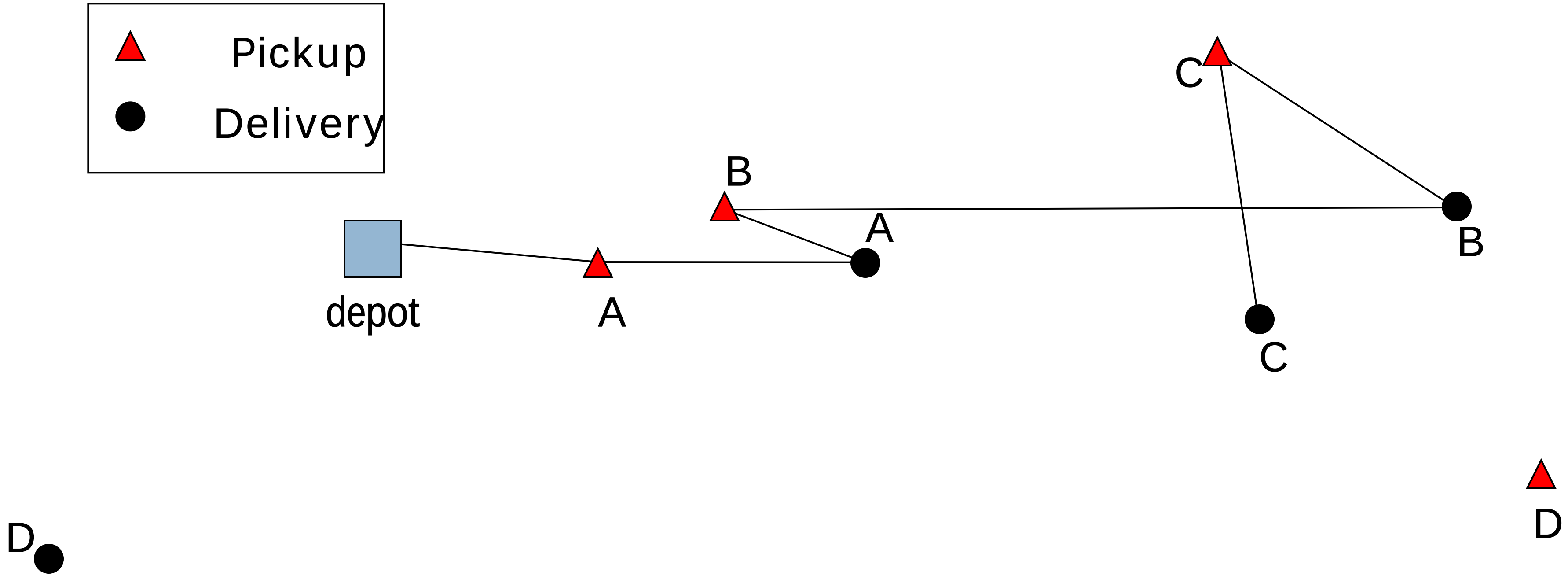
Intersperse Pickups and Deliveries?

- Can pickups and deliveries be interspersed?



