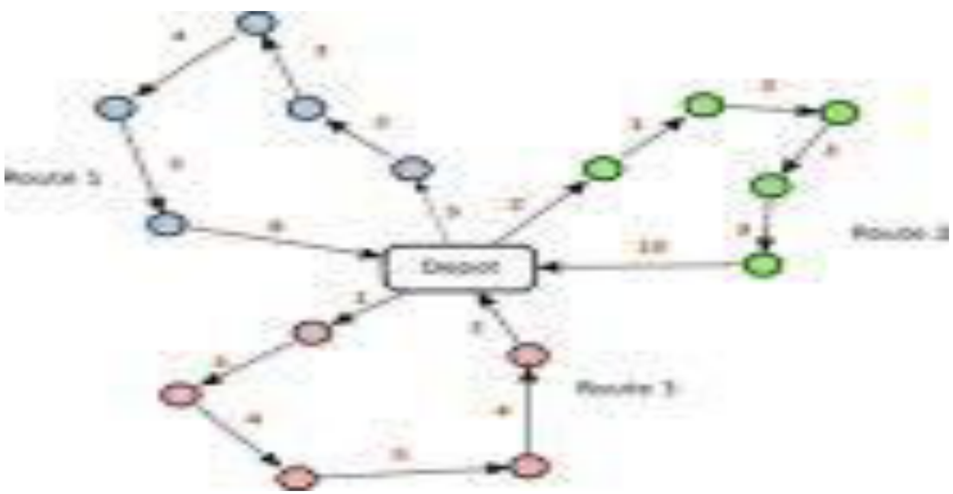
Computational Complexity CSC 5802

Professor: Tom Altman

Capacitated Problem



Agenda:

- Definition
- Example
- Solution Techniques
- Implementation

Capacitated VRP (CPRV)

CVRP is a Vehicle Routing Problem (VRP) in which a fixed fleet of delivery vehicles of uniform capacity must service known customer demands for a single commodity from a common depot at minimum transit cost.

That is, CVRP is like VRP with the additional constraint that every vehicle must have uniform capacity of a single commodity.

Objective:

The objective is to minimize the vehicle fleet and the sum of travel time, and the total demand of commodities for each route may not exceed the capacity of the vehicle which serves that route.

Feasibility:

A solution is feasible if the total quantity assigned to each route does not exceed the capacity of the vehicle which services the route.

Capacitated Vehicle Routing Problem

Given:

- Complete graph $\mathbf{G = (N, E)}$
- Set of nodes $\mathbf{N = \{0, 1, . . ., n\}}$
- Set of Edges $\mathbf{E = \{ (i, j) \mid i, j \in N ; i < j \}}$
- Cost of traveling from node **i** to node **j** $\mathbf{C_{ij}}$
- Demand per node $\mathbf{d_i}$ $\mathbf{(i \in N – \{0\})}$
- Vehicle capacity **C**
- Number of vehicles **K**

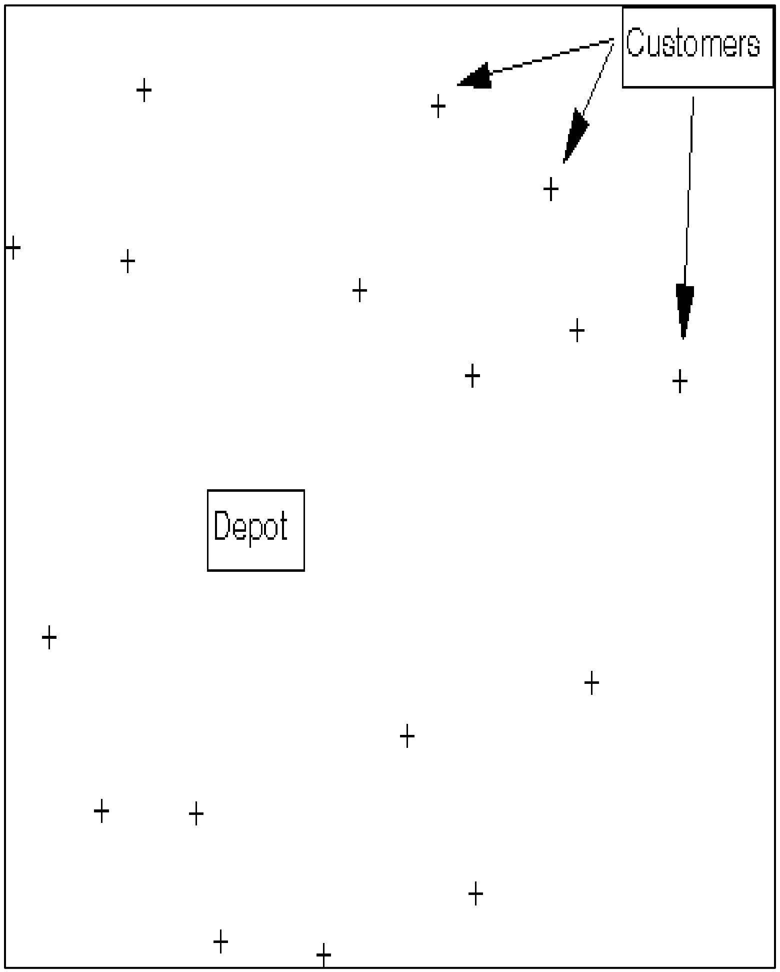
Capacitated Vehicle Routing Problem

Find:

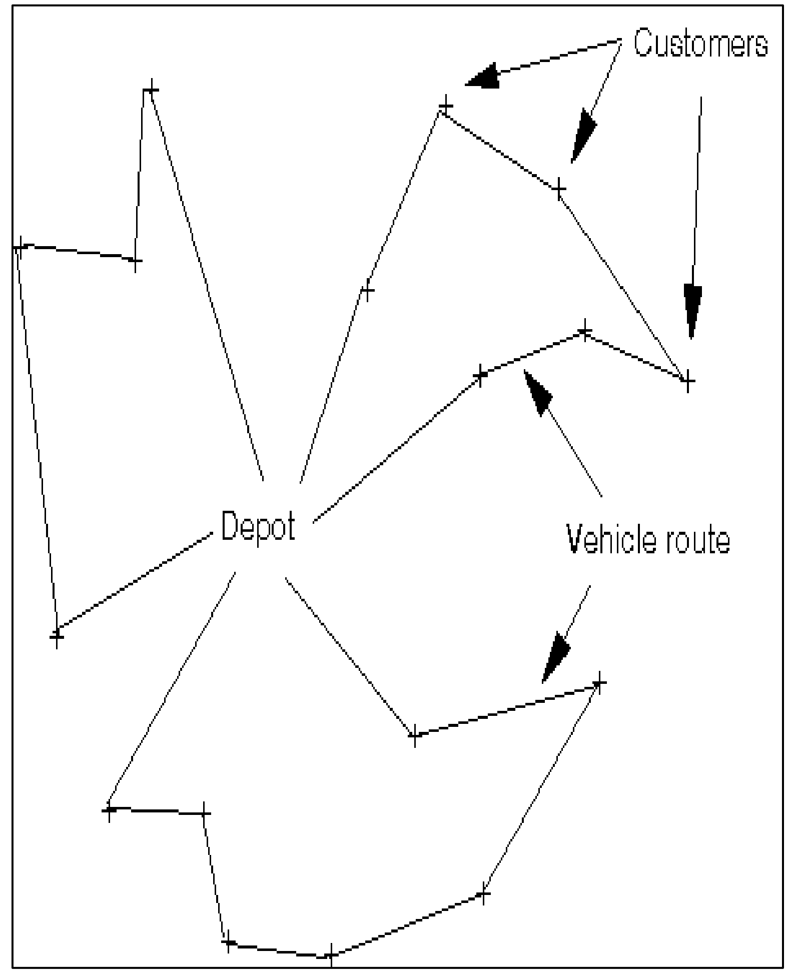
A set of at most **K** vehicle routes of total minimum cost such that:

