AE4441 - Operations Optimisation - The Vehicle Routing Problem (VRP)

Paul Roling, Alessandro Bombelli
p.c.roling@tudelft.nl, a.bombelli@tudelft.nl

Delft University of Technology, The Netherlands

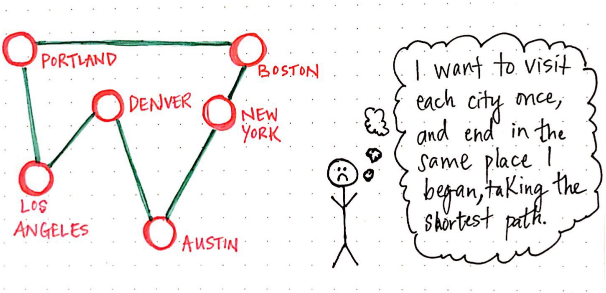
November 18, 2020

Table of Contents

The ancestor: the Traveling Salesman Problem (TSP)

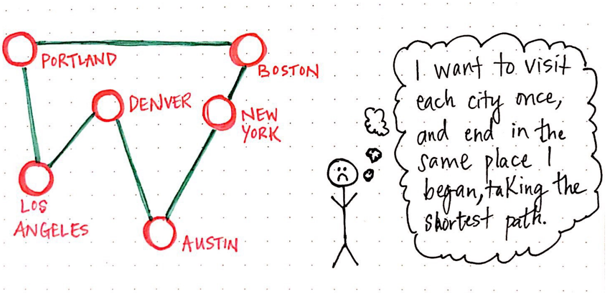
The ancestor: the Traveling Salesman Problem (TSP)

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?



The ancestor: the Traveling Salesman Problem (TSP)

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?



Let us analyze the MILP formulation

TSP: Dantzig–Fulkerson–Johnson formulation

Cities labeled as $\{1, \cdots, n\}$
Decision variable: x_{ij}=1 if salesman goes from city i to city j, 0 ow

TSP: Dantzig–Fulkerson–Johnson formulation

Cities labeled as $\{1, \cdots, n\}$
Decision variable: x_{ij}=1 if salesman goes from city i to city j, 0 ow

Objective function
min $\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij}x_{ij}$ (minimization of transportation cost)

TSP: Dantzig–Fulkerson–Johnson formulation

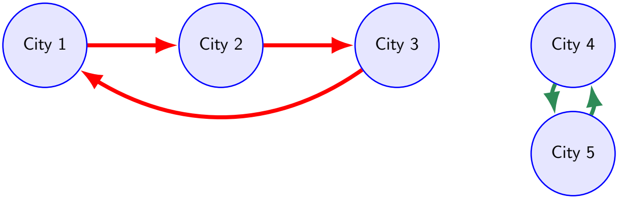
Cities labeled as $\{1, \cdots, n\}$
Decision variable: x_{ij}=1 if salesman goes from city i to city j, 0 ow

Objective function
$\min \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}$ (minimization of transportation cost)

Constraints
$\sum_{j=1, j\neq i}^{n} x_{ji} = 1, \;\; i = 1, \cdots, n$ (the salesman must enter every city)
$\sum_{j=1, j\neq i}^{n} x_{ij} = 1, \;\; i = 1, \cdots, n$ (the salesman must leave every city)

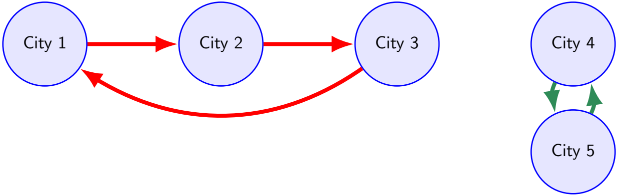
We are not done yet! **Subtour elimination constraints**

TSP: Dantzig–Fulkerson–Johnson formulation



This situation satisfies the “enter”-“exit” city constraints, but there are 2 subtours rather than a single one.

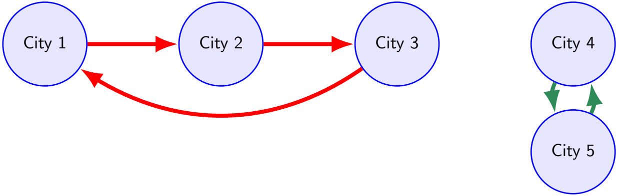
TSP: Dantzig–Fulkerson–Johnson formulation



This situation satisfies the "enter"-"exit" city constraints, but there are 2 subtours rather than a single one.

$\sum_{i \in S} \sum_{j \in S, j \neq i} x_{ij} \leq |S| - 1 \ \ \forall S \subset \{1, \cdots, n\}$ (subtour elimination constraints)

TSP: Dantzig–Fulkerson–Johnson formulation



This situation satisfies the "enter"-"exit" city constraints, but there are 2 subtours rather than a single one.

