DM204 – Spring 2011

Scheduling, Timetabling and Routing

Lecture 13
Vehicle Routing
Construction Heuristics

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Construction Heuristics
    - Construction Heuristics for CVRP
    - Construction Heuristics for VRPTW

Course Overview

- ✔ Scheduling
  - ✔ Classification
  - ✔ Complexity issues
  - ✔ Single Machine
  - ✔ Parallel Machine
  - ✔ Flow Shop and Job Shop
  - ✔ Resource Constrained Project Scheduling Model
- Timetabling
  - ✔ Sport Timetabling
  - ✔ Reservations and Education
  - ✔ University Timetabling
  - ✔ Crew Scheduling
  - ✔ Public Transports
- Vechicle Routing
  - ✔ MIP Approaches
  - Construction Heuristics
  - Local Search Algorithms

Outline

1. Construction Heuristics
    Construction Heuristics for CVRP
    Construction Heuristics for VRPTW

Outline

1. Construction Heuristics
    - Construction Heuristics for CVRP
    - Construction Heuristics for VRPTW

Construction Heuristics for CVRP

- TSP based heuristics
- Saving heuristics (Clarke and Wright)
- Insertion heuristics
- Cluster-first route-second
  - Sweep algorithm
  - Generalized assignment
  - Location based heuristic
  - Petal algorithm
- Route-first cluster-second

Cluster-first route-second seems to perform better than route-first
(Note: distinction construction heuristic / iterative improvement
is often blurred)

Construction heuristics for TSP

They can be used for route-first cluster-second or for growing multiple tours subject to capacity constraints.

- Heuristics that Grow Fragments
  - Nearest neighborhood heuristics
  - Double-Ended Nearest Neighbor heuristic
  - Multiple Fragment heuristic (aka, greedy heuristic)
- Heuristics that Grow Tours
  - Nearest Addition
  - Farthest Addition
  - Random Addition
  - Nearest Insertion
  - Farthest Insertion
  - Random Insertion
  - Clarke-Wright saving heuristic
- Heuristics based on Trees
  - Minimum spanning tree heuristic
  - Christofides' heuristics

(But recall! Concorde: http://www.tsp.gatech.edu/)

[Bentley, 1992]

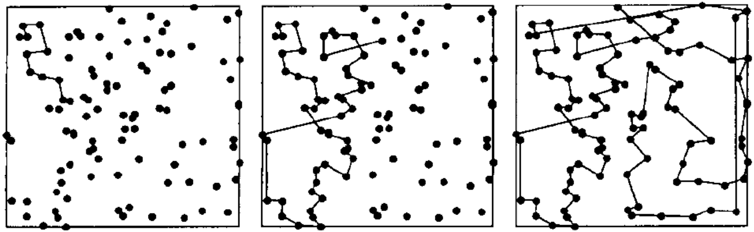


Figure 1. The Nearest Neighbor heuristic.

NN (Flood, 1956)

1. Randomly select a starting node
2. Add to the last node the closest node until no more node is available
3. Connect the last node with the first node

Running time $O(N^2)$

[Bentley, 1992]

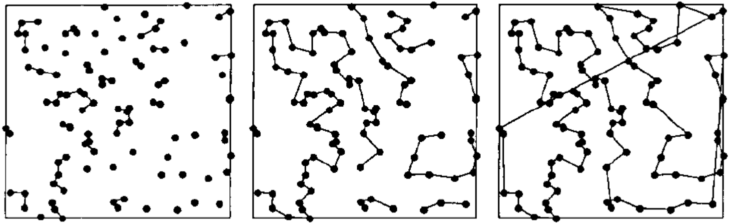


Figure 5. The Multiple Fragment heuristic.

Add the cheapest edge provided it does not create a cycle.

[Bentley, 1992]

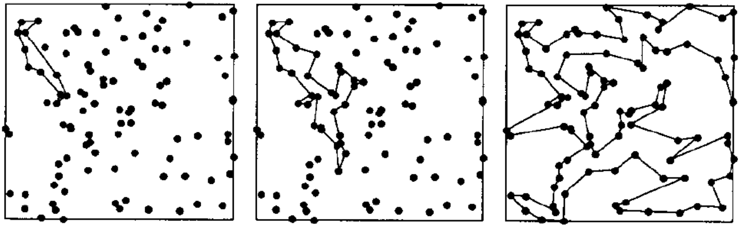


Figure 8. The Nearest Addition heuristic.