- Every route starts and ends at the depot.
- Each customer is visited exactly once.
- The sum of the demands in each vehicle route does not exceed the vehicle's capacity.

The Typical input and output for a Vehicle Routing Problem



An input for a Vehicle Routing Problem



An output for the instance above

Formulation:

Let **Q** denote the capacity of a vehicle.

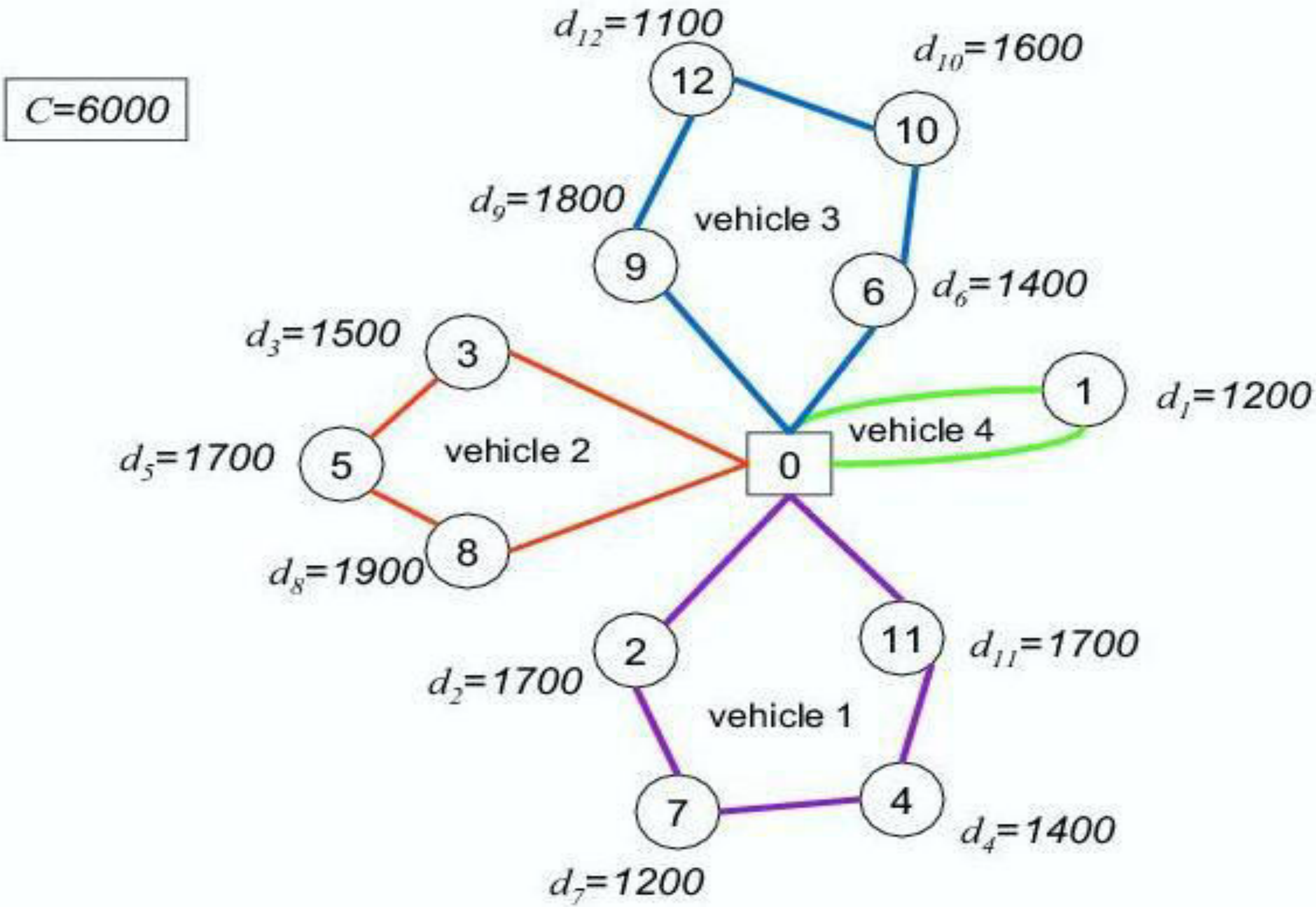
Mathematically, a solution for the CVRP is the same that VRP's one, but with the additional restriction that the total demand of all customers supplied on a route **R**i does not exceed the vehicle capacity **Q**:

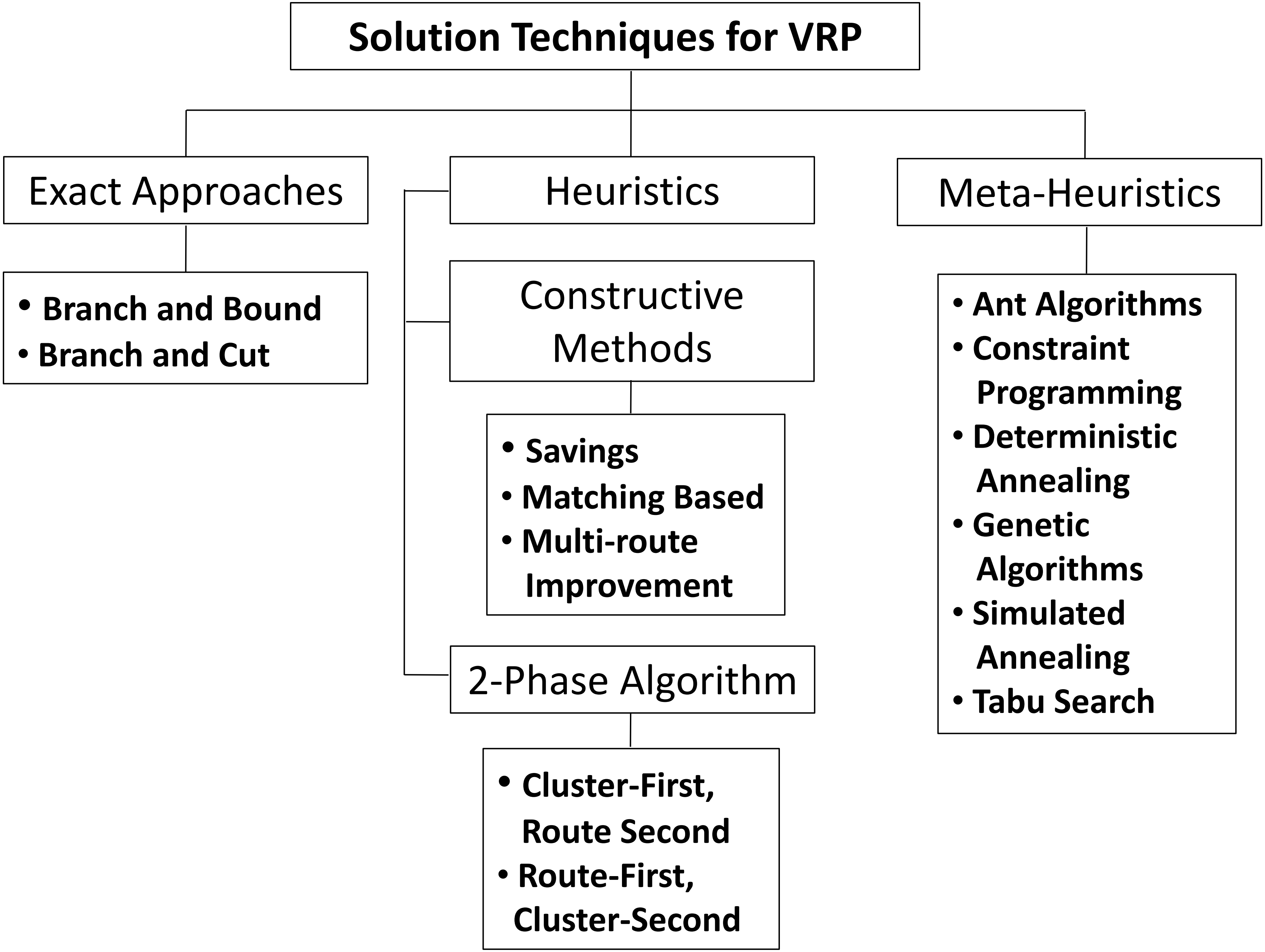
$$\sum_{i=1}^{m} di \leq Q$$

Example of CVRP instance CVRP

c_{ij}		j												
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
i	**0**		9	14	23	32	50	21	49	30	27	35	28	18
	1			21	22	36	52	24	51	36	37	41	30	20
	2				25	38	5	31	7	36	43	29	7	6
	3					42	12	35	17	44	31	31	11	6
	4						22	37	16	46	37	29	13	14
	5							41	23	10	39	9	17	16
	6								26	21	19	10	25	12
	7									30	28	16	27	12
	8										25	22	10	20
	9											20	16	8
	10												10	10
	11													10

Example of CVRP instance




Solution Techniques for VRP
Exact Approaches
Heuristics
Meta-Heuristics
• Branch and Bound
• Branch and Cut
Constructive Methods
• Savings
• Matching Based
• Multi-route Improvement
2-Phase Algorithm
• Cluster-First, Route Second
• Route-First, Cluster-Second
• Ant Algorithms
• Constraint Programming
• Deterministic Annealing
• Genetic Algorithms
• Simulated Annealing
• Tabu Search

Solution Techniques for VRP:

1 - Exact Approaches:

Propose to compute every possible solution until one of the bests is reached.

- Branch and bound (Fisher 1994)
- Branch and cut

2 - Heuristics:

Perform a relatively limited exploration of the search space and typically produce good quality solutions within modest computing times.

Constructive Methods:

Gradually build a feasible solution while keeping an eye on solution cost.

- Matching Based
- Multi-route Improvement Heuristics
  - Thompson and Psaraftis (1993)
  - Van Breedam (1994)
  - Kinderwater and Savelsbergh (1997)

2-Phase Algorithm:

The problem is decomposed into its two natural components:

1 - Clustering of vertices into feasible routes

2 - Actual route construction

with possible feedback loops between the two stages.

- Cluster-First, Route-Second Algorithms
  - Fisher and Jaikumar (1981)
  - The Petal Algorithm
  - The Sweep Algorithm
  - Taillard (1993)
- Route-First, Cluster-Second Algorithms

Meta-Heuristics:

The emphasis is on performing a deep exploration of the most promising regions of the solution space.
The quality of solutions produced by these methods is much higher than that obtained by classical heuristics.

- Ant Algorithms
- Constraint Programming
- Deterministic Annealing
- Genetic Algorithms
- Simulated Annealing
- Tabu Search
    - Granular Tabu
    - The adaptative memory procedure
    - Kelly and Xu (1999)

Matching Based Savings Algorithm:

This is an interesting modification to the standard Savings algorithm .
Wherein at each iteration the saving obtained by merging routes ***p*** and ***q*** is computed as:

$$S_{ij} = t(S_i) + t(S_j) - t(S_i \cup S_j)$$

Where $\mathbf{S_k}$ is the vertex set of route ***k***, and $\mathbf{t(S_k)}$ is the length of an optimal **TSP** solution on $\mathbf{S_k}$.

A matching problem over the sets $\mathbf{S_k}$ is solved using the $\mathbf{S_{ij}}$ values as matching costs, and the routes corresponding to optimal matchings are merged providing feasibility is maintained.

One possible variant of this basic algorithm consists on approximating the $\mathbf{t(S_k)}$ values instead of computing them exactly.

Multi-Route Improvement Algorithm:

Improvement algorithms attempt to upgrade any feasible solution by performing a sequence of edge or vertex exchanges within or between vehicle routes.

Multi-route improvement heuristics for the VRP operate on each vehicle route taken on several routes at a time.

Thompson and Psaraftis 1993:

Propose a method based on the concept of cyclic *k*-transfers that involves transferring simultaneously ***k*** demands from route $\mathbf{I}^{j}$ to route $\mathbf{I}^{\delta(j)}$ for each ***j*** and fixed integer ***k***.

