

Dealing with Feasibility and Solution Quality in a Rebalancing Vehicle Routing Problem

Juan David Palacio Domínguez

PhD Student in Mathematical Engineering

Juan Carlos Rivera Agudelo

Thesis Advisor

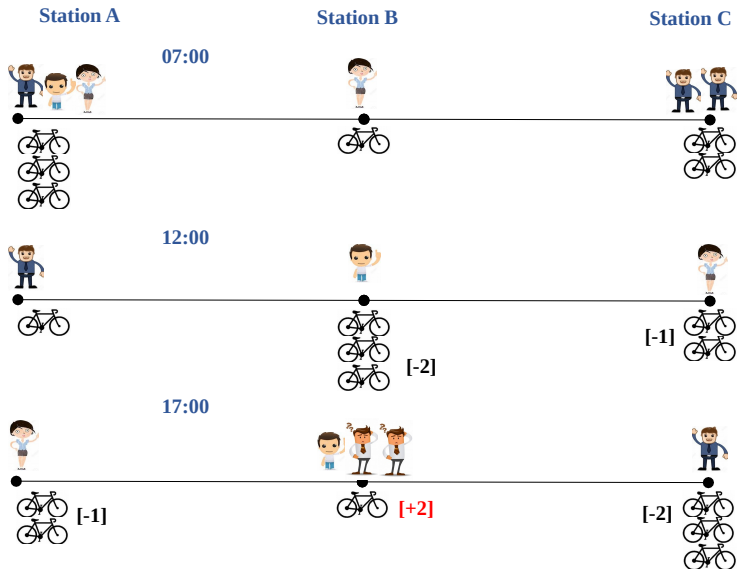
Doctoral Seminar in Mathematical Engineering



November 2, 2018

- 1 The Repositioning Vehicle Routing Problem (RP) – Description
- 2 Solution Strategies
- 3 Metaheuristics and Mathematical Formulations
- 4 Preliminary Results
- 5 Current and Future Work

Balancing a BSS



Pick up and Delivery TSP

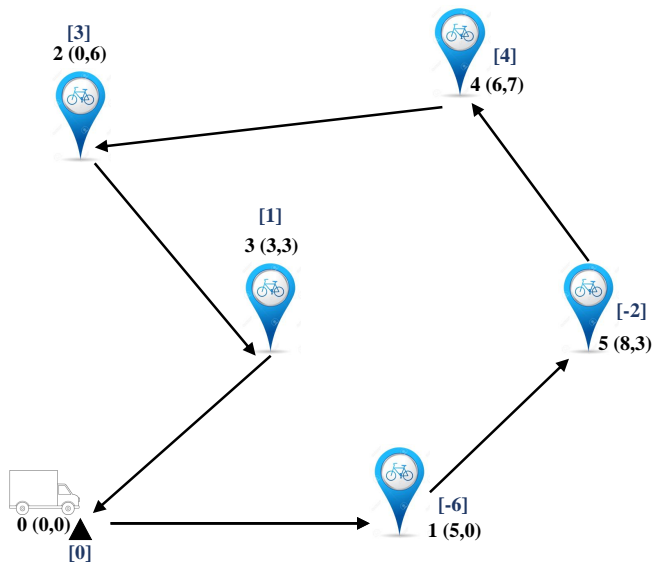
[3]
2 (0,6)



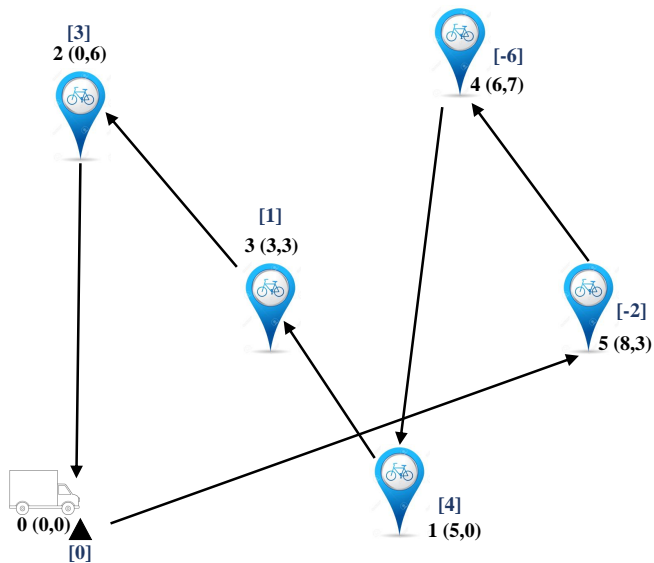
[1]
3 (3,3)



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

Algorithm 1 GRASP

```
1:  $f^* \leftarrow \infty$ 
2: for  $i = 1$  to  $GRASPIterations$  do
3:    $S \leftarrow GreedyRandomAlgorithm()$ 
4:    $S \leftarrow LocalSearch(S)$ 
5:   if  $f(S) < f^*$  then
6:      $S^* \leftarrow S$ 
7:      $f^* \leftarrow f(S)$ 
8:   end if
9: end for
10: return  $S^*$ 
```

