Towards a general framework for the Repositioning Problem in Bicycle-sharing Systems

Juan David Palacio Domínguez
PhD Student in Mathematical Engineering

Juan Carlos Rivera Agudelo
Thesis Advisor

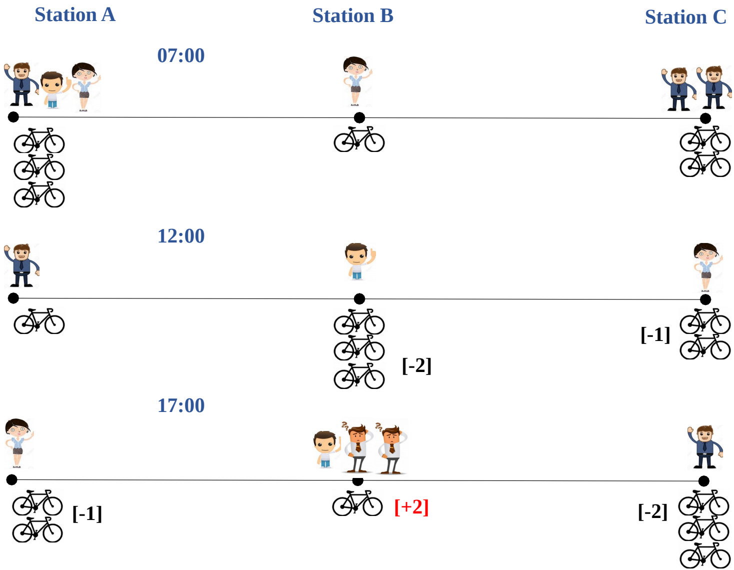
Doctoral Seminar in Mathematical Engineering