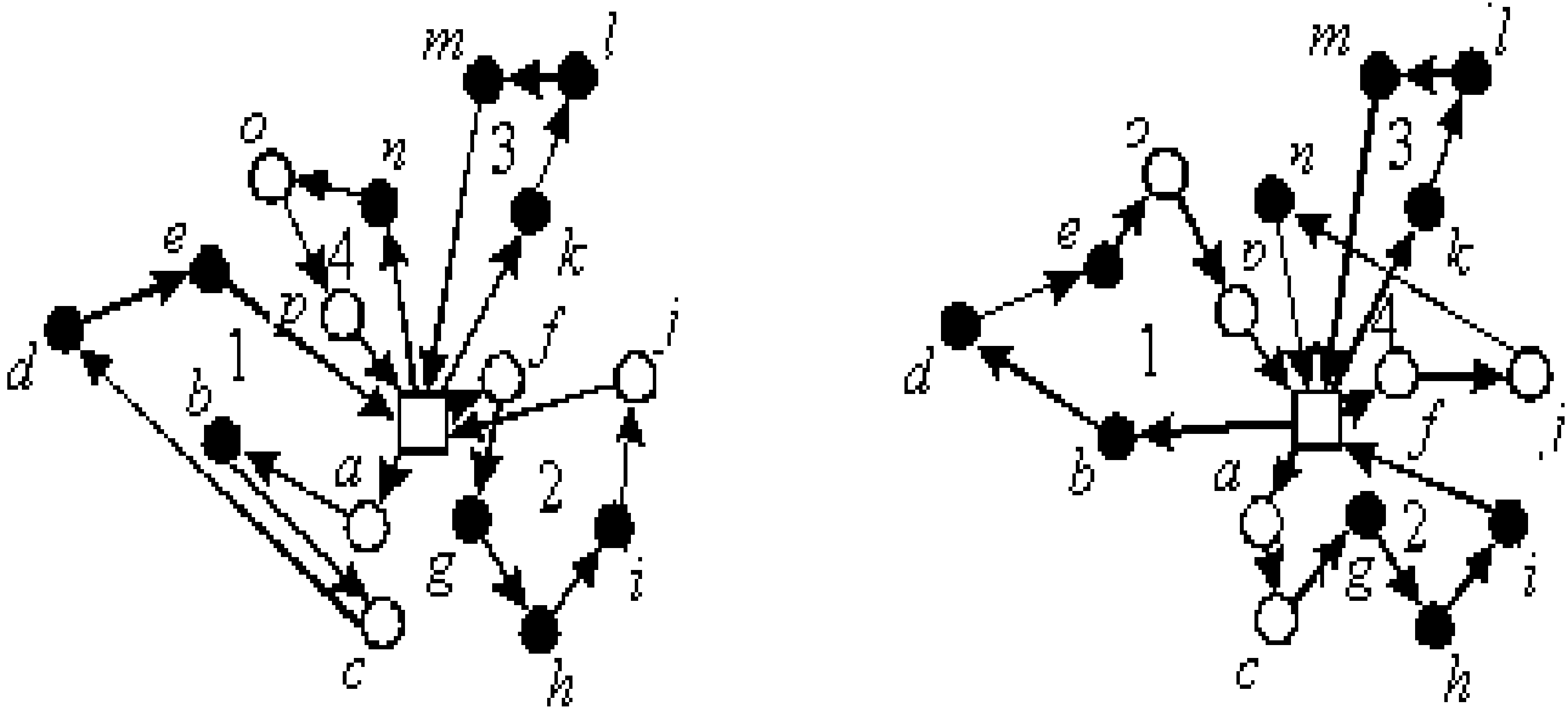
The set of routes $\{\mathbf{I}^{r}\}$, with **r = 1, . . ., m**, constitutes a feasible solution and δ is a cyclic permutation of a subset of **{1, . . ., m}**.

In particular, when $\boldsymbol{\delta}$ has fixed cardinality $\boldsymbol{C}$, we obtain a C-cyclic k-transfer.
By allowing $\boldsymbol{k}$ dummy demands on each route, demand transfers can be performed among permutations rather than cyclic permutations of routes.

Due to the complexity of the cyclic transfer neighborhood search, it is performed heuristically.

The 3-cyclic 2-transfer operator is illustrated in the figure below.

The cyclic transfer operator



The basic idea is to transfer simultaneously the customers denoted by white circles in cyclical manner between the routes.

More precisely here customers a and c in route 1, f and j in route 2 and o and p in route 4 are simultaneously transferred to routes 2, 4, and 1 respectively and route 3 remains untouched.

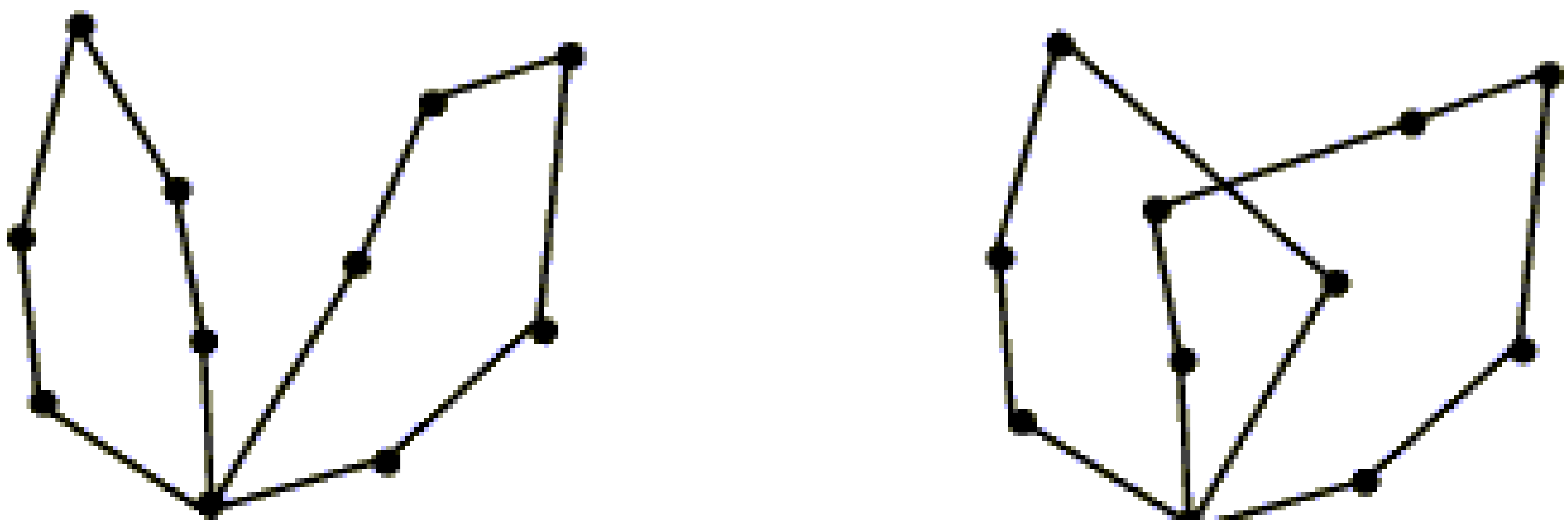
Van Breedam 1994:

Van Breedam classifies the improvement operations as **"string cross", "string exchange", "string relocation", and "string mix",** which can all be viewed as special cases of 2-cyclic exchanges, and provides a computational analysis on a restricted number of test problems.