NA

1. Select a node and its closest node and build a tour of two nodes
2. Insert in the tour the closest node v until no more node is available

Running time $O(N^3)$

[Bentley, 1992]

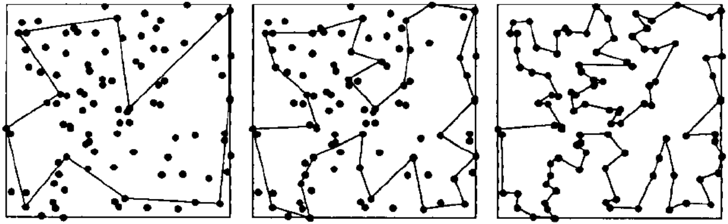


Figure 11. The Farthest Addition heuristic.

FA

1. Select a node and its farthest and build a tour of two nodes
2. Insert in the tour the farthest node v until no more node is available

FA is more effective than NA because the first few farthest points sketch a broad outline of the tour that is refined after.

Running time $O(N^3)$

[Bentley, 1992]

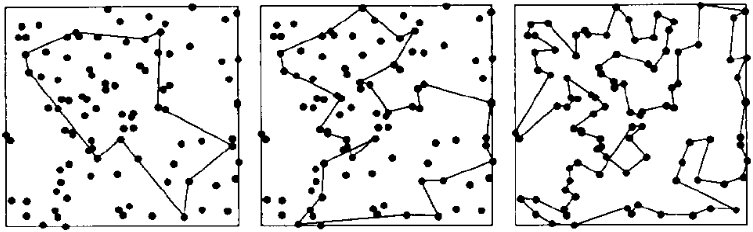


Figure 14. The Random Addition heuristic.

[Bentley, 1992]

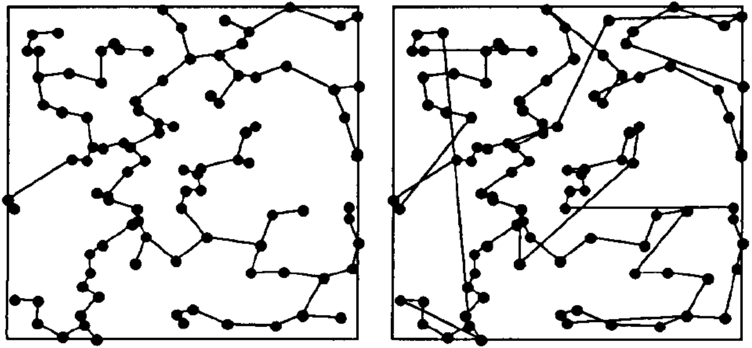


Figure 18. The Minimum Spanning Tree heuristic.

1. Find a minimum spanning tree $O(N^2)$
2. Append the nodes in the tour in a depth-first, inorder traversal

Running time $O(N^2)$ $A = MST(I)/OPT(I) \leq 2$

[Bentley, 1992]

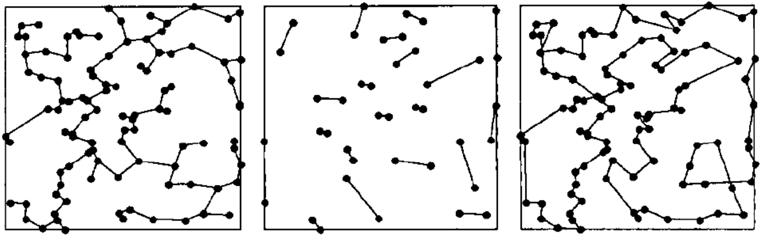
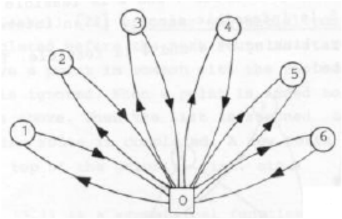


Figure 19. Christofides' heuristic.

