Hybrid approaches to 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

June 22, 2018

Outline

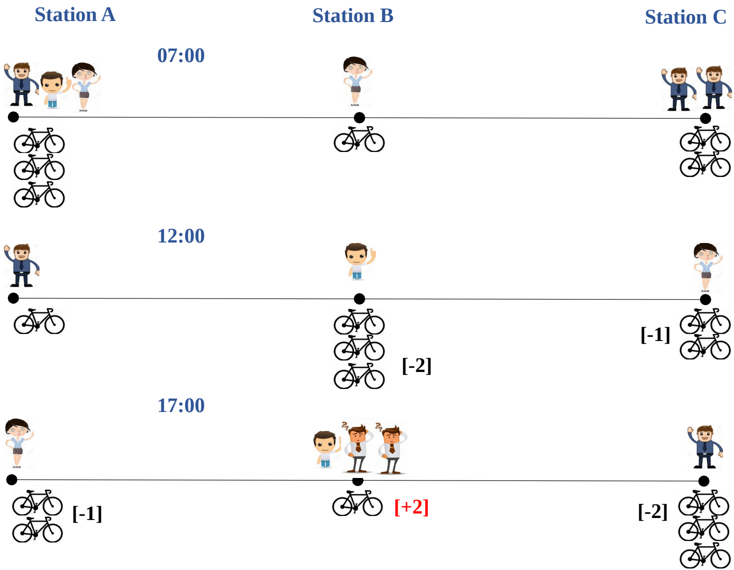
1. The Repositioning Problem (RP) – Description
2. Solution strategies
3. Hybrid algorithms in combinatorial optimization
4. Preliminary results
5. Current and future work


