Vehicle Routing Optimization in Bicycle-sharing Systems

Juan David Palacio Domínguez
PhD Student in Mathematical Engineering

Juan Carlos Rivera Agudelo
Thesis Advisor

May 22, 2017

Outline

My PhD Admission Proposal

Vehicle Routing Optimization Applied to Non-profit Sector

General Objective

To design models and efficient solution methods for a set of vehicle routing optimization problems that includes features, objectives and conditions typically faced by non-profit sectors.

My PhD Admission Proposal

Non-profit applications & problem features

Some applications:

- Bicycle-sharing systems
- Health care
- Humanitarian logistics
- Hazardous materials and waste management

Some problem features:

- Pick-up and deliveries
- Stochastic parameters
- Synchronization
- Multi-period approaches
- Consistency

Outline

BSS around the world

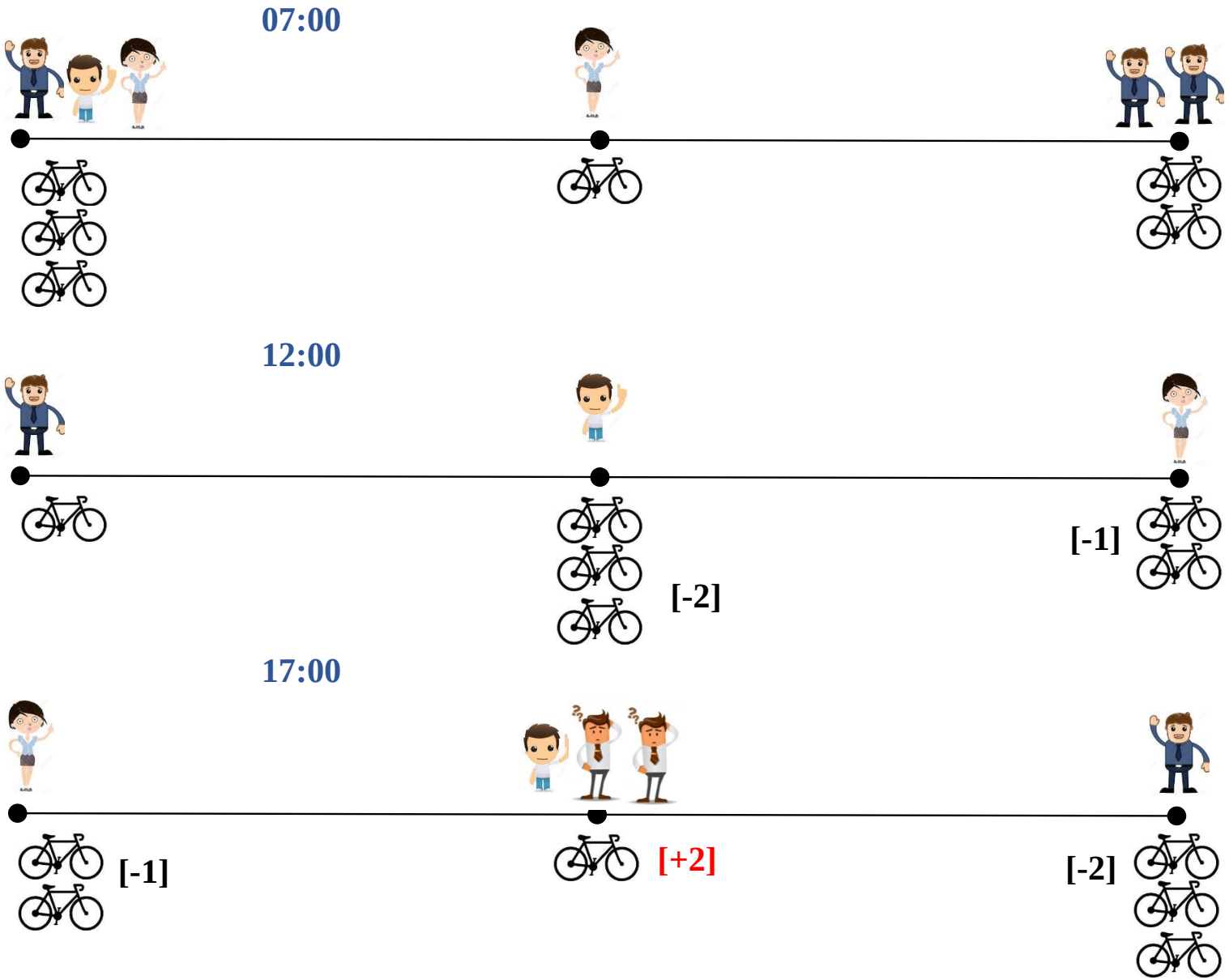