Algorithm 2 GRASP + VND

```
1:  $f^* \leftarrow \infty$ 
2: for  $i = 1$  to  $GRASPIterations$  do
3:    $S \leftarrow GreedyRandomAlgorithm()$ 
4:    $S \leftarrow VND(S)$ 
5:   if  $f(S) < f^*$  then
6:      $S^* \leftarrow S$ 
7:      $f^* \leftarrow f(S)$ 
8:   end if
9: end for
10: return  $S^*$ 
```

Seven neighborhoods within a VND method

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

Algorithm 3 GRASP + VND

```
1:  $f^* \leftarrow \infty$ 
2: for  $i = 1$  to  $GRASPIterations$  do
3:    $S \leftarrow GreedyRandomAlgorithm()$ 
4:    $S \leftarrow VND(S)$ 
5:   if  $f(S) < f^*$  then
6:      $S^* \leftarrow S$ 
7:      $f^* \leftarrow f(S)$ 
8:   end if
9: end for
10: return  $S^*$ 
```

Algorithm 4 Greedy Random Algorithm

- 1: $S \leftarrow \emptyset$
 - 2: Evaluate the incremental costs of the candidate elements
 - 3: **while** S is not a complete solution **do**
 - 4: Build the restricted candidate list (RCL)
 - 5: Select an element s from the RCL at random
 - 6: $S \leftarrow S \cup \{s\}$
 - 7: Reevaluate the incremental costs
 - 8: **end while**
 - 9: **return** S
-

Our GRASP Algorithm

Example of an unfeasible constructive solution

- Vehicle capacity (Q)=10

Station	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Demand	4	6	-3	-3	-3	4	3	-7	0	9	-7	10	-6	-7	
Load	10	6	0	3	6	9	5	2	9	9	0	7	-3	3	10

Destroy and Repair Approach

Example of an infeasible constructive solution

- Vehicle capacity (Q)=10

Station	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Demand	4	6	-3	-3	-3	4	3	-7	0	9	-7	10	-6	-7	
Load	10	6	0	3	6	9	5	2	9	9	0	7	-3	3	10

- Set of fixed ordered stations : Stations from 1 to 11
- Set of stations to remove and insert in a different position: Stations from 12 to 14

Repairing an Unfeasible Solution – MILP

- Sets

- \mathcal{N} : Set of stations
- \mathcal{N}_s : Set of fixed ordered stations
- \mathcal{N}_r : Set of stations to insert
- \mathcal{A} : Set of arcs

- 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

- s : Surplus of bikes in the solution
- $y_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$
- $x_{ij} = \begin{cases} 1 & \text{if station } i \text{ is visited anytime before station } j \\ 0 & \text{otherwise} \end{cases}$
- l_i : Load of the vehicle after visiting i^{th} station
- l^+, l^- : Maximum and minimum load in the solution

Repairing an Unfeasible Solution – MILP

$$\min f = s \quad (1)$$

subject to,

$$x_{0i} = 1 \quad \forall i \in \mathcal{N} \setminus \{0\} \quad (2)$$

$$x_{ij} + x_{ji} = 1 \quad \forall (i, j) \in \mathcal{A} \quad (3)$$

$$y_{ij} \leq x_{ij} \quad \forall (i, j) \in \mathcal{A} \quad (4)$$

$$x_{ij} = 1 \quad \forall i \in \mathcal{N}_f, j \in \mathcal{N}_f, j > i \quad (5)$$

$$\sum_{j \in \mathcal{N}} y_{ij} = 1 \quad \forall i \in \mathcal{N} \quad (6)$$

$$y_{i,i+1} + \sum_{j \in \mathcal{N}_r} y_{ij} = 1 \quad \forall i \in \mathcal{N}_f \quad (7)$$

Repairing an Infeasible Solution – MILP

$$\sum_{j \in \mathcal{N}} y_{ij} - \sum_{j \in \mathcal{N}} y_{ji} = 0 \quad \forall i \in \mathcal{N} \quad (8)$$

$$l_j \geq l_i - q_j - 2Q(1 - y_{ij}) \quad \forall (i, j) \in \mathcal{A} \quad (9)$$

$$l^+ \geq l_i \quad \forall i \in \mathcal{N} \quad (10)$$

$$l^- \leq l_i \quad \forall i \in \mathcal{N} \quad (11)$$

$$l^+ - l^- = Q + s \quad (12)$$

$$y_{ij} \in \{0, 1\} \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \quad (13)$$


$$x_{ij} \in \{0, 1\} \quad \forall i \in \mathcal{N}, j \in \mathcal{N} \quad (14)$$

$$l_i \geq 0 \quad \forall i \in \mathcal{N} \quad (15)$$

$$l^+, l^- \geq 0 \quad (16)$$

Speeding up MILP Performance

- Time limit
- Objective bound to stop optimization

**GUROBI**
OPTIMIZATION

PRODUCTS DOWNLOADS RESOURCES ACADEMIA SUPPORT ABOUT

Enter search term

Login Register

Home > Documentation >


First, set the following:


Version:


Languages:


C C++
 Java .NET
 Python MATLAB
 R

Then, choose below:

[Quick Start Guides](#) 

[Example Tour](#) 

[Reference Manual](#) 

[AMPL-Gurobi Guide](#) 

Next: [BestObjStop](#) Up: [Parameter Descriptions](#) Previous: [BarterLimit](#)

BestBdStop

Objective bound to stop optimization

Type:	double
Default value:	Infinity
Minimum value:	-Infinity
Maximum value:	Infinity

Terminates as soon as the engine determines that the best bound on the objective value is at least as good as the specified value. Optimization returns with an `USER_OBJ_LIMIT` status in this case.

Note that you should always include a small tolerance in this value. Without this, a bound that satisfies the intended termination criterion may not actually lead to termination due to numerical round-off in the bound.

Note: Only affects mixed integer programming (MIP) models

For examples of how to query or modify parameter values from our different APIs, refer to our [Parameter Examples](#).

Algorithm 5 GRASP + VND

```
1:  $c \leftarrow \infty$ 
2:  $f^* \leftarrow \infty$ 
3: for  $i = 1$  to GRASPIterations do
4:    $S \leftarrow \text{GreedyRandomAlgorithm}(c)$ 
5:    $S \leftarrow \text{VND}(S)$ 
6:   if  $f(S) < f^*$  then
7:      $S^* \leftarrow S$ 
8:      $f^* \leftarrow f(S)$ 
9:   end if
10: end for
11: return  $S^*$ 
```

Algorithm 6 Greedy Random Algorithm

```
1:  $S \leftarrow \text{GreedyRandomAlgorithm}()$ 
2: while  $S$  is unfeasible do
3:   if  $f(S) < c$  then
4:     Solve RepairMILP( $S$ )
5:   else
6:      $S \leftarrow \text{GreedyRandomAlgorithm}()$ 
7:   end if
8: end while
9: if  $c > f(S)$  then
10:   $c = f(S)$ 
11: end if
12: return  $S^*, c$ 
```

Preliminary Results

Data sets and software

- Dataset
 - Instances taken from:
<https://hhperez.webs.u11.es/PDsite/index.html>
 - Instances with 20, 30, 50, 100 nodes were tested
- Software
 - All the algorithms were implemented on C++ (Visual Studio)
 - Mathematical models were solved using Gurobi Optimizer 7.5
- Computer features
 - Intel core i7, 8Gb RAM.
 - OS: Windows 10 (x86-64)

Preliminary Results – MILP-based Constructive Algorithm

\mathcal{N}	Original constructive algorithm				
	Best z	Load	Average z	Average Load	Average % Feasible Sols
20	7853	10	9811.36	12.24	37.00%
30	12387	10	14019.04	15.08	5.50%
50	18345	10	21346.69	12.66	28.00%
100	32200	10	35516.16	15.48	10.50%

\mathcal{N}	Repair constructive algorithm - MILP					
	Best z	Load	Average z	Average Load	Average Lower Bound	Average % Feasible Sols
20	7647	10	9969.73	10.21	0.18	85.00%
30	11396	10	14131.74	11.46	1.03	34.00%
50	17618	10	21591.77	10.56	0.35	69.00%
100	32203	10	35915.57	12.33	0.61	26.50%

Preliminary Results – Hybrid Constructive Algorithm

\mathcal{N}	Original constructive algorithm				
	Best		Average	Average	Average %
	Distance	Load	Distance	Load	Feasible Sols
20	7610	10	9848.63	12.05	36.50%
30	12122	10	14089.67	14.58	10.00%
50	18590	10	21526.07	13.06	27.50%
100	31743	10	35637.01	15.79	13.50%

\mathcal{N}	Hybrid constructive algorithm				
	Best		Average	Average	Average %
	Distance	Load	Distance	Load	Feasible Sols
20	7610	10	9863.77	10	100%
30	11707	10	13973.72	10	100%
50	17691	10	21438.71	10	100%
100	30889	10	35605.6	10	100%

- Improve the performance of destroy strategy finding a different criterion to remove stations.
- Solve larger instances of the PDTSP via GRASP+VND and compare our results with reported benchmarks.
- Extend our solutions algorithms to the multiple vehicle case.

Dealing with Feasibility and Solution Quality in a Rebalancing Vehicle Routing Problem

Juan David Palacio Domínguez

`jpalac26@eafit.edu.co`

Juan Carlos Rivera Agudelo

`jrivera6@eafit.edu.co`