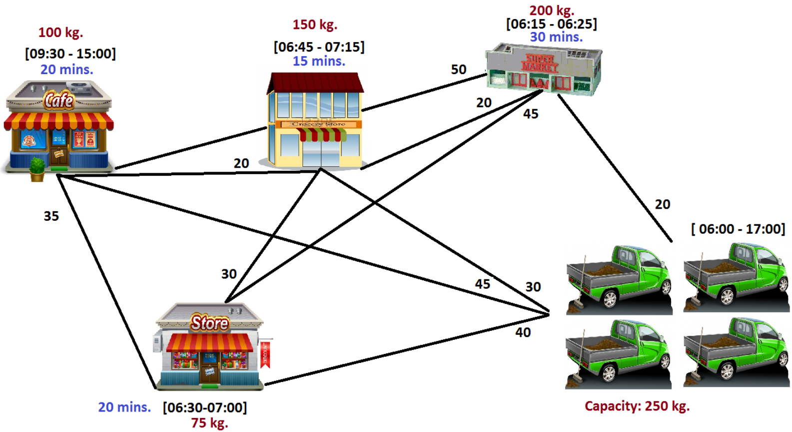
A matching-based heuristic approach to Vehicle Routing Problem with Time Windows

Murat Fırat

Post-doc researcher under DAIPEX Project
Mathematics & Computer Science, TU/e

February 20, 2014

A milk Company



Problem Description

An instance of the VRPTW includes:

Problem Description

An instance of the VRPTW includes:

- Depot 0 with time window $[e_0, l_0]$, planning horizon.

Problem Description

An instance of the VRPTW includes:

- Depot 0 with time window $[e_0, l_0]$, planning horizon.
- A set of homogenous vehicles (V), each with capacity Q.

Problem Description

An instance of the VRPTW includes:

- Depot 0 with time window $[e_0, l_0]$, planning horizon.
- A set of homogenous vehicles (V), each with capacity Q.
- A set of customers (N), every $i \in N$ having:

Problem Description

An instance of the VRPTW includes:

- Depot 0 with time window $[e_0, l_0]$, planning horizon.
- A set of homogenous vehicles (V), each with capacity Q.
- A set of customers (N), every $i \in N$ having:
  - time window $[e_i, l_i]$, demand q_i, and service time s_i.

Problem Description

An instance of the VRPTW includes:

- Depot 0 with time window $[e_0, l_0]$, planning horizon.
- A set of homogenous vehicles (V), each with capacity Q.
- A set of customers (N), every $i \in N$ having:
  - time window $[e_i, l_i]$, demand q_i, and service time s_i.
- A directed complete network $G(N \cup 0, A)$ s.t.

Problem Description

An instance of the VRPTW includes:

- Depot 0 with time window $[e_0, l_0]$, planning horizon.
- A set of homogenous vehicles (V), each with capacity Q.
- A set of customers (N), every $i \in N$ having:
  - time window $[e_i, l_i]$, demand q_i, and service time s_i.
- A directed complete network $G(N \cup 0, A)$ s.t.
  - every arc $(i, j) \in A$ has cost c_{ij} and travel time t_{ij}.

Problem Description

In a feasible solution to the VRPTW:

Problem Description

In a feasible solution to the VRPTW:

- Every vehicle $v \in V$ has a route $R_v = (0, \ldots, 0)$.

Problem Description

In a feasible solution to the VRPTW:

- Every vehicle $v \in V$ has a route $R_v = (0, \ldots, 0)$.
- $|\{v \in V | i \in R_v\}| = 1, \forall i \in N$.

Problem Description

In a feasible solution to the VRPTW:

- Every vehicle $v \in V$ has a route $R_v = (0, \ldots, 0)$.
- $|\{v \in V | i \in R_v\}| = 1, \forall i \in N$.
- Let $A_i(D_i)$ denote arrival (departure) time to (from) i, then

Problem Description

In a feasible solution to the VRPTW:

- Every vehicle $v \in V$ has a route $R_v = (0, \ldots, 0)$.
- $|\{v \in V | i \in R_v\}| = 1, \forall i \in N$.
- Let $A_i(D_i)$ denote arrival (departure) time to (from) i, then
  - $A_i \leq l_i$ and $\max\{A_i, e_i\} + s_i = D_i$.