BSS in Medellín

Encicla

BSS in Medellín

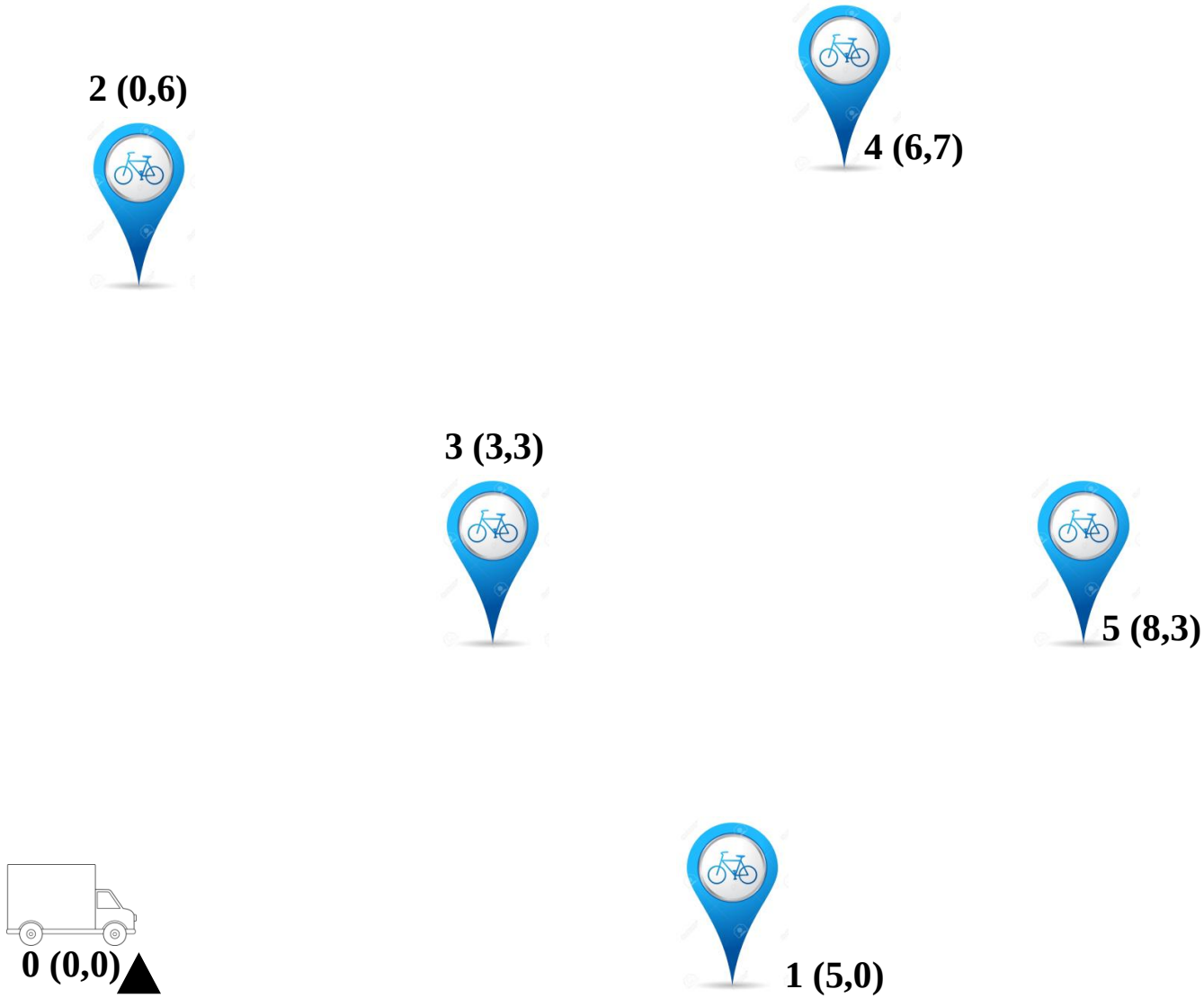
Encicla

Balancing a BSS

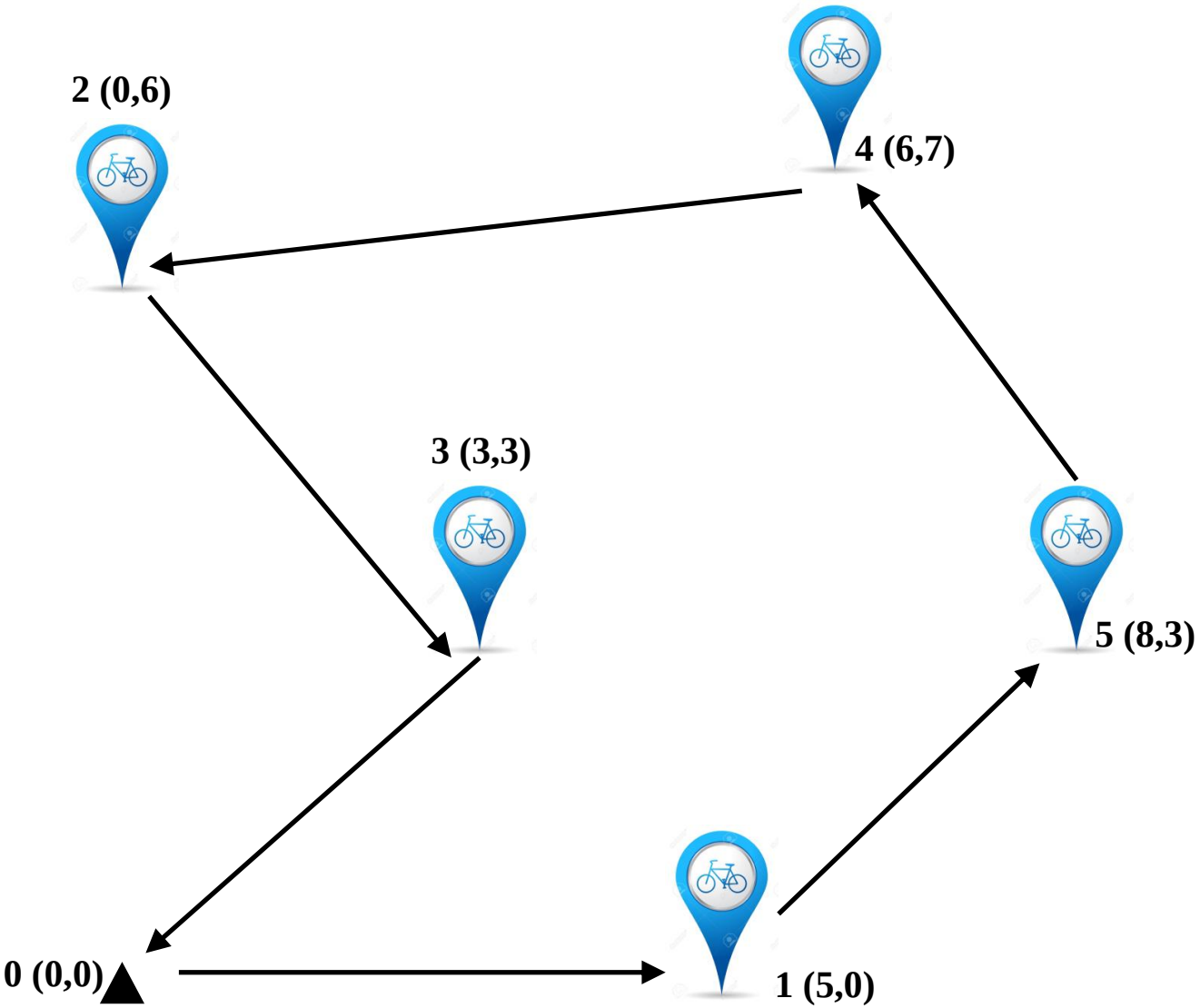


Outline