$\sum_{i \in S} \sum_{j \in S, j \neq i} x_{ij} \leq |S| - 1 \;\; \forall S \subset \{1, \cdots, n\}$ (subtour elimination constraints)

Exponential number of constraints! E.g., with five cities, we need all subtours with 2, 3, and 4 cities, i.e., $\binom{5}{2}+\binom{5}{3}+\binom{5}{4}$=25. As the number of cities increases, this set of constraints "explodes": solve problem $\rightarrow$ check violated subtour constraints, add them $\rightarrow$ solve problem again

The Vehicle Routing Problem (VRP)

- Central depot and a set of customers (nodes) $\{1, \cdots, n\}$. Central depot is divided into an origin and destination depot, resp. nodes 0 and $n+1$ (not only possible approach. Lecture 6 slides: different approach with ad-hoc variables if vehicle k is used)
- Fleet of vehicles $\{1, \cdots, K\}$
- Decision variable: x_{ij}^k=1 if vehicle k goes from node i to node j, 0 ow

The Vehicle Routing Problem (VRP)

- Central depot and a set of customers (nodes) $\{1, \cdots, n\}$. Central depot is divided into an origin and destination depot, resp. nodes 0 and $n+1$ (not only possible approach. Lecture 6 slides: different approach with ad-hoc variables if vehicle k is used)
- Fleet of vehicles $\{1, \cdots, K\}$
- Decision variable: x_{ij}^k=1 if vehicle k goes from node i to node j, 0 ow

Objective function

min $\sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=1, j\neq i}^{n+1} c_{ij} x_{ij}^k$ (minimization of transportation cost)

The Vehicle Routing Problem (VRP)

- Central depot and a set of customers (nodes) $\{1, \cdots, n\}$. Central depot is divided into an origin and destination depot, resp. nodes 0 and $n+1$ (not only possible approach. Lecture 6 slides: different approach with ad-hoc variables if vehicle k is used)
- Fleet of vehicles $\{1, \cdots, K\}$
- Decision variable: x_{ij}^k=1 if vehicle k goes from node i to node j, 0 ow

Objective function

min $\sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=1, j \neq i}^{n+1} c_{ij} x_{ij}^k$ (minimization of transportation cost)

Constraints

$\sum_{j=1}^{n+1} x_{0j}^k = 1, \;\; k = 1, \cdots, K$ (each vehicle must leave the depot)

$\sum_{j=0}^{n} x_{j,n+1}^k = 1, \;\; k = 1, \cdots, K$ (each vehicle must return to the depot)

The Vehicle Routing Problem (VRP)

$\sum_{k=1}^{K} \sum_{j=0, j \neq i}^{n} x_{ji}^{k} = 1, \;\; i = 1, \cdots, n$ (each customer must be visited by a vehicle)

$\sum_{j=0, j \neq i}^{n} x_{ji}^{k} = \sum_{j=1, j \neq i}^{n+1} x_{ij}^{k}, \;\; i = 1, \cdots, n, \;\; k = 1, \cdots, K$ (if a vehicle visits a customer, then the same vehicle must leave that customer)

The Vehicle Routing Problem (VRP)

$\sum_{k=1}^{K}\sum_{j=0,j\neq i}^{n} x_{ji}^{k} = 1, \ \ i = 1, \cdots, n$ (each customer must be visited by a vehicle)

$\sum_{j=0,j\neq i}^{n} x_{ji}^{k} = \sum_{j=1,j\neq i}^{n+1} x_{ij}^{k}, \ \ i = 1, \cdots, n, \ \ k = 1, \cdots, K$ (if a vehicle visits a customer, then the same vehicle must leave that customer)

$\sum_{i\in S}^{n}\sum_{j\in S,j\neq i}^{n} x_{ij}^{k} \leq |S| - 1, \ \ S \subset \{1, \cdots, n\}, \ \ k = 1, \cdots, K$ (subtour elimination constraints)

The Vehicle Routing Problem (VRP)

Constraints are generally imposed as a sequence of nested loops. As example, considering the constraint that every customer should be picked up by a truck, we can express it as

The Vehicle Routing Problem (VRP)

Constraints are generally imposed as a sequence of nested loops. As example, considering the constraint that every customer should be picked up by a truck, we can express it as

$\forall i \in \{\text{set customers}\}$

The Vehicle Routing Problem (VRP)

Constraints are generally imposed as a sequence of nested loops. As example, considering the constraint that every customer should be picked up by a truck, we can express it as

$\forall i \in \{\text{set customers}\}$

$\forall$ trucks

$\forall$ customers $j \neq i \rightarrow$ add x_{ji}^k

$$\sum_{k=1}^{K} \sum_{j=0, j\neq i}^{n} x_{ji}^k = 1, \;\; i = 1, \cdots, n$$

The Vehicle Routing Problem (VRP) - Variations

So far, the VRP looks extremely similar to the TSP. **There are many variations of the original VRP that include additional decision variables and constraints**

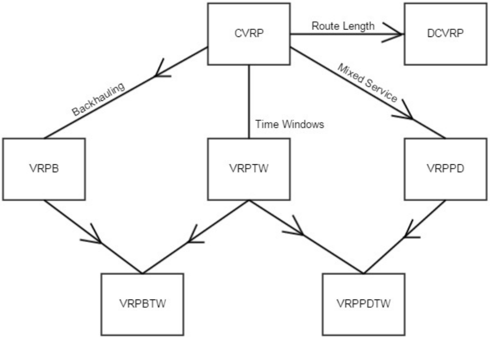


Figure: Subset of variations of the classic VRP formulation.