UNIVERSIDAD EAFIT

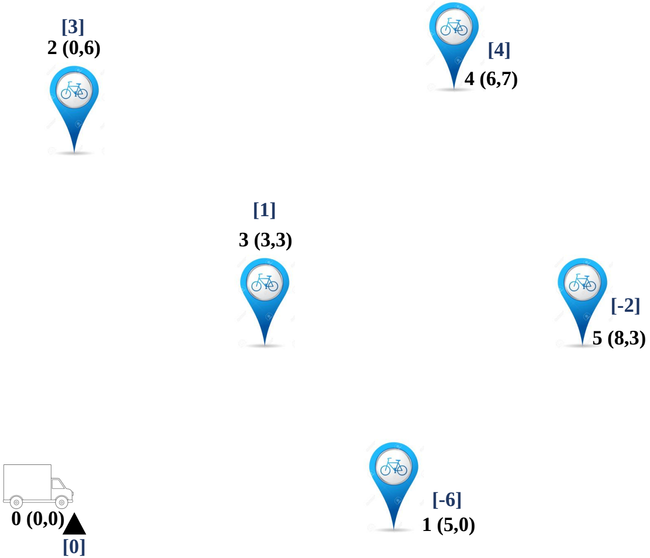
October 20, 2017

Outline

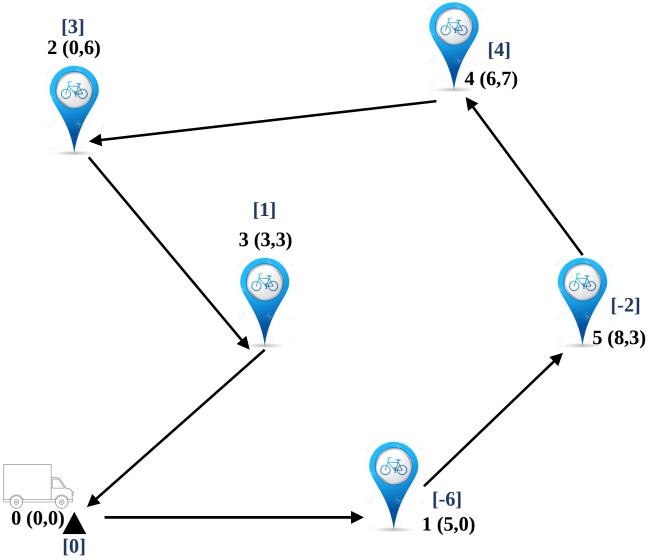
Balancing a BSS



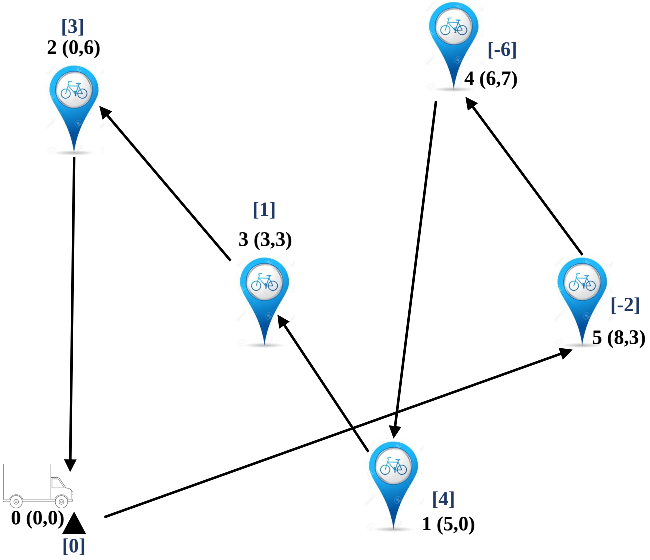
Pick up and Delivery TSP



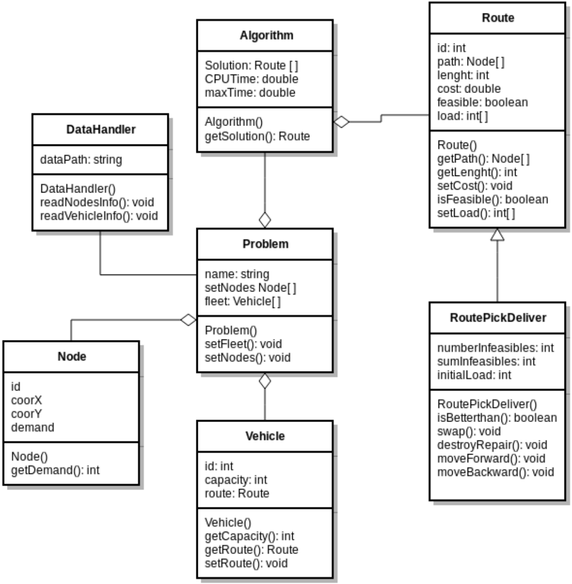
Pick up and Delivery TSP



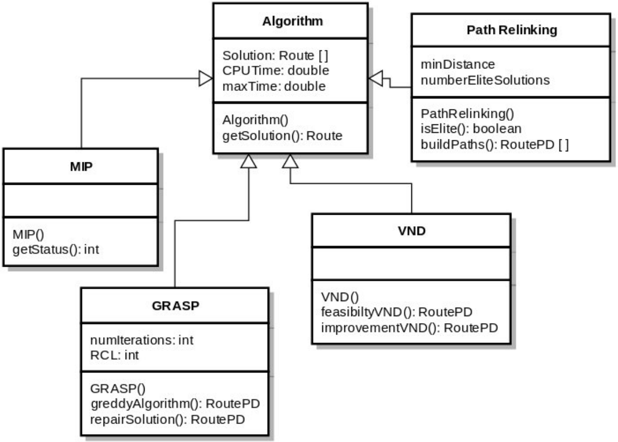
Pick up and Delivery TSP



General Framework for the RP



General Framework for the RP



Outline

Solution Strategies - Single Vehicle Case

- Mathematical Formulations
  - Traveling Salesman Problem (TSP)
  - Pick up and Delivery TSP (PDTSP)
  - PDTSP with Split Demand (PDTSPSD)
- Heuristic Algorithms
  - Nearest Neighbor (TSP)
  - Extensions of Nearest Neighbor for PDTSP and PDTSPSD
- Metaheuristic Algorithms
  - Greedy Randomized Adaptive Search Procedure (GRASP)
  - Path Relinking
  - Variable Neighborhood Descent (VND)

GRASP Algorithm

$f^* \leftarrow \infty$;
for $i = 1$ to *GRASPIterations* **do**
  $S \leftarrow$GreedyRandomAlgorithm();
  $S \leftarrow$LocalSearch(S);
  if $f(S) < f^*$ **then**
    $S^* \leftarrow S$;
    $f^* \leftarrow f(S)$;
  end if
end for
return S^*;