E 53 St & Madison Av
Midtown
Unlock
NYC.
citi bike

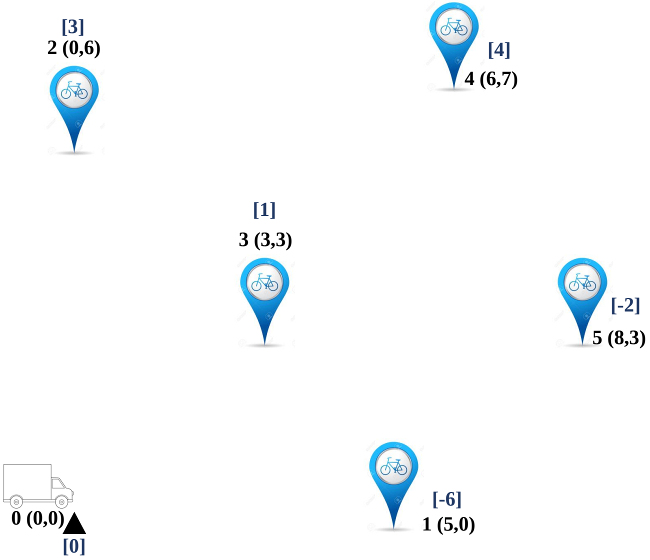

bicing


encicla

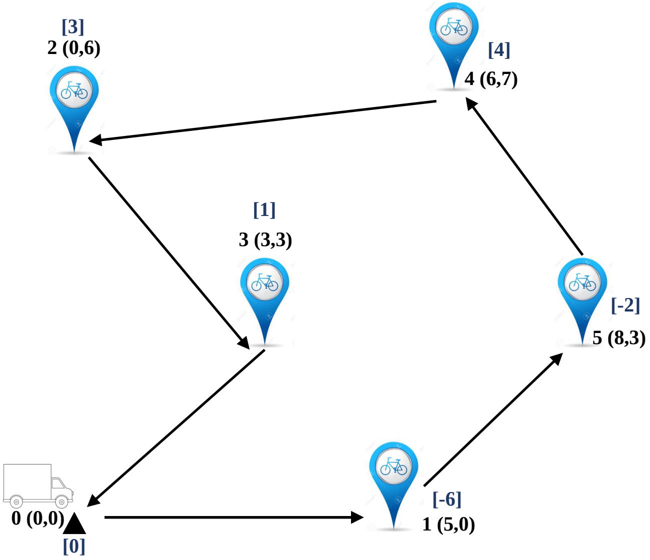
Balancing a BSS



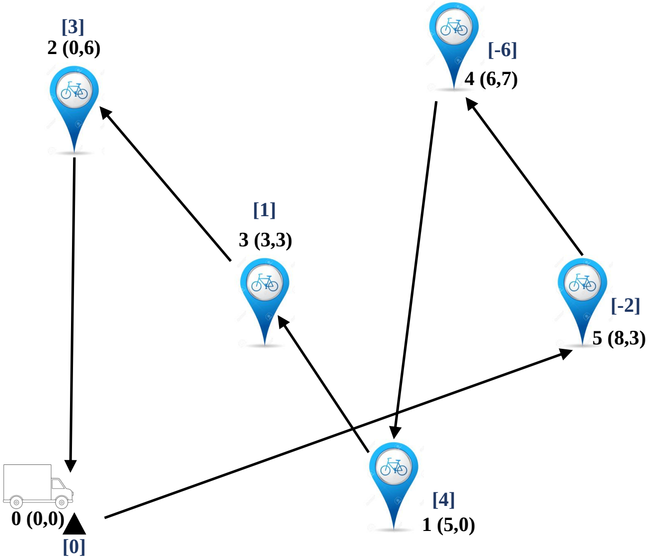
Pick up and Delivery TSP



Pick up and Delivery TSP



Pick up and Delivery TSP



Solution strategies – Single vehicle case

- Mixed Integer Programming Models (MILPs)
  - 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)
- Hybrid approaches (matheuristics)
  - MILP based local search operators
  - MILP based post-optimization procedures
  - PDTPSP decomposition

A GRASP algorithm

Algorithm 1 GRASP

$f^* \leftarrow \infty$
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^*

Our GRASP algorithm

Algorithm 2 GRASP + VND

$f^* \leftarrow \infty$
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
return S^*

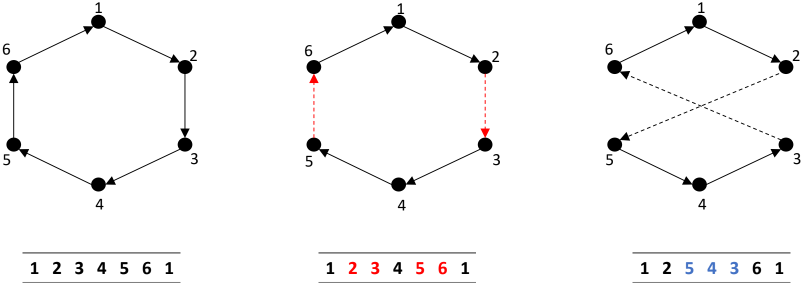
VND Structure

Six neighborhoods within a VND method

- Destroy and Repair
- Lin & Kernighan algorithm
- Or-opt(λ), $\lambda = 2, 3$
- Forward insertion
- Backward insertion

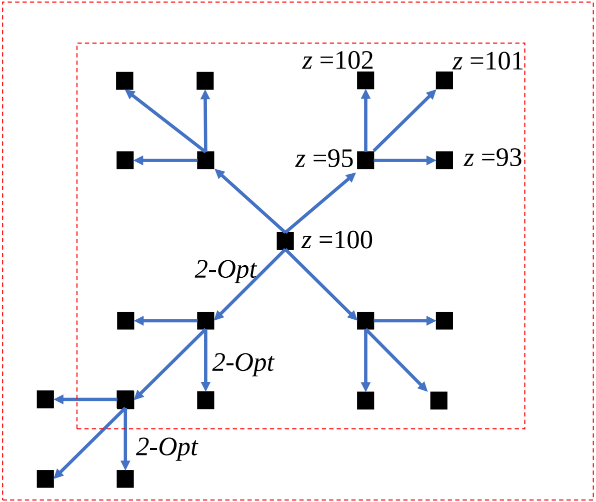
Lin & Kernighan algorithm

The Lin & Kernighan algorithm is based on 2-opt operator



Lin & Kernighan algorithm

Being z the total distance for a PDTSP solution:



Or-opt(λ), $\lambda = 2, 3$

Move forward or backward λ nodes (some movements are infeasibles):

Path	**1**	**2**	**3**	**4**	**5**	**6**	**1**
Demand	0	-5	2	-4	3	4	0

Or -opt (2)

Path	**1**	**2**	**3**	**4**	**5**	**6**	**1**
Demand	0	-5	2	-4	3	4	0
Path	**1**	**4**	**5**	**2**	**3**	**6**	**1**
Demand	0	-4	3	-5	2	4	0

Or -opt (3)

Path	**1**	**2**	**3**	**4**	**5**	**6**	**1**
Demand	0	-5	2	-4	3	4	0
Path	**1**	**4**	**5**	**6**	**2**	**3**	**1**
Demand	0	-4	3	4	-5	2	0

Hybrid algorithms in combinatorial optimization

Exact approaches

Strategies able to provide the optimal solution for an optimization problem. E.g.: Linear Programming, Mixed (and) Integer Programming, Dynamic Programming.