Problem Description

In a feasible solution to the VRPTW:

- Every vehicle $v \in V$ has a route $R_v = (0, \ldots, 0)$.
- $|\{v \in V | i \in R_v\}| = 1, \forall i \in N$.
- Let $A_i(D_i)$ denote arrival (departure) time to (from) i, then
  - $A_i \leq l_i$ and $\max\{A_i, e_i\} + s_i = D_i$.
- Vehicle capacities are respected, $\sum_{i \in R_v} q_i \leq Q, \forall v \in V$.

Problem Description

In a feasible solution to the VRPTW:

- Every vehicle $v \in V$ has a route $R_v = (0, \dots, 0)$.
- $|\{v \in V | i \in R_v\}| = 1, \forall i \in N$.
- Let $A_i(D_i)$ denote arrival (departure) time to (from) i, then
  - $A_i \leq l_i$ and $\max\{A_i, e_i\} + s_i = D_i$.
- Vehicle capacities are respected, $\sum_{i \in R_v} q_i \leq Q, \forall v \in V$.
- Objectives:

Problem Description

In a feasible solution to the VRPTW:

- Every vehicle $v \in V$ has a route $R_v = (0, \dots, 0)$.
- $|\{v \in V | i \in R_v\}| = 1, \forall i \in N$.
- Let $A_i(D_i)$ denote arrival (departure) time to (from) i, then
  - $A_i \leq l_i$ and $\max\{A_i, e_i\} + s_i = D_i$.
- Vehicle capacities are respected, $\sum_{i \in R_v} q_i \leq Q, \forall v \in V$.
- Objectives:
  - Minimize (1) number of vehicles used, (2) total travel distance.

Our matching-based heuristic approach

Our matching-based heuristic approach

Our matching-based heuristic approach

- Our approach is a greedy, and constructive heuristic.

Our matching-based heuristic approach

- Our approach is a greedy, and constructive heuristic.
- A subset of customers are iteratively inserted into partial solution.

Our matching-based heuristic approach

- Our approach is a greedy, and constructive heuristic.
- A subset of customers are iteratively inserted into partial solution.
- We find a many-to-one type matching between (subsets) of vehicles and customers.

Our matching-based heuristic approach

- Our approach is a greedy, and constructive heuristic.
- A subset of customers are iteratively inserted into partial solution.
- We find a many-to-one type matching between (subsets) of vehicles and customers.
- The matching must respect capacities, and timing contraints.
- In an iteration,

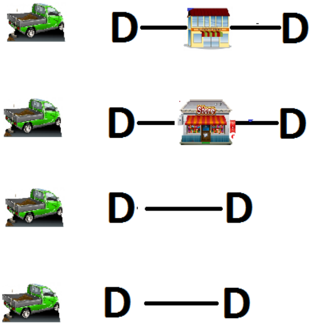
Our matching-based heuristic approach

- Our approach is a greedy, and constructive heuristic.
- A subset of customers are iteratively inserted into partial solution.
- We find a many-to-one type matching between (subsets) of vehicles and customers.
- The matching must respect capacities, and timing contraints.
- In an iteration,
  - new customers are inserted into routes.

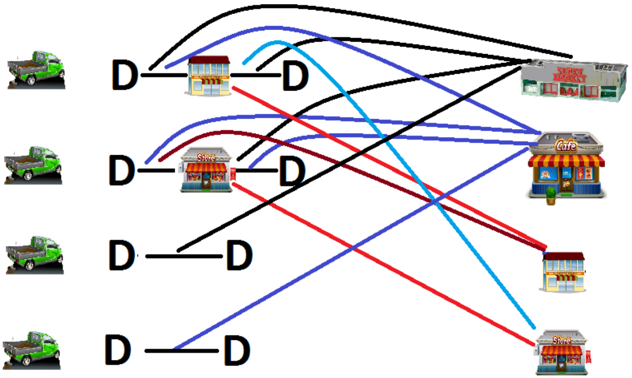
Our matching-based heuristic approach

- Our approach is a greedy, and constructive heuristic.
- A subset of customers are iteratively inserted into partial solution.
- We find a many-to-one type matching between (subsets) of vehicles and customers.
- The matching must respect capacities, and timing contraints.
- In an iteration,
  - new customers are inserted into routes.
  - existing customers may be swapped among routes.