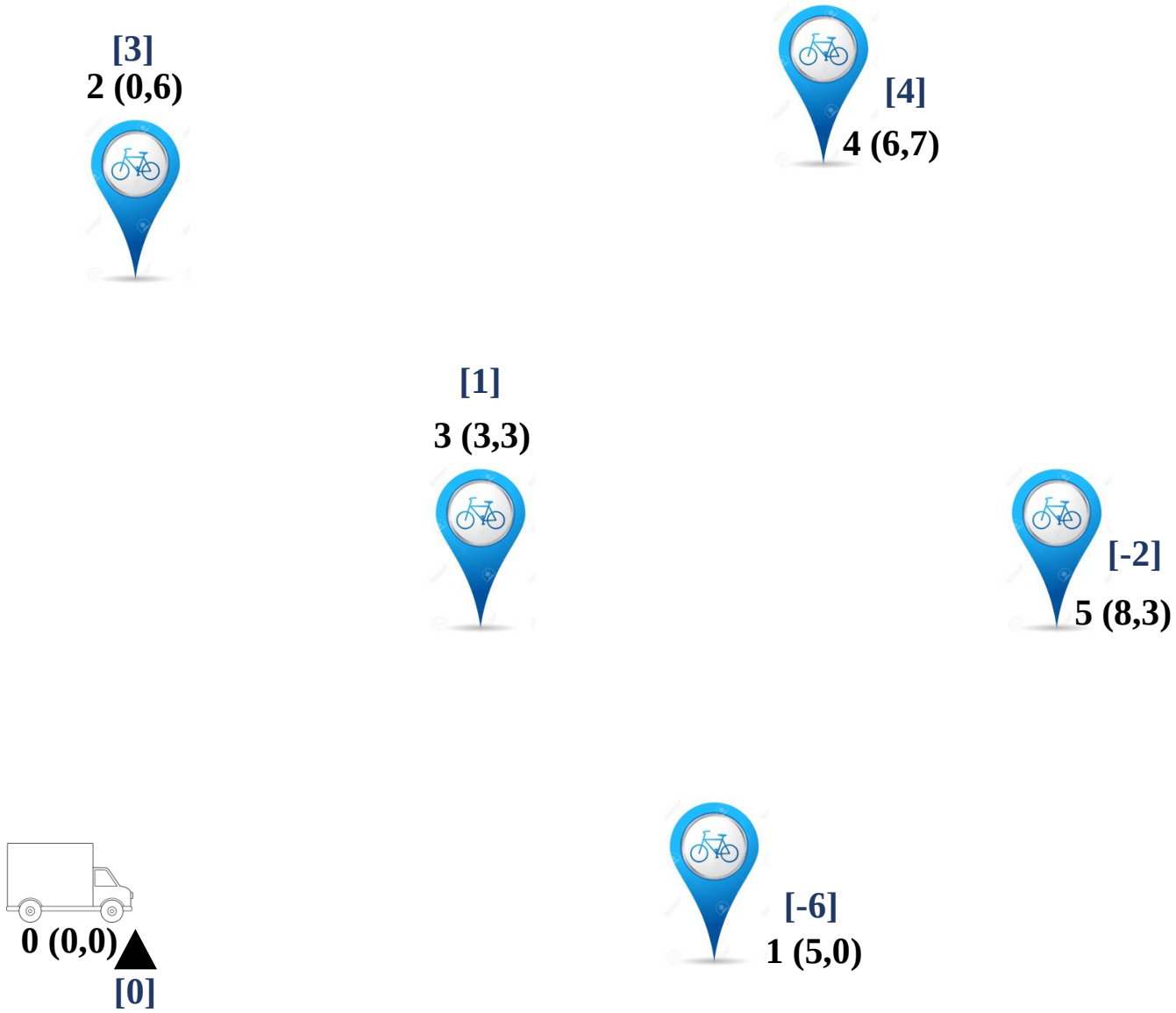
The Traveling Salesman Problem



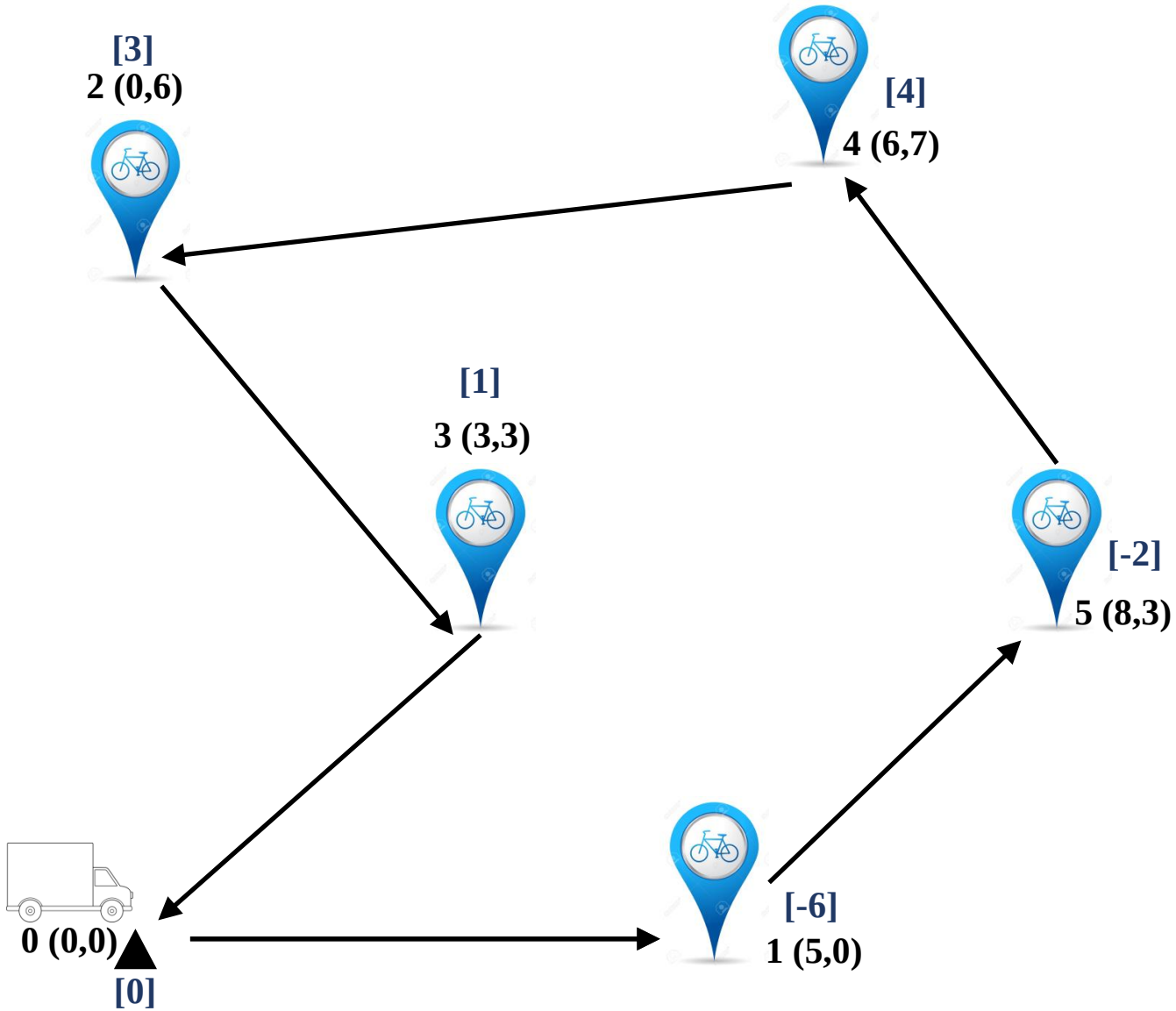
The Traveling Salesman Problem



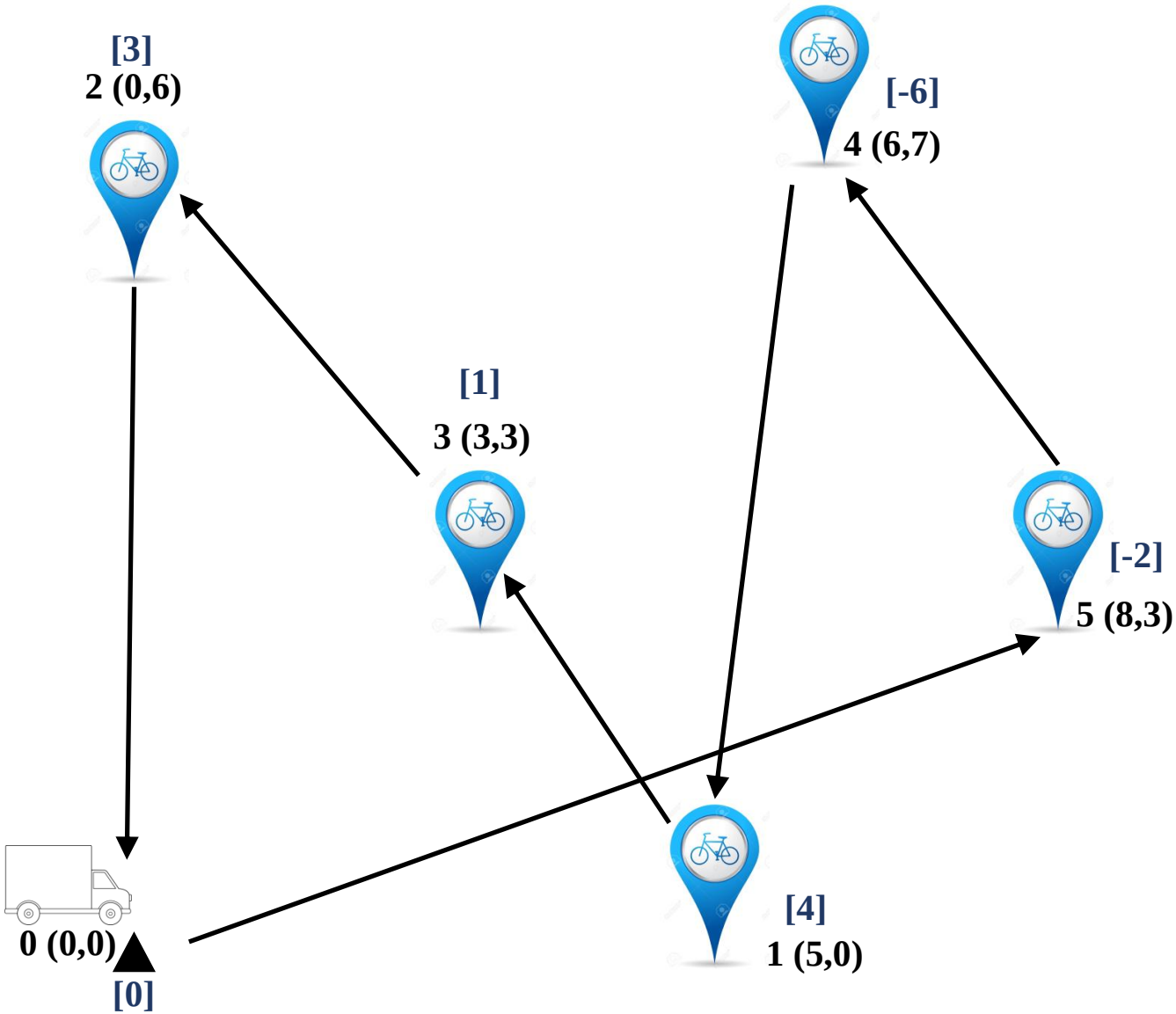