The Capacitated Vehicle Routing Problem (CVRP)

- each vehicle is characterized by a maximum transportable capacity Q (e.g., max. volume or weight)
- q_i is the demand of customer i

The Capacitated Vehicle Routing Problem (CVRP)

- each vehicle is characterized by a maximum transportable capacity Q (e.g., max. volume or weight)
- q_i is the demand of customer i

Additional constraints

$\sum_{i=0}^{n} \sum_{j=1, j \neq i}^{n} q_j x_{ij}^k \leq Q, \;\; k = \{1, \cdots, K\}$ (capacity constraints)

The Vehicle Routing Problem with Time Windows (VRPTW)

- each customer must be visited within a time window $[e_i, l_i]$
- τ_i is the continuous decision variable representing the start of service time at customer i
- p_i is the processing time at node i, t_{ij} is the travel time between nodes i and j

The Vehicle Routing Problem with Time Windows (VRPTW)

- each customer must be visited within a time window $[e_i, l_i]$
- τ_i is the continuous decision variable representing the start of service time at customer i
- p_i is the processing time at node i, t_{ij} is the travel time between nodes i and j

Additional constraints

$e_i \leq \tau_i \leq l_i, \;\; i = \{1, \cdots, n\}$ (lower and upper bound on τ_i)
$\tau_j \geq \tau_i + p_i + t_{ij} - (1 - \sum_{k=1}^{K} x_{ij}^{k})\mathbb{M} \;\; i = 1, \cdots, n, \; j = 1, \cdots, n \; (j \neq i)$
(time precedence constraints) $\rightarrow \mathbb{M}$ is a sufficiently big number (big-M formulation)

The Vehicle Routing Problem with Time Windows (VRPTW)

- each customer must be visited within a time window $[e_i, l_i]$
- τ_i is the continuous decision variable representing the start of service time at customer i
- p_i is the processing time at node i, t_{ij} is the travel time between nodes i and j

Additional constraints

$e_i \leq \tau_i \leq l_i, \;\; i = \{1, \cdots, n\}$ (lower and upper bound on τ_i)
$\tau_j \geq \tau_i + p_i + t_{ij} - (1 - \sum_{k=1}^{K} x_{ij}^k)\mathbb{M} \;\; i = 1, \cdots, n, \; j = 1, \cdots, n \; (j \neq i)$
(time precedence constraints) $\rightarrow \mathbb{M}$ is a sufficiently big number (big-M formulation)

If a vehicle is visiting i and then j, the visit time at j cannot be sooner than the visit time at i, plus the processing time at i and the traveling time from i to j

The Vehicle Routing Problem with Time Windows (VRPTW)

Example of time precedence constraint: suppose customer i must be visited within the $[60, 180]$ minutes time-interval, and that customer j must be visited within the $[100, 240]$ minutes time-interval. When setting up the model, we do not know if a vehicle will visit customers i and j in sequence beforehand. Hence, we need to write the time precedence constraint in a way that works either way.

The Vehicle Routing Problem with Time Windows (VRPTW)

Example of time precedence constraint: suppose customer i must be visited within the $[60, 180]$ minutes time-interval, and that customer j must be visited within the $[100, 240]$ minutes time-interval. When setting up the model, we do not know if a vehicle will visit customers i and j in sequence beforehand. Hence, we need to write the time precedence constraint in a way that works either way.

If $\sum_{k=1}^{K} x_{ij} = 1$, then we have $\tau_j \geq \tau_i + p_i + t_{ij}$. Assuming $p_i = 30$ minutes and $t_{ij} = 45$ minutes, and recalling we cannot visit i before $t = 60$ minutes we can write $\tau_j \geq 60 + 30 + 45 = 135$ minutes

The Vehicle Routing Problem with Time Windows (VRPTW)

Example of time precedence constraint: suppose customer i must be visited within the $[60, 180]$ minutes time-interval, and that customer j must be visited within the $[100, 240]$ minutes time-interval. When setting up the model, we do not know if a vehicle will visit customers i and j in sequence beforehand. Hence, we need to write the time precedence constraint in a way that works either way.

If $\sum_{k=1}^{K} x_{ij} = 1$, then we have $\tau_j \geq \tau_i + p_i + t_{ij}$. Assuming $p_i = 30$ minutes and $t_{ij} = 45$ minutes, and recalling we cannot visit i before $t = 60$ minutes we can write $\tau_j \geq 60 + 30 + 45 = 135$ minutes

If $\sum_{k=1}^{K} x_{ij} = 0$, then no vehicle goes from i to j. The constraint can be re-written as $\mathbb{M} \geq \tau_i + p_i + t_{ij} - \tau_j$. What is the minimum value of $\mathbb{M}$ that always satisfies the inequality? $\rightarrow$
$\mathbb{M} = \max\left[0, l_i + p_i + t_{ij} - e_j\right] = 155$ minutes

The Vehicle Routing Problem with Time Windows (VRPTW)

The role of $\mathbb{M}$ is similar to an IF-ELSE statement. IF $\sum_{k=1}^{K} x_{ij} = 1$, then the time precedence constraint between i and j must be satisfied. ELSE, the constraint cannot disappear, hence the resulting inequality should be satisfied no matter what.