1. Find the minimum spanning tree T. $O(N^2)$
2. Find nodes in T with odd degree and find the cheapest perfect matching M in the complete graph consisting of these nodes only. Let G be the multigraph of all nodes and edges in T and M. $O(N^3)$
3. Find an Eulerian walk (each node appears at least once and each edge exactly once) on G and an embedded tour. $O(N)$

Running time $O(N^3)$ $A = CH(I)/OPT(I) \leq 3/2$

Construction Heuristics Specific for VRP



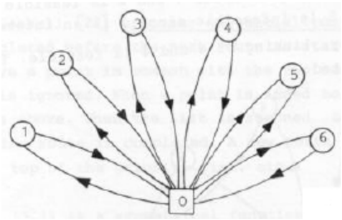
Clarke-Wright Saving Heuristic (1964)

1. Start with an initial allocation of one vehicle to each customer (0 is the depot for VRP or any chosen city for TSP)

Sequential:

2. consider in turn route $(0, i, \ldots, j, 0)$ determine s_{ki} and s_{jl}
3. merge with $(k, 0)$ or $(0, l)$

Construction Heuristics Specific for VRP

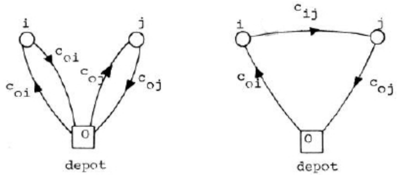


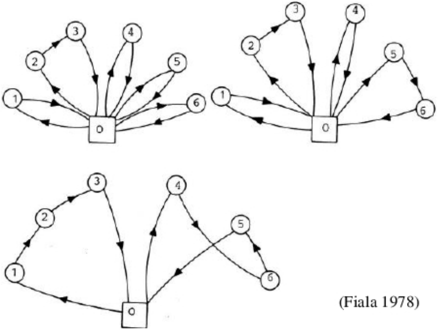
Clarke-Wright Saving Heuristic (1964)

1. Start with an initial allocation of one vehicle to each customer (0 is the depot for VRP or any chosen city for TSP)

Parallel:

2. Calculate saving $s_{ij} = c_{0i} + c_{0j} - c_{ij}$ and order the saving in non-increasing order
3. scan s_{ij}
   merge routes if i) i and j are not in the same tour ii) neither i and j are interior to an existing route iii) vehicle and time capacity are not exceeded





(Fiala 1978)

Matching Based Saving Heuristic

1. Start with an initial allocation of one vehicle to each customer (0 is the depot for VRP or any chosen city for TSP)
2. Compute $s_{pq} = t(S_p) + t(S_q) - t(S_p \cup S_q)$ where $t(\cdot)$ is the TSP solution
3. Solve a max weighted matching on the S_k with weights s_{pq} on edges. A connection between a route p and q exists only if the merging is feasible.

Insertion Heuristic

$$\alpha(i,k,j) = c_{ik} + c_{kj} - \lambda c_{ij}$$

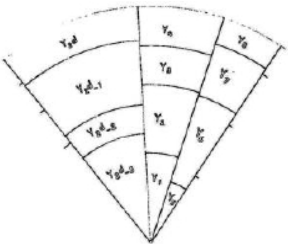
$$\beta(i,k,j) = \mu c_{0k} - \alpha(i,k,j)$$

1. construct emerging route $(0, k, 0)$
2. compute for all k unrouted the feasible insertion cost:

$$\alpha^*(i_k, k, j_k) = \min_p \{\alpha(i_p, k, i_{p+1})\}$$

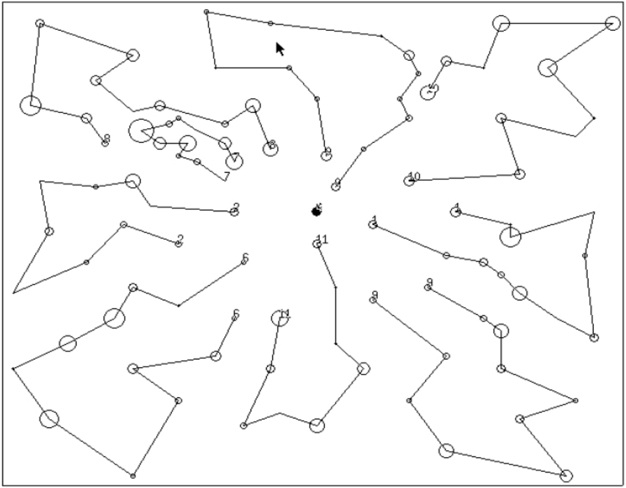
if no feasible insertion go to 1 otherwise choose k^* such that

$$\beta^*(i_k^*, k^*, j_k^*) = \max_k \{\beta(i_k, k, j_k)\}$$



Cluster-first route-second: Sweep algorithm [Wren & Holliday (1971)]

1. Choose i^* and set $\theta(i^*) = 0$ for the rotating ray
2. Compute and rank the polar coordinates (θ, ρ) of each point
3. Assign customers to vehicles until capacity not exceeded. If needed start a new route. Repeat until all customers scheduled.