Intialization



Bipartite graph of insertions



Bipartite graph of insertions

Bipartite graph of insertions

- Every edge represents a legal insertion.

Bipartite graph of insertions

- Every edge represents a legal insertion.
- Find a matching to insert all customers into routes.

Bipartite graph of insertions

- Every edge represents a legal insertion.
- Find a matching to insert all customers into routes.
- This is formulated as an ILP model.

Bipartite graph of insertions

- Every edge represents a legal insertion.
- Find a matching to insert all customers into routes.
- This is formulated as an ILP model.
- Objective: Minimize the extra travel due to insertion of customers.

Bipartite graph of insertions

- Every edge represents a legal insertion.
- Find a matching to insert all customers into routes.
- This is formulated as an ILP model.
- Objective: Minimize the extra travel due to insertion of customers.
- Assigning a customer to an empty vehicle has high cost.

Solomon instances: 100 clustered customers, short routes

Instance	$\#Routes(B*)$	Travel dist.(B)	$\#Routes(H**)$	Travel dist.(H)
c101	10	828.94	10	828.94
c102	10	828.94	10	1036.83
c103	10	828.06	10	1119.47
c104	10	824.78	10	1312.11
c105	10	828.94	10	860.79
c106	10	828.94	10	900.64
c107	10	828.94	10	995.32
c108	10	828.94	10	912.73
c109	10	828.94	10	1111.33

$*$ Best known solution values.

$**$ Solution values of our heuristic.

Coded in JAVA, LP Solver: CPLEX 12.0, Processor: 2.7 GHz, RAM: 8GB.

Solomon instances: 100 clustered customers, long routes

Instance	$\#Routes(B*)$	Travel dist.(B)	$\#Routes(H**)$	Travel dist.(H)
c201	3	591.56	3	624.38
c202	3	591.56	3	908.14
c203	3	591.17	3	939.50
c204	3	590.6	3	1032.29
c205	3	588.88	3	731.37
c206	3	588.49	3	846.00
c207	3	588.29	4	749.532
c208	3	588.32	3	792.77

$*$ Best known solution values.

$**$ Solution values of our heuristic.

Coded in JAVA, LP Solver: CPLEX 12.0, Processor: 2.7 GHz, RAM: 8GB.

Solomon instances: 100 randomly located customers

Instance	$\#Routes(B*)$	Travel dist.(B)	$\#Routes(H**)$	Travel dist.(H)
r101	19	1650.8	19	1680.46
r102	17	1486.12	17	1577.41
r103	13	1292.68	13	1362.71
r104	9	1007.31	10	1173.78
r105	14	1377.11	14	1568.73
r106	12	1252.03	12	1568.73
r107	10	1104.66	11	1248.55
r108	9	960.88	10	1200.33
r109	11	1194.73	12	1372.47
r110	10	1118.84	11	1309.13
r111	10	1096.72	11	1246.59
r112	9	982.14	10	1129.07

$*$ Best known solution values.

$**$ Solution values of our heuristic.

Coded in JAVA, LP Solver: CPLEX 12.0, Processor: 2.7 GHz, RAM: 8GB.

Conclusions and further directions

- Our heuristic solves each instance within several minutes.

Conclusions and further directions

- Our heuristic solves each instance within several minutes.
- We construct good-quality solutions w.r.t number of vehicles.

Conclusions and further directions

- Our heuristic solves each instance within several minutes.
- We construct good-quality solutions w.r.t number of vehicles.
- There is (always) more room to improve this heuristic.

Conclusions and further directions

- Our heuristic solves each instance within several minutes.
- We construct good-quality solutions w.r.t number of vehicles.
- There is (always) more room to improve this heuristic.
- We will adapt this heuristic to VRPTW with stochastic travel times.

Conclusions and further directions

- Our heuristic solves each instance within several minutes.
- We construct good-quality solutions w.r.t number of vehicles.
- There is (always) more room to improve this heuristic.
- We will adapt this heuristic to VRPTW with stochastic travel times.
- We will use this heuristic in a Branch-and-Price method for VRPTW to find good routes.

Thanks for your attention