The Vehicle Routing Problem with Time Windows (VRPTW)

The role of $\mathbb{M}$ is similar to an IF-ELSE statement. IF $\sum_{k=1}^{K} x_{ij} = 1$, then the time precedence constraint between i and j must be satisfied. ELSE, the constraint cannot disappear, hence the resulting inequality should be satisfied no matter what.

Since it is an inequality, and not an equality, a very high value of $\mathbb{M}$ would satisfy the inequality as well. Why did we need to compute that specific value?

The Vehicle Routing Problem with Time Windows (VRPTW)

The role of $\mathbb{M}$ is similar to an IF-ELSE statement. IF $\sum_{k=1}^{K} x_{ij} = 1$, then the time precedence constraint between i and j must be satisfied. ELSE, the constraint cannot disappear, hence the resulting inequality should be satisfied no matter what.

Since it is an inequality, and not an equality, a very high value of $\mathbb{M}$ would satisfy the inequality as well. Why did we need to compute that specific value?

That is the **minimum** value that always satisfies the inequality. Finding good lower bounds on $\mathbb{M}$ might avoid computational issues when running the branch-and-bound solver.

The Vehicle Routing Problem with Time Windows (VRPTW)

We do have more variables and constraints, but **we do not need to impose subtour elimination constraints because of time precedence constraints**

The Vehicle Routing Problem with Time Windows (VRPTW)

We do have more variables and constraints, but **we do not need to impose subtour elimination constraints because of time precedence constraints**

Example: suppose a subtour A-B-C-A. Writing time precedence constraints:

$\tau_B \geq \tau_A + p_A + t_{AB}$ (1)
$\tau_C \geq \tau_B + p_B + t_{BC}$ (2)
$\tau_A \geq \tau_C + p_C + t_{CA}$ (3)

where (1) and (3) are clearly in contradiction.

The Vehicle Routing Problem with Backhauls (VRPB)

After delivering demand to customers (linehaul), vehicles can pickup demand from other customers to be transported back to the depot (backhaul).

The Vehicle Routing Problem with Backhauls (VRPB)

After delivering demand to customers (linehaul), vehicles can pickup demand from other customers to be transported back to the depot (backhaul).

Interesting logistics application, because vehicles are used more efficiently. On the way back to the depot, instead of just driving empty, they can pickup demand that would need to be transported back to the depot anyway.

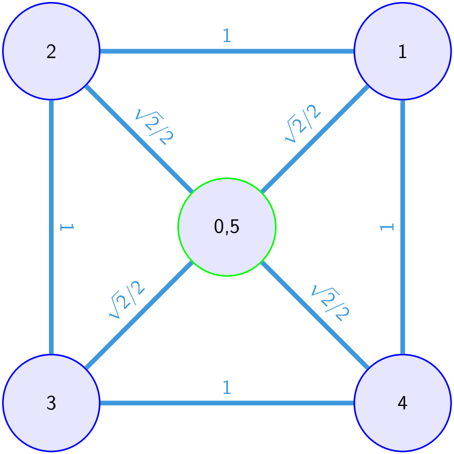
The Vehicle Routing Problem with Backhauls (VRPB)

After delivering demand to customers (linehaul), vehicles can pickup demand from other customers to be transported back to the depot (backhaul).

Interesting logistics application, because vehicles are used more efficiently. On the way back to the depot, instead of just driving empty, they can pickup demand that would need to be transported back to the depot anyway.

Change in network topology: backhaul customers can only be visited after all linehaul customers have been visited. Hence, there are no edges (i) from the depot to backhaul customers, (ii) from backhaul customers to linehaul customers.

VRP: implementation example



Note: this example is characterized by symmetry. For an asymmetric example, please check lecture 6

VRP: implementation example

- origin depot (node 0), customers (nodes $1, \cdots, 4$), destination depot (node 5). Origin and destination depot are the same depot physically, but different nodes from a graph perspective
- 2 vehicles
- $c_{ij} = d_{ij} \;\; \forall \;\; (i, j)$: our objective is to minimize traveled distance
- $q_1 = q_2 = q_3 = q_4 = 25$, $Q = 60$

VRP: implementation example - Objective function

```
Minimize
  0.7 x[0,1,0] + 0.7 x[0,2,0]
   + 0.7 x[0,3,0] + 0.7 x[0,4,0] + x[1,2,0]
   + 1.41 x[1,3,0] + x[1,4,0] + 0.7 x[1,5,0]
   + x[2,1,0] + x[2,3,0] + 1.41 x[2,4,0]
   + 0.7 x[2,5,0] + 1.41 x[3,1,0] + x[3,2,0]
   + x[3,4,0] + 0.7 x[3,5,0] + x[4,1,0]
   + 1.41 x[4,2,0] + x[4,3,0] + 0.7 x[4,5,0]
   + 0.7 x[0,1,1] + 0.7 x[0,2,1]
   + 0.7 x[0,3,1] + 0.7 x[0,4,1] + x[1,2,1]
   + 1.41 x[1,3,1] + x[1,4,1] + 0.7 x[1,5,1]
   + x[2,1,1] + x[2,3,1] + 1.41 x[2,4,1]
   + 0.7 x[2,5,1] + 1.41 x[3,1,1] + x[3,2,1]
   + x[3,4,1] + 0.7 x[3,5,1] + x[4,1,1]
   + 1.41 x[4,2,1] + x[4,3,1] + 0.7 x[4,5,1]
```