Metaheuristic Algorithms

GRASP + VND

$f^* \leftarrow \infty$;
for $i = 1$ to *GRASPIterations* **do**
  S ←GreedyRandomAlgorithm();
  S ←VND(S);
  if $f(S) < f^*$ **then**
    $S^* \leftarrow S$;
    $f^* \leftarrow f(S)$;
  end if
end for
return S^*;

Metaheuristic Algorithms

GRASP + VND + Post-Optimization

$f^* \leftarrow \infty$;
for $i = 1$ to *GRASPIterations* **do**
  $S \leftarrow$GreedyRandomAlgorithm();
  $S \leftarrow$VND(S);
  if $f(S) < f^*$ **then**
    $S^* \leftarrow S$;
    $f^* \leftarrow f(S)$;
  end if
end for
$S^* \leftarrow$VND'(S^*);
return S^*;

GRASP + VND + Post-Optimization with Path Relinking

$f^* \leftarrow \infty$;
$\xi \leftarrow \emptyset$;
for $i = 1$ to *GRASPIterations* **do**
  $S \leftarrow$GreedyRandomAlgorithm();
  $S \leftarrow$VND(S);
  if $f(S) < f^*$ **then**
    $S^* \leftarrow S$;
    $f^* \leftarrow f(S)$;
  end if
  if isElite(S)=**true then**
    $\xi \leftarrow \xi \cup S$;
  end if
end for
$S^* \leftarrow$PathRelinking(ξ);
return S^*;

Path Relinking

- Distance between solutions i and j: $\Delta(S_i, S_j)$

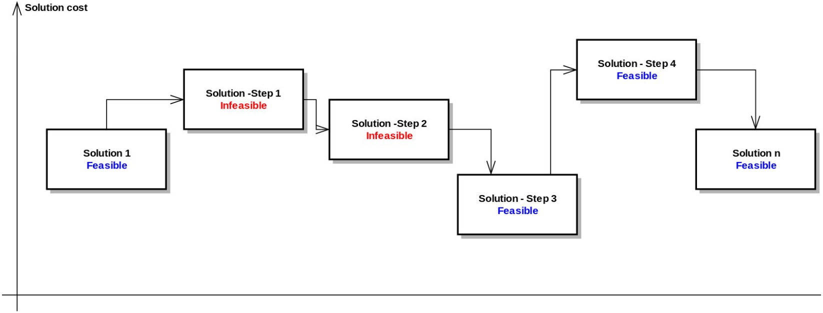
Solutions									
S_i	0	3	4	7	1	2	6	5	0
S_j	0	1	2	3	4	5	6	7	0

$$\Delta(S_i, S_j) = 6$$

- Distance between solution i and the elite solutions set: $\Delta(S_i, \xi)$

$$\Delta(S_i, \xi) = \min_{S_k \in \xi}\{\Delta(S_i, S_k)\}$$

Path Relinking



Path Relinking

Table: Path Relinking - Forward Strategy

	Paths									Distance to S_f
S_0	0	3	4	7	1	2	6	5	0	6
S_1	0	1	2	3	4	7	6	5	0	4
S_2	0	1	2	3	4	5	7	6	0	3
S_3	0	1	2	3	4	5	6	7	0	0
S_f	0	1	2	3	4	5	6	7	0	

Path Relinking

Table: Path Relinking - Backward Strategy

	Path									Distance to S_f
S_0	0	1	2	3	4	5	6	7	0	6
S_1	0	3	4	1	2	5	6	7	0	5
S_2	0	3	4	7	1	2	5	6	0	3
S_3	0	3	4	7	1	2	6	5	0	0
S_f	0	3	4	7	1	2	6	5	0	

VND Structure

Five neighborhoods (so far) within a VND method

- Forward insertion
- Backward insertion
- Swap
- 2-Opt
- Destroy and Repair
- A network-based neighborhood (an idea...)