In Van Breedam's analysis, there are considered four operations:

1 - String Cross (SC):

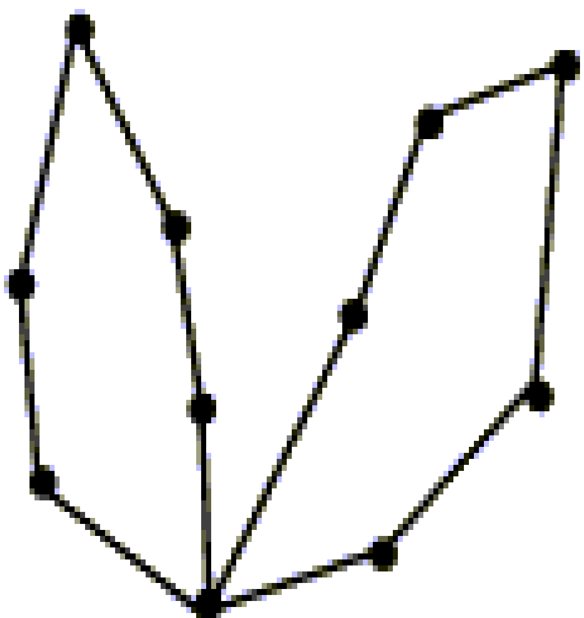
Two strings (or chains) of vertices are exchanged by crossing two edges of two different routes.



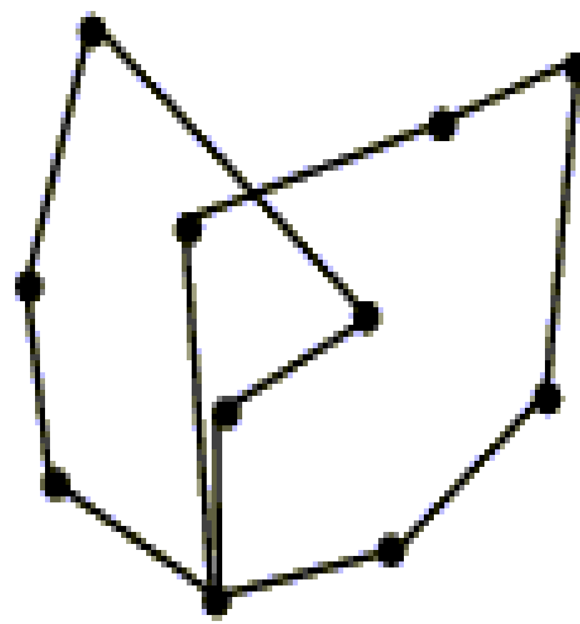
a) Before b) After

2 - String Exchange (SE):

Two strings of at most k vertices are exchanged between two routes.



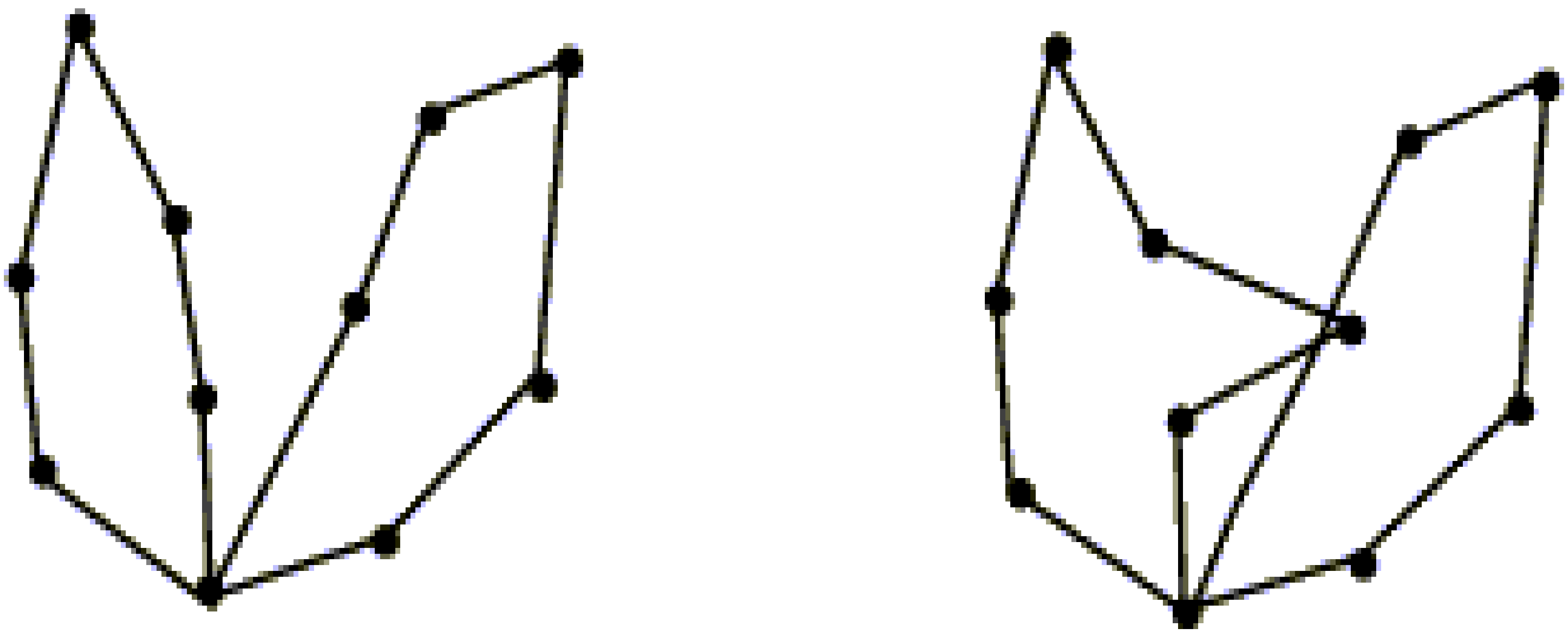
a) Before



b) After

3 - String Relocation (SR):

A string of at most k vertices is moved from one route to another, typically with k = 1 or 2.



a) Before b) After

4 - String Mix (SM):

The best move between SE and SR is selected.

To evaluate these moves, Van Breedam considers two local improvement strategies:

1 - First Improvement (FI):

Consists of implementing the first move that improves the objective function.

2 - Best Improvement (BI):

Evaluates all the possible moves and implements the best one.

Van Breedam then defines a set of parameters that can influence the behavior of the local improvement procedure:

- The initial solution **(poor, good)**
- The string length **(*k*)** for moves of type **SE, SR, SM (*k* = 1 or 2)**
- The selection strategy **(FI, BI)**
- The evaluation procedure for a string length ***k* > 1** (evaluate all possible string lengths between a pair of routes, increase ***k*** when a whole evaluation cycle has been completed without identifying an improvement move).

Kinderwater and Savelsbergh 1997:

Heuristic tours are not considered in isolation, so paths and customers are exchanged between different tours.