VRP: implementation example - Decision variables

```
Binaries
 x[0,1,0] x[0,2,0] x[0,3,0] x[0,4,0] x[0,5,0]
 x[1,2,0] x[1,3,0] x[1,4,0] x[1,5,0] x[2,1,0]
 x[2,3,0] x[2,4,0] x[2,5,0] x[3,1,0] x[3,2,0]
 x[3,4,0] x[3,5,0] x[4,1,0] x[4,2,0] x[4,3,0]
 x[4,5,0] x[0,1,1] x[0,2,1] x[0,3,1] x[0,4,1]
 x[0,5,1] x[1,2,1] x[1,3,1] x[1,4,1] x[1,5,1]
 x[2,1,1] x[2,3,1] x[2,4,1] x[2,5,1] x[3,1,1]
 x[3,2,1] x[3,4,1] x[3,5,1] x[4,1,1] x[4,2,1]
 x[4,3,1] x[4,5,1]
```

VRP: implementation example - Constraints

Note: not all constraints are shown to keep it reasonably short

```
Subject To
 OrigDepot_0: x[0,1,0] + x[0,2,0] + x[0,3,0] + x[0,4,0]
 + x[0,5,0] = 1
 OrigDepot_1: x[0,1,1] + x[0,2,1] + x[0,3,1] + x[0,4,1]
 + x[0,5,1] = 1
 DestDepot_0: x[0,5,0] + x[1,5,0] + x[2,5,0] + x[3,5,0]
 + x[4,5,0] = 1
 DestDepot_1: x[0,5,1] + x[1,5,1] + x[2,5,1] + x[3,5,1]
 + x[4,5,1] = 1
```

Vehicles must leave the origin depot ("OrigDepot") and go back to destination depot ("DestDepot")

VRP: implementation example - Constraints

```
 VisitCustomer_1: x[0,1,0] + x[2,1,0] + x[3,1,0] + x[4,1,0]
 + x[0,1,1] + x[2,1,1] + x[3,1,1] + x[4,1,1] = 1
 FlowBalance_1_0: x[0,1,0] - x[1,2,0] - x[1,3,0] - x[1,4,0]
 - x[1,5,0] + x[2,1,0]
 + x[3,1,0] + x[4,1,0] = 0
 FlowBalance_1_1: x[0,1,1] - x[1,2,1] - x[1,3,1] - x[1,4,1]
 - x[1,5,1] + x[2,1,1]
 + x[3,1,1] + x[4,1,1] = 0
Capacity_0: 25 x[0,1,0] + 25 x[0,2,0] + 25 x[0,3,0]
 + 25 x[0,4,0] + 25 x[1,2,0] + 25 x[1,3,0] + 25 x[1,4,0]
 + 25 x[2,1,0] + 25 x[2,3,0] + 25 x[2,4,0] + 25 x[3,1,0]
 + 25 x[3,2,0] + 25 x[3,4,0] + 25 x[4,1,0] + 25 x[4,2,0]
 + 25 x[4,3,0] <= 60
```

Customer 1 must be visited ("VisitCustomer"). If a vehicle visits customer 1, then it must leave customer 1 ("FlowBalance"). Capacity constraint ("Capacity")

VRP: implementation example - Constraints

```
Subtour_1_2_0: x[1,2,0] + x[2,1,0]  <= 1
Subtour_1_3_0: x[1,3,0] + x[3,1,0]  <= 1
Subtour_1_2_3_0: x[1,2,0] + x[1,3,0] + x[2,1,0]
+ x[2,3,0] + x[3,1,0] + x[3,2,0]  <= 2
```

Subtour elimination constraints ("Subtour")

VRP: implementation example - Solution

Optimal solution found (tolerance 5.00e-02) Best objective 4.83e+00, best bound 4.83e+00, gap 0.0000%

The two routes are 0-1-2-5 and 0-4-3-5 (**Note**: there exist multiple equivalent optimal solutions because of symmetry) $\rightarrow$ **Case 1**

VRP: implementation example - Solution

Optimal solution found (tolerance 5.00e-02) Best objective 4.83e+00, best bound 4.83e+00, gap 0.0000%

The two routes are 0-1-2-5 and 0-4-3-5 (**Note**: there exist multiple equivalent optimal solutions because of symmetry) → **Case 1**

What happens if we increase Q to 100?

Optimal solution found (tolerance 5.00e-02) Best objective 4.4e+00, best bound 4.4e+00, gap 0.0000%

Now, a single vehicle is sufficient, with route 0-4-3-2-1-5. The second route is the zero-cost zero-distance route 0-5 → **Case 2**

VRP: implementation example - Solution

Optimal solution found (tolerance 5.00e-02) Best objective 4.83e+00, best bound 4.83e+00, gap 0.0000%

The two routes are 0-1-2-5 and 0-4-3-5 (**Note**: there exist multiple equivalent optimal solutions because of symmetry) $\rightarrow$ **Case 1**

What happens if we increase Q to 100?