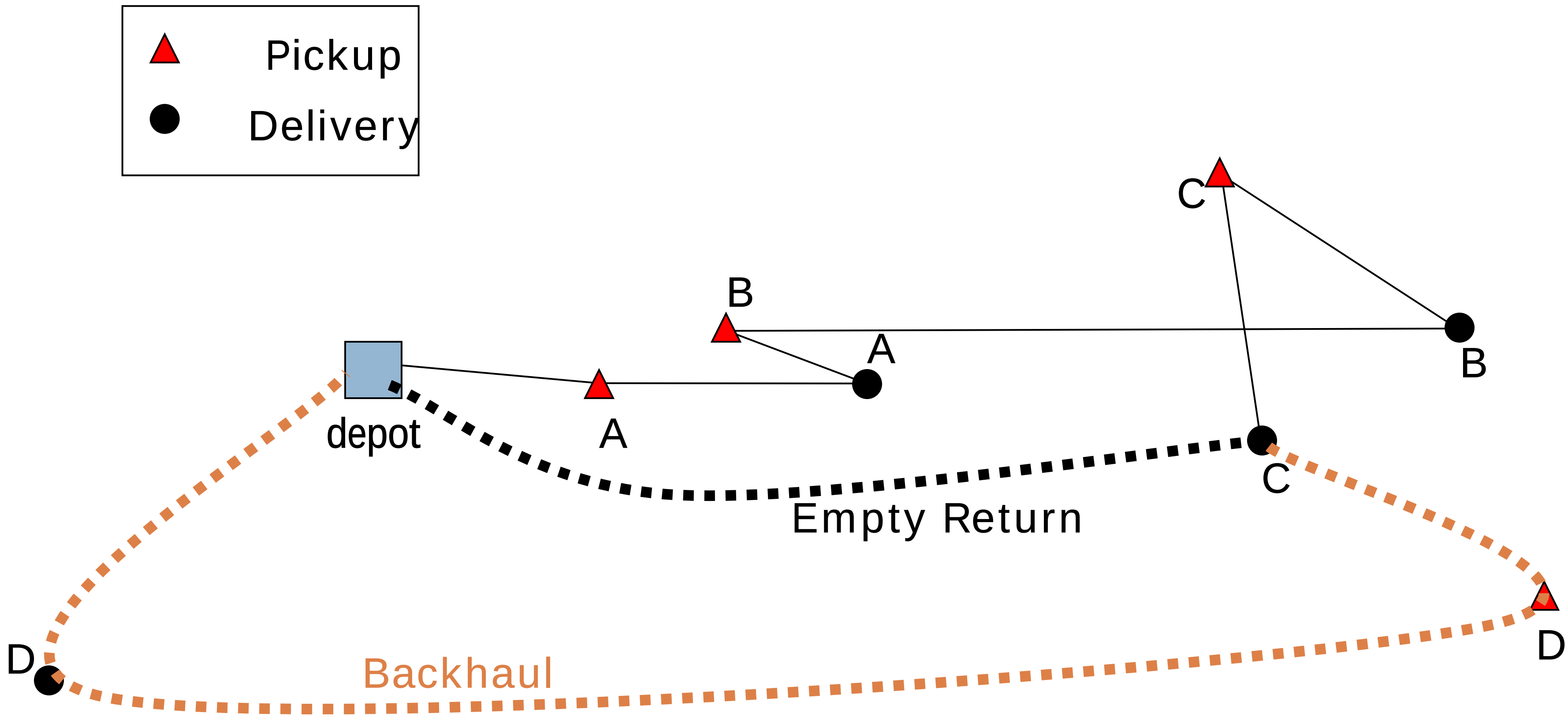
Backhauls

- If vehicle does not end at depot, should it return empty (deadhead) or find a backhaul?
  - How far out of the way should it look for a backhaul?



Backhauls

- Compare profit from deadheading and carrying backhaul.