The operations that make these changes are:

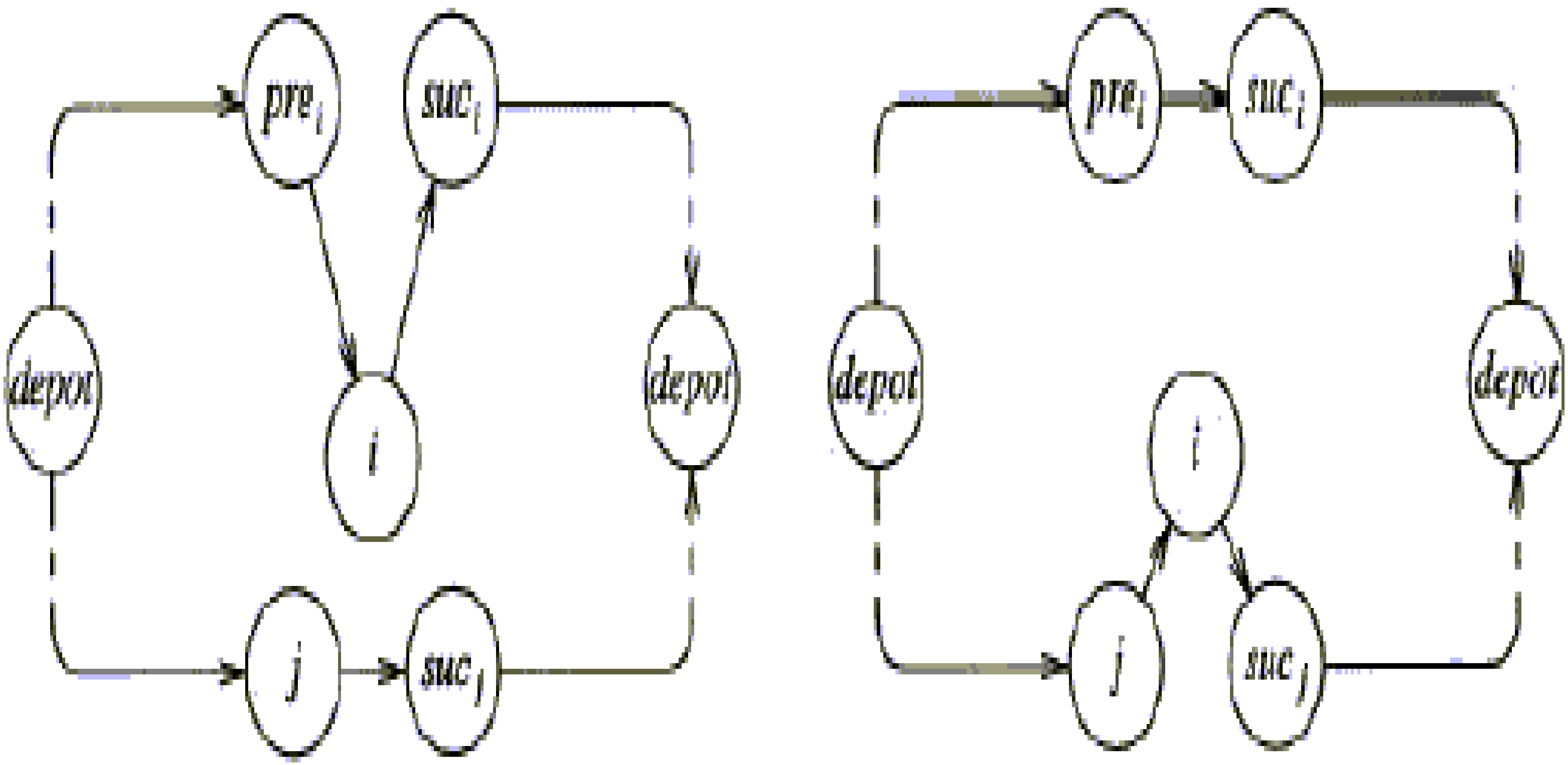
1 - Customer Relocation

2 – Crossover

3 - Customer Exchange

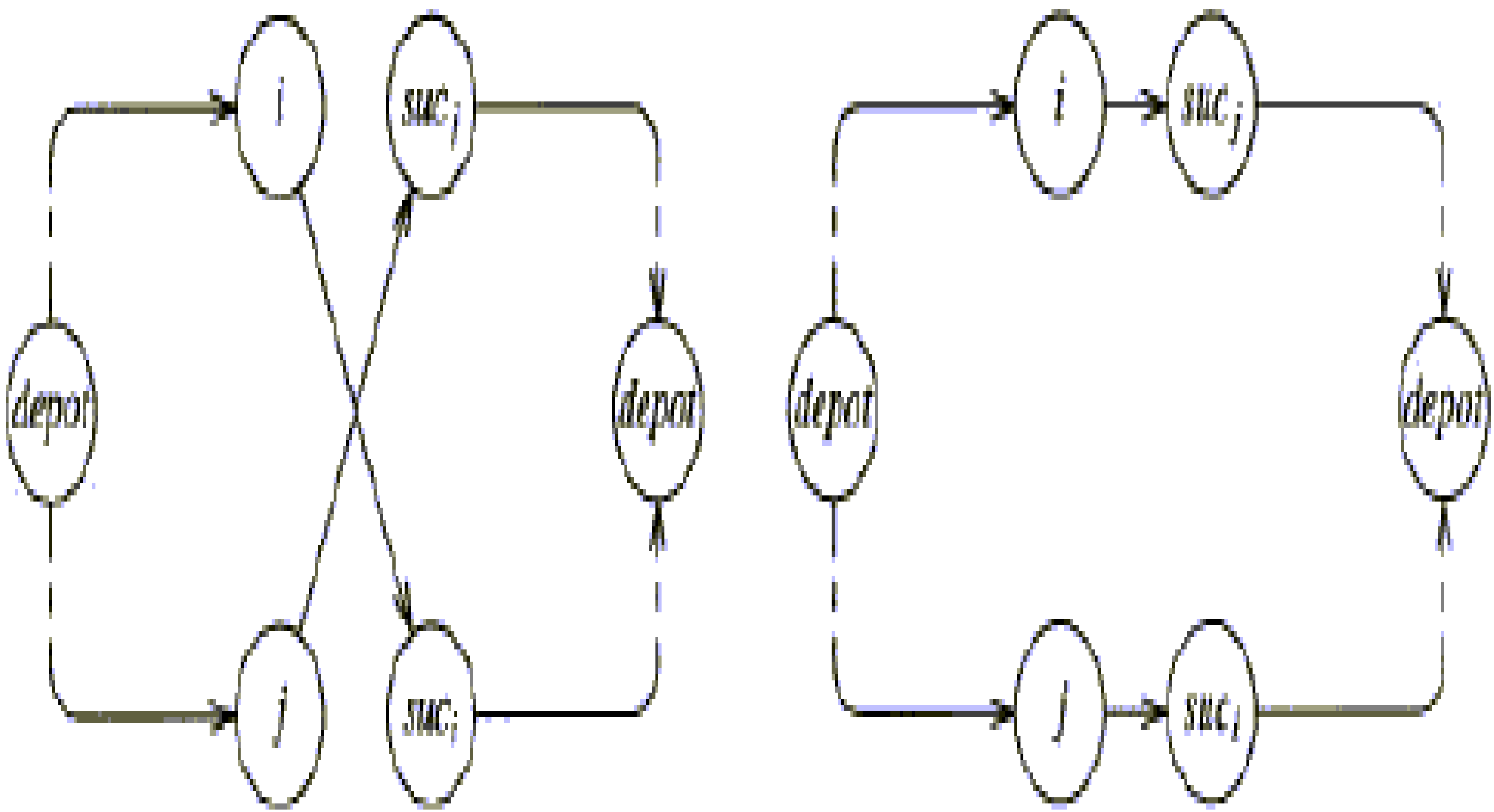
1 - Customer Relocation:

A customer located at one route is changed to another one:



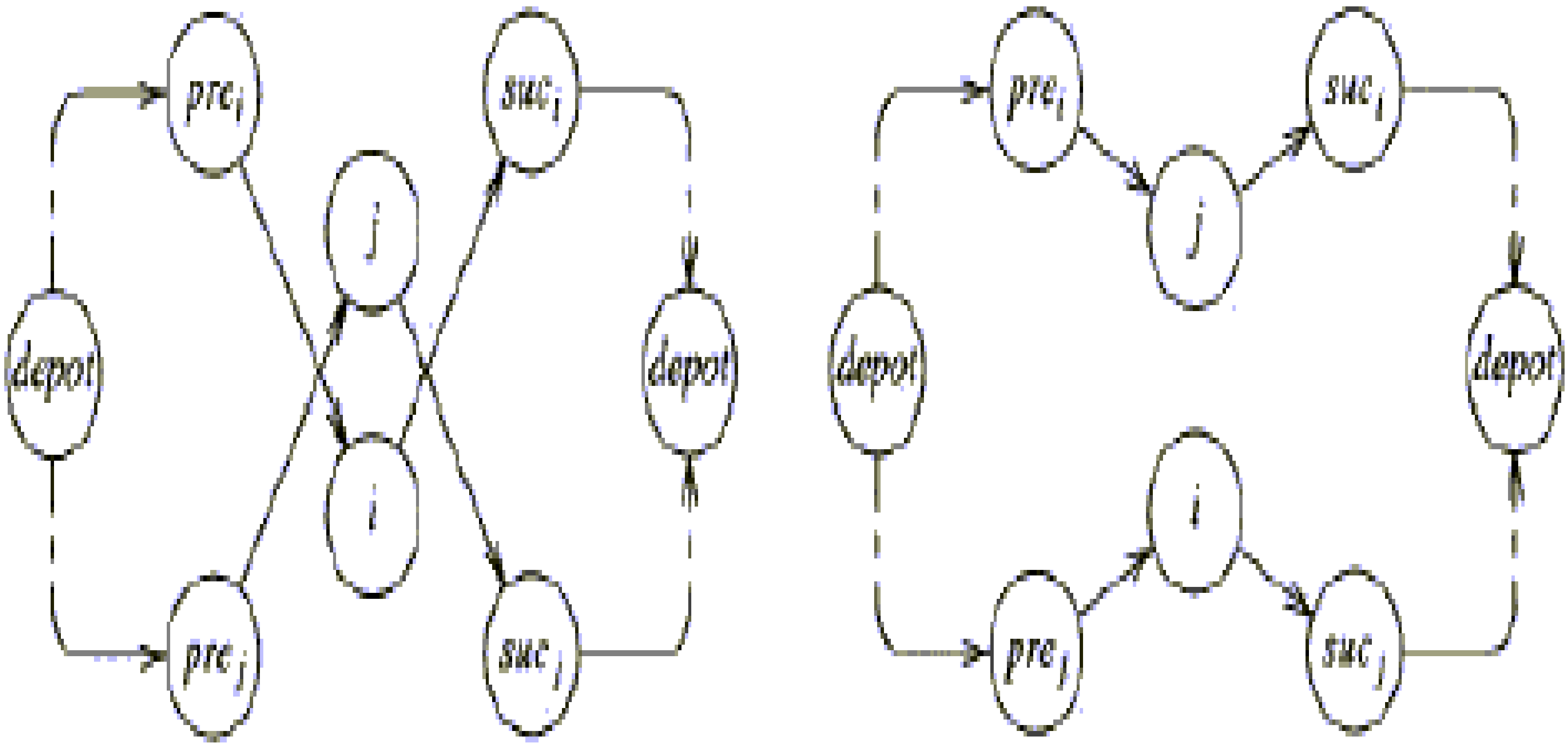
2 - Crossover:

Two routes are mixed at one point

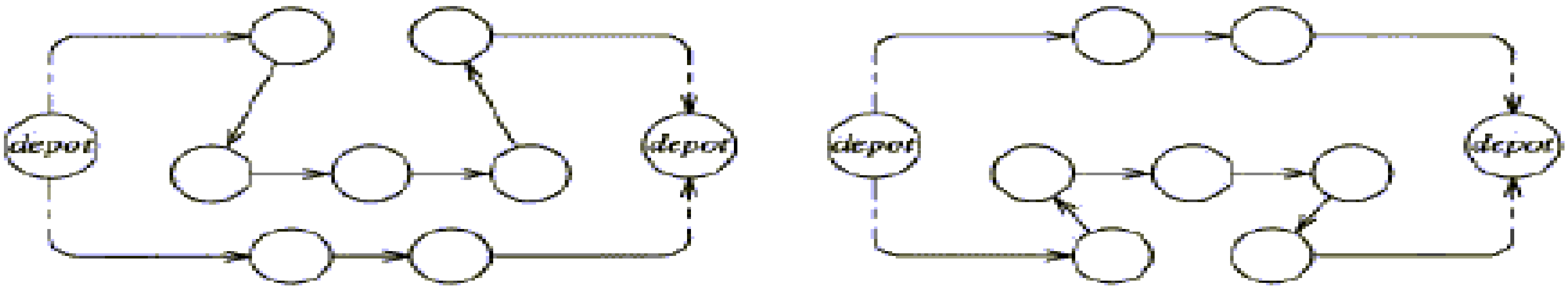


3 - Customer Exchange:

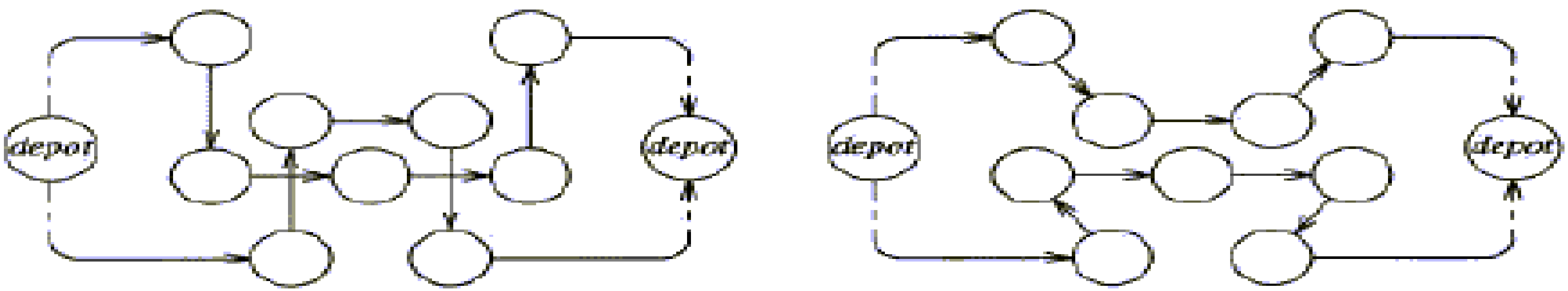
Two customers of two different routes are interchanged between the two routes