Optimal solution found (tolerance 5.00e-02) Best objective 4.4e+00, best bound 4.4e+00, gap 0.0000%

Now, a single vehicle is sufficient, with route 0-4-3-2-1-5. The second route is the zero-cost zero-distance route 0-5 $\rightarrow$ **Case 2**

What if we add a maximum route length per vehicle equal to 2 with constraint set $\sum_{i=1}^{n} d_{0i}x_{0i}^{k} + \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} d_{ij}x_{ij}^{k} + \sum_{i=1}^{n} d_{i,n+1}x_{i,n+1}^{k} \leq 2 \quad k = 1, \cdots, K$?

Model is infeasible Best objective -, best bound -, gap -

We cannot carry out all deliveries with just 2 vehicles $\rightarrow$ **Case 3**

VRP: implementation example - Solution

What if we only have 2 vehicles? $\rightarrow$ change in objective function.
Minimize penalty incurred when a customer is not served

VRP: implementation example - Solution

What if we only have 2 vehicles? $\rightarrow$ change in objective function. Minimize penalty incurred when a customer is not served

Define $z_i = 1$ if customer i is not served and 0 ow, while $\mathbb{P}_i$ is a (monetary) penalty if customer i is not served

VRP: implementation example - Solution

What if we only have 2 vehicles? $\rightarrow$ change in objective function. Minimize penalty incurred when a customer is not served

Define $z_i = 1$ if customer i is not served and 0 ow, while $\mathbb{P}_i$ is a (monetary) penalty if customer i is not served

The new objective function is $\min \sum_{i=1}^{n} \mathbb{P}_i z_i$, and we can relax one of the basic constraints as $\sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{j=0, j \neq i}^{n} x_{ji}^k + z_i = 1, \ \ i = 1, \cdots, n$
$\rightarrow$ **customer i is either served by a vehicle, or not served at all**

VRP: implementation example - Solution

What if we only have 2 vehicles? $\rightarrow$ change in objective function. Minimize penalty incurred when a customer is not served

Define $z_i = 1$ if customer i is not served and 0 ow, while $\mathbb{P}_i$ is a (monetary) penalty if customer i is not served

The new objective function is $\min \sum_{i=1}^{n} \mathbb{P}_i z_i$, and we can relax one of the basic constraints as $\sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{j=0, j \neq i}^{n} x_{ji}^{k} + z_i = 1, \quad i = 1, \cdots, n$
$\rightarrow$ **customer i is either served by a vehicle, or not served at all**

Setting $\mathbb{P}_1 = 10,000$, $\mathbb{P}_2 = 100$, $\mathbb{P}_3 = 10,000$, and $\mathbb{P}_4 = 5,000$, we obtain

Optimal solution found (tolerance 5.00e-02) Best objective 5.1e+03, best bound 5.1e+03, gap 0.0000% $\rightarrow$ customers 2 and 4 ("cheaper") are not visited $\rightarrow$ **Case 4**

VRP: implementation example - Solution

Now, let us remove the maximum route length constraint, go back to the original objective function (minimization of distance traveled), but introduce **traveling times** and **time windows** (in generic time units) as follows:

VRP: implementation example - Solution

Now, let us remove the maximum route length constraint, go back to the original objective function (minimization of distance traveled), but introduce **traveling times** and **time windows** (in generic time units) as follows:

1: $[0, 6]$, 2: $[0, 1.1]$, 3: $[0, 1.1]$, 4: $[1, 3]$
$t_{01} = t_{02} = t_{03} = t_{04} = t_{15} = t_{25} = t_{35} = t_{45} = 1$
$t_{12} = t_{23} = t_{34} = t_{41} = 2$ (and vice versa)
$t_{13} = t_{24} = 2.5$ (and vice versa)

VRP: implementation example - Solution

Now, let us remove the maximum route length constraint, go back to the original objective function (minimization of distance traveled), but introduce **traveling times** and **time windows** (in generic time units) as follows:

1: $[0, 6]$, 2: $[0, 1.1]$, 3: $[0, 1.1]$, 4: $[1, 3]$
$t_{01} = t_{02} = t_{03} = t_{04} = t_{15} = t_{25} = t_{35} = t_{45} = 1$
$t_{12} = t_{23} = t_{34} = t_{41} = 2$ (and vice versa)
$t_{13} = t_{24} = 2.5$ (and vice versa)

Optimal solution found (tolerance 5.00e-02) Best objective 4.83e+00, best bound 4.83e+00, gap 0.0000%

The two routes are 0-2-1-5 and 0-3-4-5 $\rightarrow$ customers 2 and 3 are served first because of time window restrictions $\rightarrow$ **Case 5**

VRP: Implementation Example - Solution

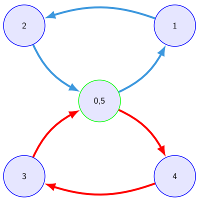


Figure: Solution for **Case 1**.

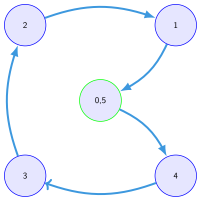


Figure: Solution for **Case 2**.

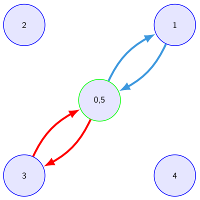


Figure: Solution for **Case 4**.

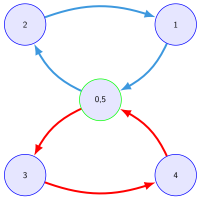


Figure: Solution for **Case 5**.