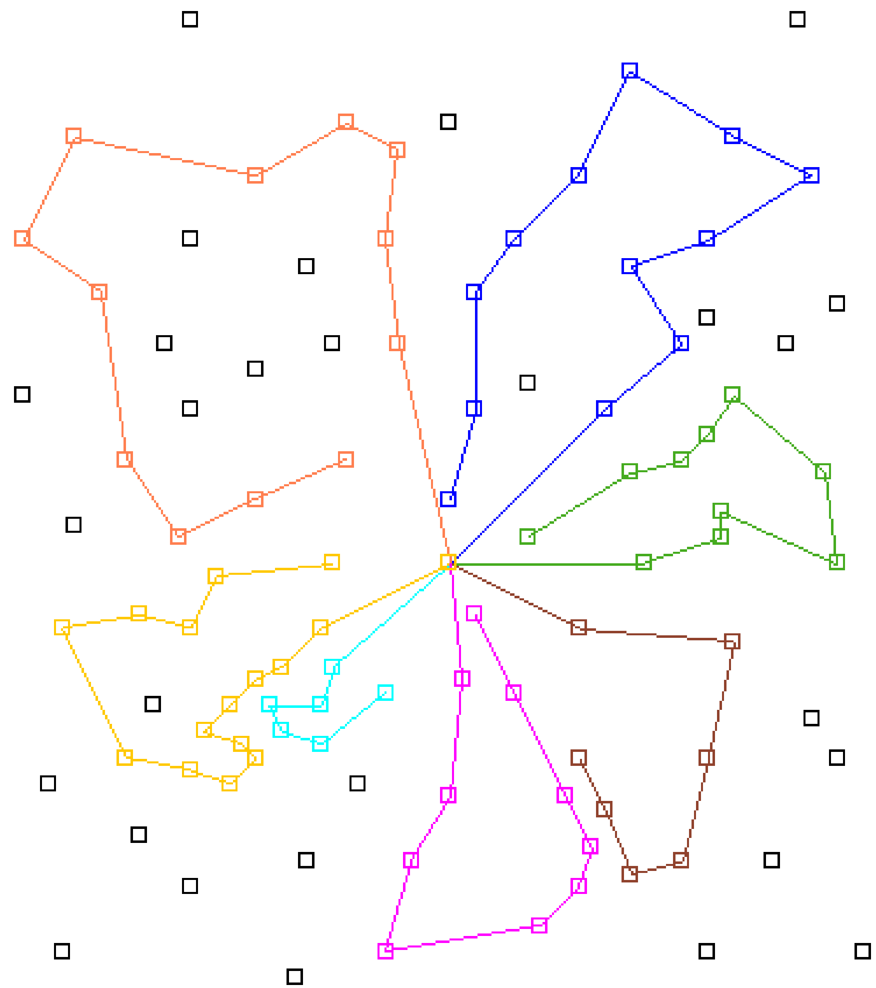
Fleet size and mix

- Heterogonous vehicles
  - Vehicles of different capacities, costs, speeds etc
- Fleet size and mix problem
  - Decide the correct number of each type of vehicle
  - Strategic decision
- Can be the most important part of optimization

Other variants

Profitable tour problem

- Not all visits need to be completed
- Known profit for each visit
- Choose a subset that gives maximum return (*profit from visits – routing cost*)



Other variants

Period Routing

- Routing with periodical deliveries
- Same routes every week / fortnight
- Deliver to different customers with different frequencies: patterns
- 3-part problem
  - Choose pattern for each cust
  - Choose qty for each delivery
  - Design route for each day

M	T	W	T	F
	?		?	
?			?	
?		?		?
?	?	?	?	?

VRP meets the real world

- *Rich VRPs*
- Attempt to model constraints common to many real-life enterprises
  - Multiple Time windows
  - Multiple Commodities
  - Heterogeneous vehicles
  - Compatibility constraints
    - Goods for customer *A* can't travel with goods from customer *B*
    - Goods for customer *A* can't travel on vehicle C

VRP meets the real world

- **Other real-world considerations**
- Fatigue rules and driver breaks
- Vehicle re-use (multiple trips per day)
- Ability to change vehicle characteristics (composition)
  - Add trailer, or move compartment divider
- Use of limited resources
  - e.g limited docks for loading, hence need to stagger dispatch times
- Variable loading / unloading times

VRP meets the real world

- **Yet more constraints**
- Only two types of product on each vehicle
- Consistent constraints
  - Customers visited in ‘patterns’ (Period Routing)
  - Same driver every day
  - Around the same time
- Meet ferry
- Blood transport (dynamic time window)
- Promiscuous driver constraint

VRP meets the real world

- New data sources

- Routing with time-of-day dependent travel times
  - Uses historical data to forecast travel time at different times of day

- Routing with dynamic travel times
  - Uses live traffic information feed to update expected travel time dynamically

VRP meets the real world

- Stochastic Routing
- What if things don't go according to plan?
- Sources of uncertainty
  - Uncertainty in existence (do I even need to visit)
  - Uncertainty in quantity (how much is actually required)
  - Uncertainty in travel times (traffic)
  - Uncertainty in duration (maintenance engineer)
- Optimal solution can be brittle
  - If something is not quite right, whole solution falls apart

VRP meets the real world

- E.g. garbage collection
- Wet rubbish is heavy. On rainy days, trucks may have more load than usual (uncertainty in quantity)
- Need stops near dump in case they have to double back
- = Recourse.