Cluster-first route-second: Generalized-assignment-based algorithm [Fisher & Jaikumur (1981)]

1. Choose a j_k at random for each route k
2. For each point compute

$$d_{ik} = \min\{c_{0,i} + c_{i,j_k} + c_{j_k,0}, c_{0j_k} + c_{j_k,i} + c_{i,0}\} - (c_{0,j_k} + c_{j_k,0})$$

3. Solve GAP with d_{ik}, Q and q_i

Cluster-first route-second: Location based heuristic [Bramel & Simchi-Levi (1995)]

1. Determine seeds by solving a capacitated location problem (k-median)
2. Assign customers to closest seed

(better performance than insertion and saving heuristics)

Cluster-first route-second: Petal Algorithm

1. Construct a subset of feasible routes

2. Solve a set partitioning problem

Route-first cluster-second [Beasley, 1983]

1. Construct a TSP tour over all customers

2. Choose an arbitrary orientation of the TSP;
   partition the tour according to capacity constraint;
   repeat for several orientations and select the best
   Alternatively, solve a shortest path in an acyclic digraph with costs on arcs: $d_{ij} = c_{0i} + c_{0j} + l_{ij}$ where l_{ij} is the cost of traveling from i to j in the TSP tour.

(not very competitive)

Exercise

Which heuristics can be used to minimize K
and which ones need to have K fixed a priori?

Construction Heuristics for VRPTW

Extensions of those for CVRP [Solomon (1987)]

- Saving heuristics (Clarke and Wright)
- Time-oriented nearest neighbors
- Insertion heuristics
- Time-oriented sweep heuristic

Time-Oriented Nearest-Neighbor

- Add the unrouted node “closest” to the depot or the last node added without violating feasibility
- Metric for “closest”:

$$c_{ij} = \delta_1 d_{ij} + \delta_2 T_{ij} + \delta_3 v_{ij}$$

d_{ij} geographical distance

T_{ij} time distance

v_{ij} urgency to serve j

Insertion Heuristics

Step 1: Compute for each unrouted costumer u the *best feasible position* in the route:

$$c_1(i(u), u, j(u)) = \min_{p=1,\ldots,m} \{c_1(i_{p-1}, u, i_p)\}$$

(c_1 is a composition of increased time and increase route length due to the insertion of u)
(see next slide for efficiency issues)

Step 2: Compute for each unrouted customer u which can be feasibly inserted:

$$c_2(i(u^*), u^*, j(u^*)) = \max_u \{\lambda d_{0u} - c_1(i(u), u, j(u))\}$$

(max the benefit of servicing a node on a partial route rather than on a direct route)

Step 3: Insert the customer u^* from Step 2

Efficiency issues

Push forward

- Let's assume waiting is allowed and s_i indicates service times
- $[e_i, l_i]$ time window, w_i waiting time
- $b_i = \max\{e_i, b_j + s_j + t_{ji}\}$ begin of service
- insertion of u: $(i_0, i_1, \ldots, i_p, \mathbf{u}, i_{p+1}, \ldots, i_m)$
- $PF_{i_{p+1}} = b^{new}_{i_{p+1}} - b_{i_{p+1}} \geq 0$ push forward
- $PF_{i_{r+1}} = \max\{0, PF_{i_r} - w_{i_{r+1}}\}, \qquad p \leq r \leq m-1$

Theorem

The insertion is feasible if and only if:

$$b_u \leq l_u \quad \text{and} \quad PF_{i_r} + b_{i_r} \leq l_{i_r} \quad \forall p < r \leq m$$

Check vertices k, $u \leq k \leq m$ sequentially.

- if $b_k + PF_k > l_k$ then stop: the insertion is infeasible
- if $PF_k = 0$ then stop: the insertion is feasible