A recap on Branch and Bound

When using a **Branch and Bound** solver, a tree structure is created where every node corresponds to a different relaxation of the original problem with some **binary variables** $x \in \{0, 1\}$ relaxed to be **continuous variables** $0 \leq x \leq 1$. Every node (relaxed problem) is solved using the **simplex method**

A recap on Branch and Bound

When using a **Branch and Bound** solver, a tree structure is created where every node corresponds to a different relaxation of the original problem with some **binary variables** $x \in \{0, 1\}$ relaxed to be **continuous variables** $0 \leq x \leq 1$. Every node (relaxed problem) is solved using the **simplex method**

The very first node is a special one. It is called **root node** and **every** binary variable is relaxed. What can happen when we solve the associated problem? (i) optimal solution satisfies all integrality constraints $\rightarrow$ we are done, (ii) optimal solution does not satisfy all integrality constraints $\rightarrow$ we need to **branch by creating two sub-problems for every node** with more restricting integrality constraints

A recap on Branch and Bound

When solving a node (for a **minimization problem**), the following cases can occur:

- problem is infeasible. No need to further branch that node (node is **fathomed**
- problem has a feasible solution that satisfies all integrality constraints. Is that solution the best (lowest) until that point? That solution is the new **best incumbent**. Otherwise, the solution is discarded
- problem does not satisfy all integrality constraints. Is the solution higher that the best incumbent? Node is fathomed. Otherwise, keep branching

A recap on Branch and Bound

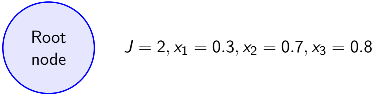


Figure: Example with 3 binary variables x_1, x_2, and x_3.

A recap on Branch and Bound

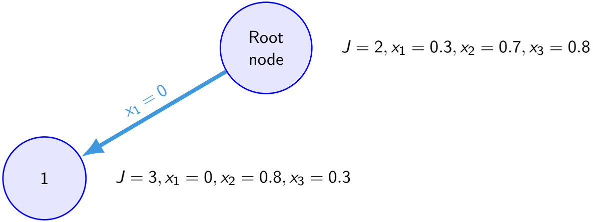


Figure: Example with 3 binary variables x_1, x_2, and x_3.

A recap on Branch and Bound

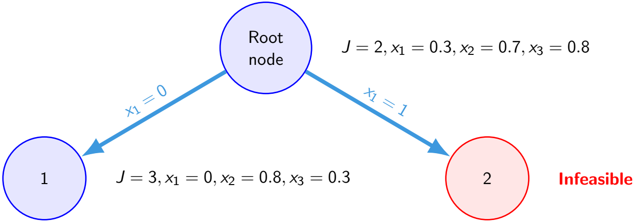


Figure: Example with 3 binary variables x_1, x_2, and x_3.

A recap on Branch and Bound

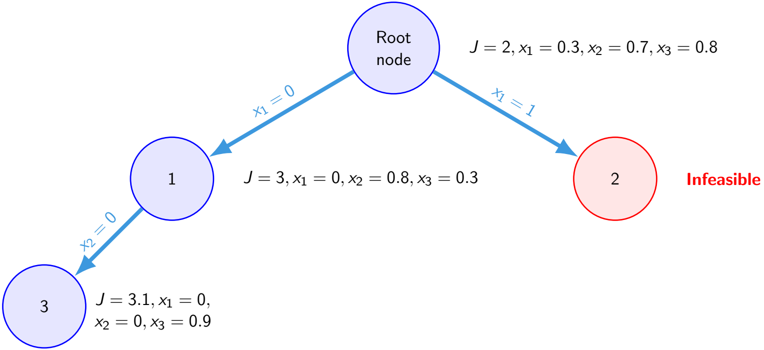


Figure: Example with 3 binary variables x_1, x_2, and x_3.

A recap on Branch and Bound

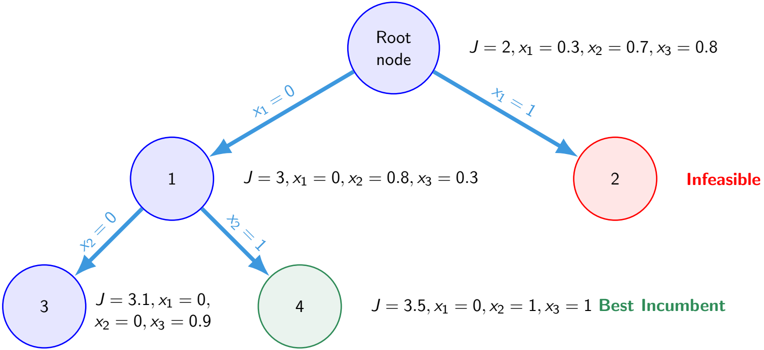


Figure: Example with 3 binary variables x_1, x_2, and x_3.