VND - Destroy and Repair

- Destroy and Repair

Route	0	5	3	2	1	4	0
q	0	-2	1	3	-6	4	
Load	0	2	1	-2	4	0	

n	s
1	2

n: number of infeasible loads

s: sum of infeasible loads

- Randomly delete *m* stations from the path

Route	0	5	~~3~~	2	~~1~~	4	0
q	0	-2	~~1~~	3	~~-6~~	4	
Load	0	2	1	-2	4	0	

n	s
1	2

n: number of infeasible loads

s: sum of infeasible loads

VND - Destroy and Repair

- Compute the new incomplete tour and its load

Removed stations: 1 and 3 where $q_1 = -6$ and $q_3 = 1$

Route	0	5	2	4	0
q	0	-2	3	4	
Load	0	2	-1	-5	

n	s
2	6

n: number of infeasible loads

s: sum of infeasible loads

VND - Destroy and Repair

- Insert the removed stations trying to avoid infeasibility

Removed stations: 1 and 3 where $q_1 = -6$ and $q_3 = 1$

Route	0	5	1	2	4	0
q	0	-2	-6	3	4	
Load	0	2	8	5	1	

n	s
0	0

n: number of infeasible loads

s: sum of infeasible loads

VND - Destroy and Repair

- Insert the removed stations trying to avoid infeasibility

Removed stations: 3 where $q_3 = 1$

Route	0	5	1	2	4	3	0	n	s
q	0	-2	-6	3	4	1			
Load	0	2	8	5	1	0		0	0

n: number of infeasible loads

s: sum of infeasible loads

VND - A network-based neighborhood

Route	0	5	3	2	1	4	0
q	0	-2	1	3	-6	4	
Load	0	2	1	-2	4	0	

n	s
1	2

n: number of infeasible loads

s: sum of infeasible loads

- Remove *m* nodes from the solution
- Is it possible to find the best position to insert them again?

VND - A network-based neighborhood

Route	0	5	3	2	1	4	0
q	0	-2	1	3	-6	4	
Load	0	2	1	-2	4	0	

n	s
1	2

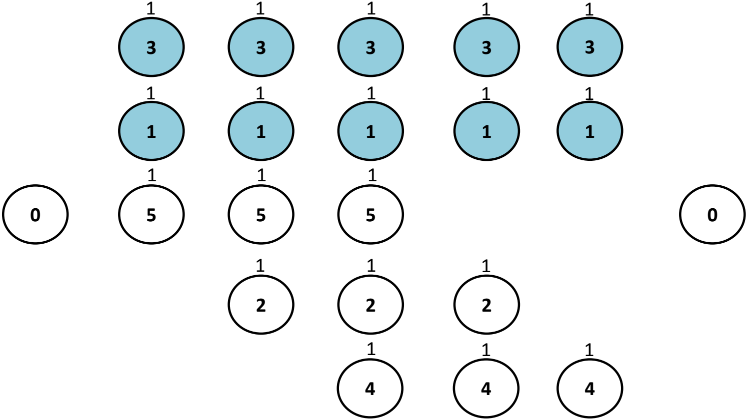
n: number of infeasible loads

s: sum of infeasible loads

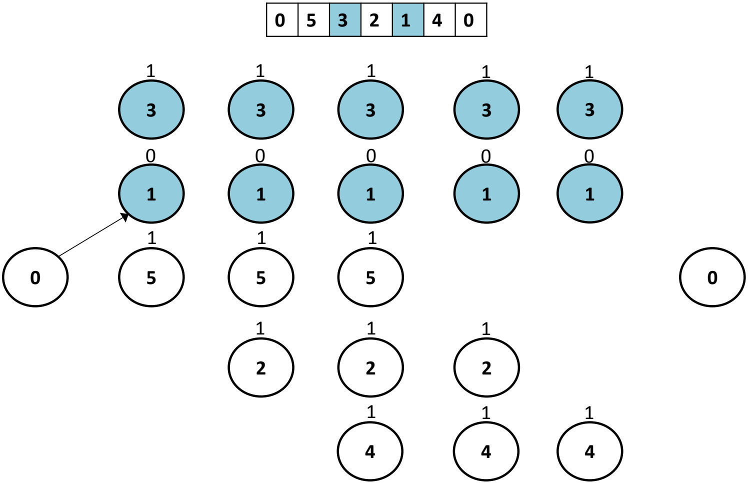
- Remove *m* nodes from the solution
- Is it possible to find the **best** position to insert them again?