(Meta)Heuristic algorithms

Strategies able to provide good (*near to optimal*) solutions for an optimization problem in a decent computational time. E.g.: Greddy Randomized Adaptive Search Procedure (GRASP), Local Search (LS).

Hybrid algorithms (matheuristics)

Exact approaches + (Meta)Heuristic algorithms

A GRASP algorithm

Algorithm 3 GRASP

$f^* \leftarrow \infty$;
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^*

Algorithm 4 A first GRASP based matheuristic

```
f* ← ∞;
for i = 1 to GRASPIterations do
    S ←GreedyRandomAlgorithm();
    S ←SolveMILP(S);  //A MILP as intesification procedure
    if f(S) < f* then
        S* ← S;
        f* ← f(S);
    end if
end for
return  S*
```

A GRASP based matheuristic – An example

Algorithm 5 A second GRASP based matheuristic

$f^* \leftarrow \infty$;
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
$S^* \leftarrow$SolveMILP(S^*) // A MILP as post-optimization procedure
return S^*

Selective TSP-based matheuristic

Algorithm 6 Selective TSP-based matheuristic

$\mathcal{A} \leftarrow \emptyset$
while $|\mathcal{A}| < |\mathcal{N}|$ **do**
    $\mathcal{A} \leftarrow$ SolveSelectiveMILP($\mathcal{A}$)
end while
$S^* \leftarrow$BuildPath($\mathcal{A}$)
return S^*

MILP for the Selective TSP matheuristic (SolveSelectiveMILP)

- Sets
  - $\mathcal{N}$: Set of stations (including the depot)
  - $\mathcal{N}_s$: Set of stations (excluding the depot)
  - $\mathcal{A}$: Set of fixed arcs from previous solutions
- Parameters
  - c_{ij} : Cost of traveling from station i to station j
  - q_i : Demand or slack of bicycles in station i
  - Q : Vehicle capacity
- Decision Variables
  - $y_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$
  - x_{ij} : Vehicle load when traveling from i to j
  - z_{ij} : Position of arc (i,j) in the solution

MILP for the Selective TSP matheuristic (`SolveSelectiveMILP`)

$$\min f = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} c_{ij} \cdot y_{ij} \tag{1}$$

subject to,

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} y_{ij} \geq |\mathcal{A}| + 1 \tag{2}$$

$$\sum_{k \in \mathcal{N}} z_{ik} - \sum_{k \in \mathcal{N}} z_{jk} \geq 1 \qquad \forall\ (i,j) \in \mathcal{A} \tag{3}$$

$$\sum_{j \in \mathcal{N},\ i \neq j} y_{ij} \leq 1 \qquad \forall\ i \in \mathcal{N} \tag{4}$$

$$\sum_{i \in \mathcal{N}} y_{ij} = \sum_{i \in \mathcal{N}} y_{ji} \qquad \forall\ j \in \mathcal{N} \tag{5}$$

MILP for the Selective TSP matheuristic (`SolveSelectiveMILP`)

$$x_{ij} \leq Q \cdot y_{ij} \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{6}$$

$$\sum_{j \in \mathcal{N}} x_{ji} - \sum_{j \in \mathcal{N}} x_{ij} = q_i \cdot \sum_{j \in \mathcal{N}} y_{ij} \qquad \forall\ i \in \mathcal{N} \tag{7}$$

$$\sum_{j \in \mathcal{N}} z_{ji} - \sum_{j \in \mathcal{N}} z_{ij} = \sum_{j \in \mathcal{N}} y_{ij} \qquad \forall\ i \in \mathcal{N}_s \tag{8}$$

$$z_{ij} \leq |\mathcal{N}| \cdot y_{ij} \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{9}$$

$$y_{ij} \in \{0, 1\} \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{10}$$

$$z_{ij} \in \mathcal{Z}^+ \cup \{0\} \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{11}$$

$$x_{ij} \geq 0 \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{12}$$

Selective TSP + VND - based matheuristic

Algorithm 7 Selective TSP + VND – based matheuristic

$\mathcal{A} \leftarrow \emptyset$
while $|\mathcal{A}| < |\mathcal{N}|$ **do**
    $\mathcal{A} \leftarrow$ SolveSelectiveMIP($\mathcal{A}$)
end while
$S^* \leftarrow$BuildPath($\mathcal{A}$)
$S^* \leftarrow$VND(S^*)
return S^*