A recap on Branch and Bound

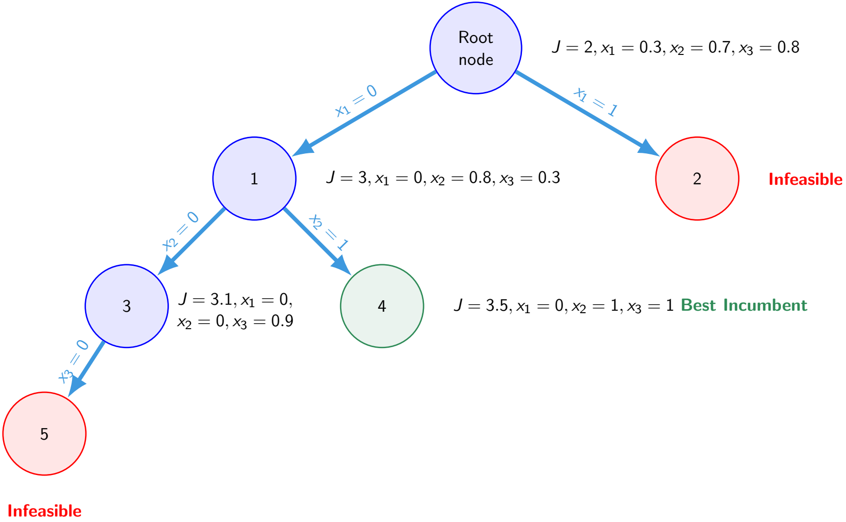


Figure: Example with 3 binary variables x_1, x_2, and x_3.

A recap on Branch and Bound

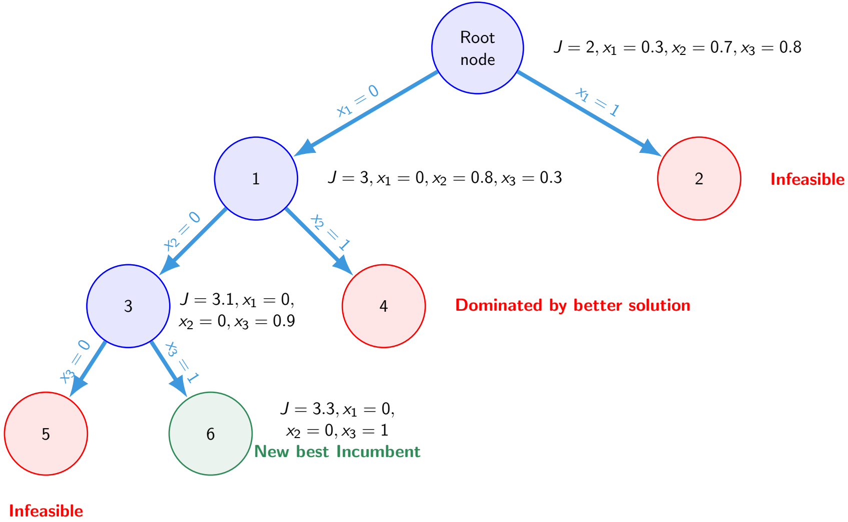


Figure: Example with 3 binary variables x_1, x_2, and x_3.

A recap on Branch and Bound

When does the Branch and Bound algorithm stop? When the **gap optimality** is zero or lower than a pre-defined threshold.

A recap on Branch and Bound

When does the Branch and Bound algorithm stop? When the **gap optimality** is zero or lower than a pre-defined threshold.

Gap optimality: percentile relative difference between **best incumbent** and **lower bound** (for a minimization problem). The lower bound is the lowest objective function of a node that still needs to be further branched. As the branch and bound progresses, the best incumbent can only decrease, while the lower bound can only increase. When gap optimality is zero, the **optimal solution** has been found

A recap on Branch and Bound

When does the Branch and Bound algorithm stop? When the **gap optimality** is zero or lower than a pre-defined threshold.

Gap optimality: percentile relative difference between **best incumbent** and **lower bound** (for a minimization problem). The lower bound is the lowest objective function of a node that still needs to be further branched. As the branch and bound progresses, the best incumbent can only decrease, while the lower bound can only increase. When gap optimality is zero, the **optimal solution** has been found

Does a high value of gap optimality mean we are far from an optimal solution? Not necessarily. We might have already found the optimal solution, but many unexplored nodes remain, hence the lower bound is still very low

The tip of the iceberg

This presentation covered only a (small) subset of variations of the VRP. The literature is much richer, depending on the specific application of interest.

How to structure the assignment

- Start with a simple VRP model (maybe capacitated and with time-windows?)

How to structure the assignment

- Start with a simple VRP model (maybe capacitated and with time-windows?)
- Create your own dataset (insights into the solution already)

How to structure the assignment

- Start with a simple VRP model (maybe capacitated and with time-windows?)
- Create your own dataset (insights into the solution already)
- Solve the model with Branch and Bound

How to structure the assignment

- Start with a simple VRP model (maybe capacitated and with time-windows?)
- Create your own dataset (insights into the solution already)
- Solve the model with Branch and Bound
- Make the model (and/or the dataset) more complicated

How to structure the assignment

- Start with a simple VRP model (maybe capacitated and with time-windows?)
- Create your own dataset (insights into the solution already)
- Solve the model with Branch and Bound
- Make the model (and/or the dataset) more complicated
- Post-processing: (1) sensitivity analysis, (2) data visualization, (3) conclusions and recommendations

AE4441 - Operations Optimisation - The Vehicle Routing Problem (VRP)

Paul Roling, Alessandro Bombelli
p.c.roling@tudelft.nl, a.bombelli@tudelft.nl

Delft University of Technology, The Netherlands

November 18, 2020