VND - A network-based neighborhood

0	5	3	2	1	4	0

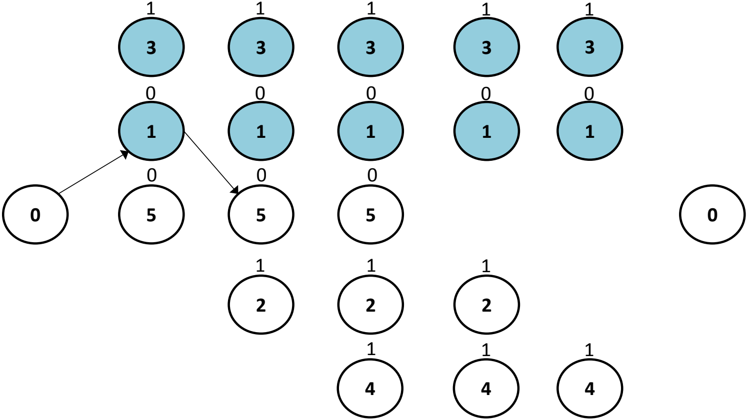


VND - A network-based neighborhood

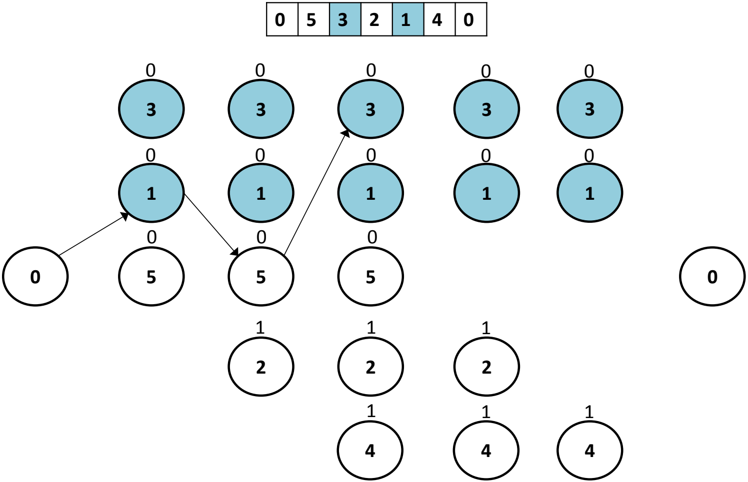


VND - A network-based neighborhood

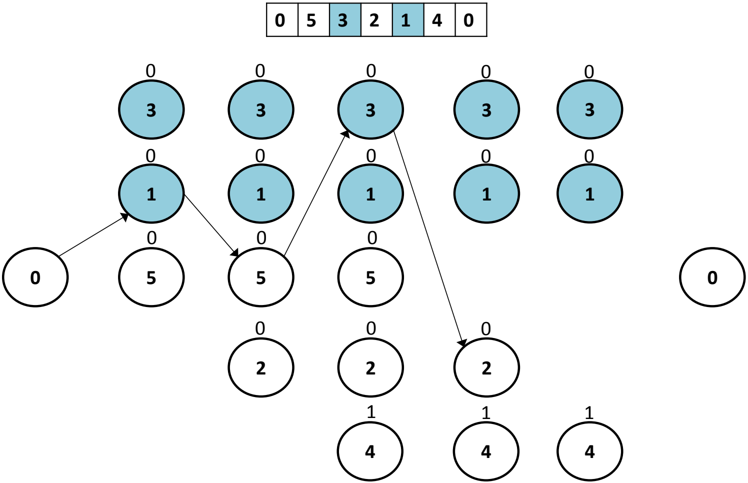
0	5	3	2	1	4	0



VND - A network-based neighborhood

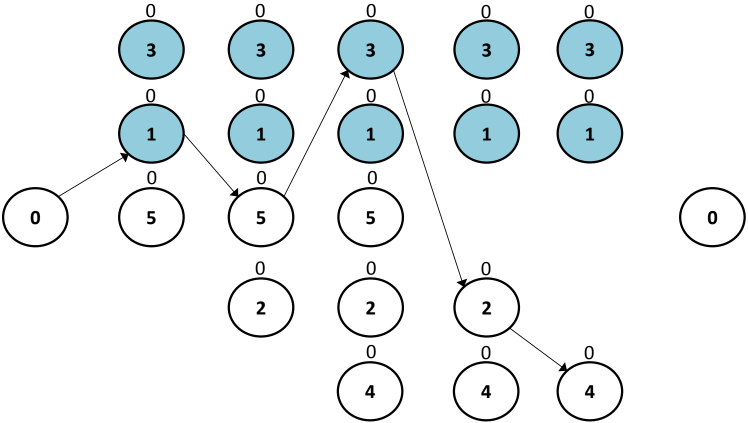


VND - A network-based neighborhood

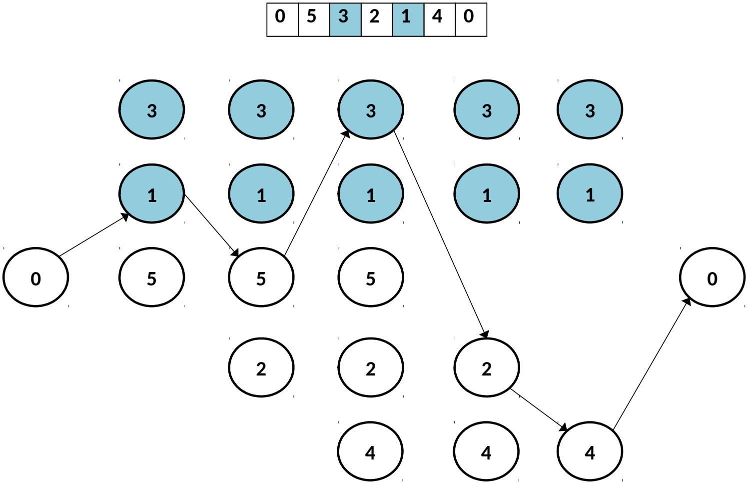


VND - A network-based neighborhood

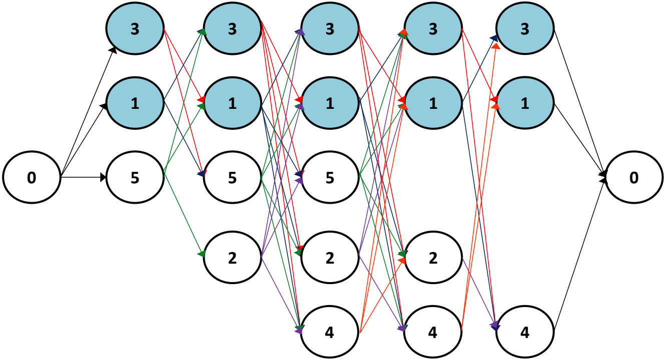
0	5	3	2	1	4	0



VND - A network-based neighborhood



VND - A network-based neighborhood



How to solve it?

- Constrained Shortest Path algorithms
- Nearest Neighbor with lower bounds computation

Outline

Mathematical Formulation for the HFPDVRP

- Sets
  - $\mathcal{N}$: Set of stations
  - $\mathcal{V}$: Set of vehicles
- Parameters
  - c_{ij} : Traveling cost from station i to station j
  - q_i : Demand or slack of bicycles in station i
  - Q^v : Capacity of vehicle v
- Decision Variables
  - $w_i^v = \begin{cases} 1 & \text{if station } i \text{ is visited by vehicle } v \\ 0 & \text{otherwise} \end{cases}$
  - $y_{ij}^v = \begin{cases} 1 & \text{if arc } (i,j) \text{ is transversed by vehicle } v \\ 0 & \text{otherwise} \end{cases}$
  - x_{ij}^v : Load of vehicle v when traveling from i to j
  - z_{ij}^v : Position of arc (i,j) in the route of vehicle v