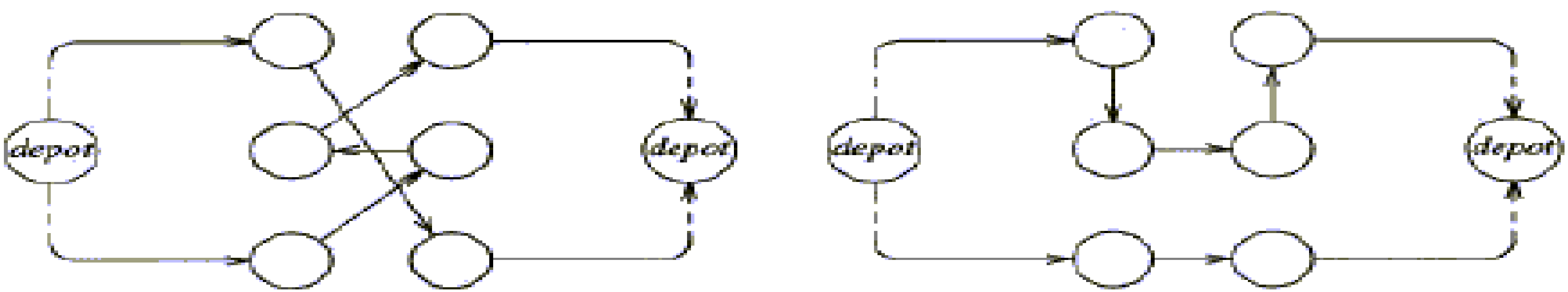
More Complex Examples:



(a) Relocation of a path.



(b) Exchange of two paths.



(c) A crossover plus 2-exchange.

Fisher and Jaikumar Algorithm 1981:

This is well known algorithm and it solves a Generalized Assignment Problem **(GAP)** to form the clusters.

The number of vehicles ***K*** is fixed.

The algorithm can be described as follows:

Step1. Seed Selection:

Choose seed points $\mathbf{j_k}$ in $\boldsymbol{V}$ to initialize each cluster $\boldsymbol{k}$.

Step2. Allocation of Customers to Seeds:

Compute the cost $\mathbf{d_{ik}}$ of allocating each customer $\boldsymbol{i}$ to each cluster $\boldsymbol{k}$ as .

$$\mathbf{d_{ijk} = \min\{c_{0i}+c_{ijk}+c_{jk0}, c_{0jk}+c_{jki}+c_{i0}\} - (c_{0jk}+c_{jk0})}$$

Step3. Generalized Assignment:

Solve a **GAP** with costs $\mathbf{d_{ij}}$, customer weights $\mathbf{q_i}$ and vehicle capacity $\boldsymbol{Q}$.

Step4. TSP Solution:

Solve a **TSP** for each cluster corresponding to the **GAP** solution.

Petal Algorithm:

It is a natural extension of the sweep algorithm It is used to generate several routes, called ***petals*** , and make a final selection by solving a set partitioning problem of the form:

min $\sum_{k \in s} d_k x_k$

subject to: $\sum_{k \in s} a_{ki} x_k$ **= 1, (i = 1, . . ., n),**

x_k = 1 or 0 , (k ∈ S)

Where:

S is the set of routes,
$\mathbf{x_k} = \mathbf{1}$ if and only if route ***k*** belongs to the solution,
$\mathbf{a_{ik}}$ is the binary parameter equal to **1** only if vertex ***i*** belongs to route ***k***,
$\mathbf{d_k}$ is the cost of petal ***k***.

If routes correspond to contiguous sectors of vertices, then this problem possesses the column circular property and be solved in polynomial time.

The Sweep Algorithm:

The sweep algorithm applies to planar instances of the VRP. It consists of two parts:

- **Split:**
  Feasible clusters are initialed formed rotating a ray centered at the depot.
- **TSP:**
  A vehicle routing is then obtained for each cluster by solving a **TSP**.

Some implementations include a post-optimization phase in which vertices are exchanged between adjacent clusters, and routes are reoptimized.

A simple implementation of this method is as follows, where we assume that each vertex ***i*** is represented by its polar coordinates (θ_i, p_i), where θ_i is the angle and p_i is the ray length.

Step1. Route Initialization:

Choose an unused vehicle ***k***.

Step2. Route Construction:

Starting from the unrouted vertex having the smallest angle, assign vertices to the vehicle ***k*** as long as its capacity or the maximal route length is not exceeded.

If unrouted vertices remain go to Step1.

Step3. Route Optimization:

Optimize each vehicle route separately by solving the corresponding **TSP**.

Taillard's Algorithm 1993:

Uses the **λ**-interchange generation mechanism where individual routes are reoptimized.

Decomposes the main problems into subproblems.

In planar problems, these subproblems are obtained by initially partitioning vertices into sectors centered at the depot, and into concentric regions within each sector.

Each subproblem can be solved independently, but periodical moves of vertices to adjacent sectors are necessary.

This make sense when the depot is centered and vertices are uniformly distributed in the plane.

This decomposition method is particularly well suited for parallel implementation as subproblems can then be distributed among the various processors.

Route-First Cluster-Second Method:

Route-first, cluster-second methods construct in a first phase a giant **TSP** tour, disregarding side constraints, and decompose this tour into feasible vehicle routes in a second phase.

This idea applies to problems with a free number of vehicles.

It was first put forward by Beasley who observed that the second phase problem is a standard shortest path problem on an acyclic graph and can thus be solved in $\mathbf{O(n^2)}$ time.

In the shortest path algorithm, the cost dij of traveling between nodes ***i*** and ***j*** is equal to $\mathbf{c_{0i} + c_{0j} + l_{ij}}$, where $\mathbf{l_{ij}}$ is the cost of traveling from ***i*** to ***j*** on the **TSP** tour.

Conclusion:

Near all of The techniques the are used for solving Vehicle Routing Problems are heuristics and metaheuristics because no exact algorithm can be guaranteed to find optimal tours within reasonable computing time when the number of cities is large.

This is due to the NP-Hardness of the problem.

The Implementation

References

1 - The VRP Web. http://neo.lcc.uma.es/radi-aeb/WebVRP/.

2 - The Capacitated Vehicle Routing Problem (CVRP). http://columbus.uniandes.edu.co:5050/dspace/bitstream/1992/772/5/JGA-CVRP-Example.pdf