GRASP + MILP post-optimization procedure

Algorithm 8 GRASP + MILP post-optimization procedure

$f^* \leftarrow \infty$, $\xi \leftarrow \emptyset$
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
$S^* \leftarrow$Improve(ξ);
return S^*;

Improve elite solution algorithm

Algorithm 9 Improve elite solutions algorithm

1: **for** $i = 1$ to $|\xi|$ **do**
2: $\mathcal{A} \leftarrow$Destroy($\xi_{[i]}$)
3: $S \leftarrow$RepairMILP($\mathcal{A}$)
4: **if** $f(S) < f^*$ **then**
5: $S^* \leftarrow S$;
6: $f^* \leftarrow f(S)$;
7: **end if**
8: **end for**
9: **return** S^*;

MILP for the repairing function (`RepairMILP`)

$$\min f = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} c_{ij} \cdot y_{ij} \tag{13}$$

subject to,

$$\sum_{k \in \mathcal{N}} z_{ik} - \sum_{k \in \mathcal{N}} z_{jk} \geq 1 \qquad \forall\ (i,j) \in \mathcal{A} \tag{14}$$

$$\sum_{j \in \mathcal{N},\ i \neq j} y_{ij} = 1 \qquad \forall\ i \in \mathcal{N} \tag{15}$$

$$\sum_{i \in \mathcal{N}} y_{ij} = \sum_{i \in \mathcal{N}} y_{ji} \qquad \forall\ j \in \mathcal{N} \tag{16}$$

$$x_{ij} \leq Q \cdot y_{ij} \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{17}$$

MILP for the repairing function (`RepairMILP`)

$$\sum_{j \in \mathcal{N}} x_{ji} - \sum_{j \in \mathcal{N}} x_{ij} = q_i \qquad \forall\ i \in \mathcal{N} \tag{18}$$

$$\sum_{j \in \mathcal{N}} z_{ji} - \sum_{j \in \mathcal{N}} z_{ij} = 1 \qquad \forall\ i \in \mathcal{N}_s \tag{19}$$

$$z_{ij} \leq |\mathcal{N}| \cdot y_{ij} \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{20}$$

$$y_{ij} \in \{0, 1\} \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{21}$$

$$z_{ij} \in \mathcal{Z}^+ \cup \{0\} \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{22}$$

$$x_{ij} \geq 0 \qquad \forall\ i \in \mathcal{N}, j \in \mathcal{N} \tag{23}$$

- 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 – GRASP + VND

Table: GRASP + VND results with GRASPIterations=200 and 10 runs

		GRASP + VND				
Number stations	Optimal solution	Best solution	Best GAP	Average solution	Average GAP	Average time (s)
9	38.15	38.15	0.00%	38.15	0.00%	0.03
14	36.01	36.01	0.00%	36.01	0.00%	0.12
16	84.84	84.84	0.00%	84.84	0.00%	0.12
22	95.84	95.84	0.00%	95.84	0.00%	0.23
29	13529.2	13529.2	0.00%	13563.57	0.25%	0.46
42	1446.33	1446.33	0.00%	1451.48	0.36%	1.24

Preliminary results – Selective TSP matheuristic

Table: Selective TSP + VND matheuristic results

		Selective TSP + VND matheuristic					
Number stations	Optimal solution	Iterative phase (Selective TSP)			Selective TSP + VND		
		Solution	GAP	Time (s)	Solution	GAP	Time (s)
9	38.15	40.65	6.55%	1.10	38.15	0.00%	1.11
14	36.01	36.74	2.03%	10.69	36.01	0.00%	10.70
16	84.84	85.50	0.78%	31.84	84.84	0.00%	31.87
22	95.84	116.69	21.76%	260.33	106.08	10.68%	260.35
29	13529.2	17210.6	27.21%	206.71	16108.1	19.06%	207.24
42	1446.33	1855.9	29.32%	442.92	1579.86	9.23%	442.97

current and future work

- Improve the performance of the matheuristic algorithms by designing new decomposition strategies.
- Solve larger instances of the PDTSP via GRASP + VND and compare our results with reported benchmarks.
- Extend our solutions algorithms to the multiple vehicle clase.

Hybrid approaches to the Repositioning Problem in Bicycle-Sharing Systems

Juan David Palacio Domínguez

jpalac26@eafit.edu.co

Juan Carlos Rivera Agudelo

jrivera6@eafit.edu.co