Mathematical Formulation for the HFPDVRP

$$\min f = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} c_{ij} \cdot \sum_{v \in \mathcal{V}} y_{ij}^{v}$$

subject to,

$$\sum_{j \in \mathcal{N}, i \neq j} y_{ij}^{v} = w_{i}^{v} \qquad \forall\ i \in \mathcal{N} \setminus \{0\}, v \in \mathcal{V}$$

$$\sum_{j \in \mathcal{N}, j \neq 0} y_{0j}^{v} = 1 \qquad \forall\ v \in \mathcal{V}$$

$$\sum_{i \in \mathcal{N}} y_{ij}^{v} = \sum_{i \in \mathcal{N}} y_{ji}^{v} \qquad \forall\ j \in \mathcal{N}, v \in \mathcal{V}$$

$$x_{ij}^{v} \leq Q^{v} \cdot y_{ij}^{v} \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}$$

Mathematical Formulation for the HFPDVRP

$$\sum_{k \in \mathcal{N}} x_{ki}^{v} - \sum_{j \in \mathcal{N}} x_{ij}^{v} = q_i \cdot w_i^{v} \qquad \forall\, i \in \mathcal{N}, v \in \mathcal{V}$$

$$\sum_{k \in \mathcal{N}} z_{ki}^{v} - \sum_{j \in \mathcal{N}} z_{ij}^{v} = w_i^{v} \qquad \forall\, i \in \mathcal{N} \setminus \{0\}, v \in \mathcal{V}$$

$$z_{ij}^{v} \leq |\mathcal{N}| \cdot y_{ij}^{v} \qquad \forall\, i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}$$

$$w_i^{v} \in \{0, 1\} \qquad \forall\, i \in \mathcal{N}, v \in \mathcal{V}$$

$$y_{ij}^{v} \in \{0, 1\} \qquad \forall\, i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}$$

$$z_{ij}^{v} \in \mathcal{Z}^{+} \cup \{0\} \qquad \forall\, i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}$$

$$x_{ij}^{v} \geq 0 \qquad \forall\, i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}$$

- Dataset
  - Instances adapted from TSPLib Library (elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html)
  - Instances with 9, 14, 16, 22, 29, 42 nodes were tested
- Software
  - All the algorithms were implemented on C++
  - Mathematical models were solved using Gurobi Optimizer 7.1
- Computer features
  - Intel Core i7, 64Gb RAM.
  - OS: Linux - Debian 8 (x86-64)

Preliminary Results - Homogeneous Fleet

- $Q = 10$
- $\max_{i \in \mathcal{N}}\{|q_i|\} = 10$

	$\|V\| = 1$		$\|V\| = 2$		$\|V\| = 3$	
$\|N\|$	Distance	CPU time (s)	Distance	CPU time (s)	Distance	CPU time (s)
9	26	0.19	21	0.16	-	-
14	24	0.07	21	1.05	20	2.52
16	61	0.39	53	0.95	51	2.09
22	36	0.68	30	10.83	26	49.01
29	10 957	223.73	9 932	2 348.11	9 022	1488.22

Preliminary Results - Heterogeneous Fleet

- $Q_1 = 10$
- $Q_2 = 8$
- $Q_3 = 8$
- $\max_{i \in \mathcal{N}}\{|q_i|\} = 10$

	$\|V\| = 2$		$\|V\| = 3$	
$\|N\|$	Distance	CPU time (s)	Distance	CPU time (s)
9	21	0.21	-	-
14	21	0.52	20	2.80
16	53	0.84	51	1.96
22	32	11.92	32	72.39
29	10 331	365.98	10 052	534.93

Preliminary Results - Heterogeneous Fleet

- $Q_1 = 12$
- $Q_2 = 10$
- $Q_3 = 8$
- $\max_{i \in \mathcal{N}}\{|q_i|\} = 10$

	$\lvert V \rvert = 2$		$\lvert V \rvert = 3$	
$\lvert N \rvert$	Distance	CPU time (s)	Distance	CPU time (s)
9	21	0.11	-	-
14	21	0.54	20	2.27
16	53	0.84	51	9.57
22	23	10.62	23	29.59
29	8 620	144.705	8 846	347.38

Current and Future Work

- Design new network-based neighborhoods able to improve solution quality.
- Design a real-world instance for the RP using data provided by Encicla program.
- Design exact and heuristic strategies able to include synchronization features in several routes.

Towards a general framework for the Repositioning Problem in Bicycle-sharing Systems

Juan David Palacio Domínguez

jpalac26@eafit.edu.co

Juan Carlos Rivera Agudelo

jrivera6@eafit.edu.co