Pick up and Delivery TSP



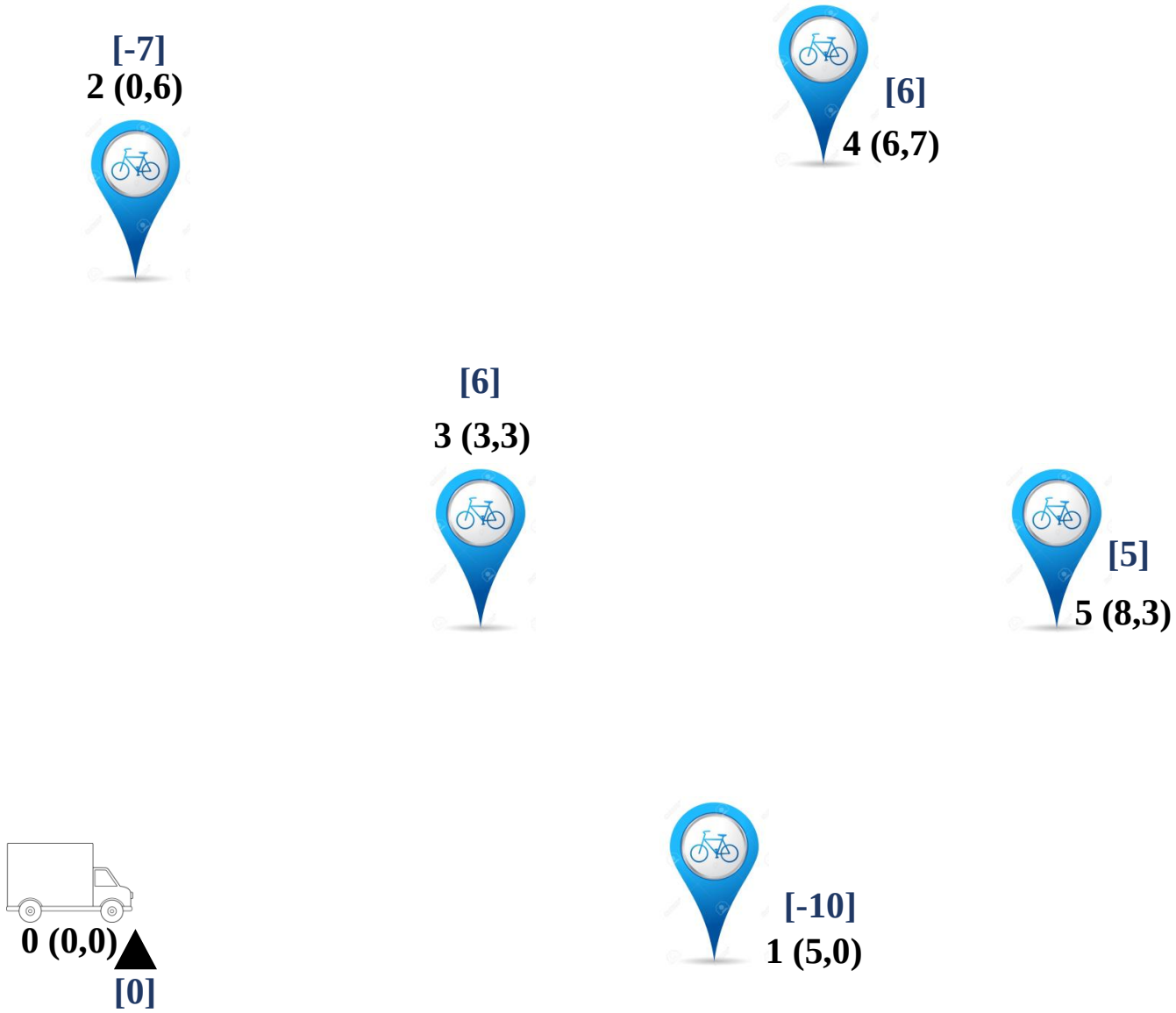
Pick up and Delivery TSP



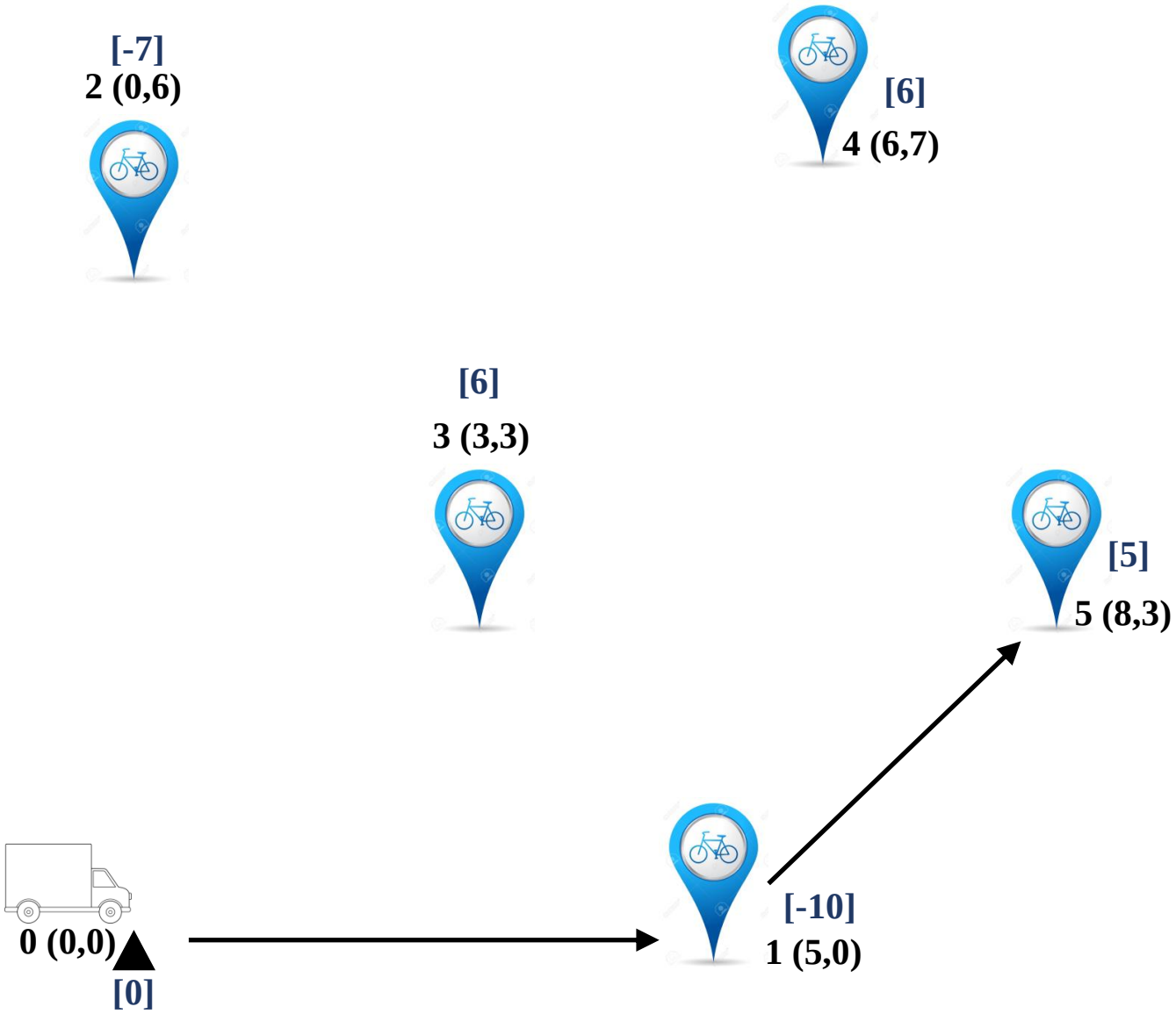
Pick up and Delivery TSP



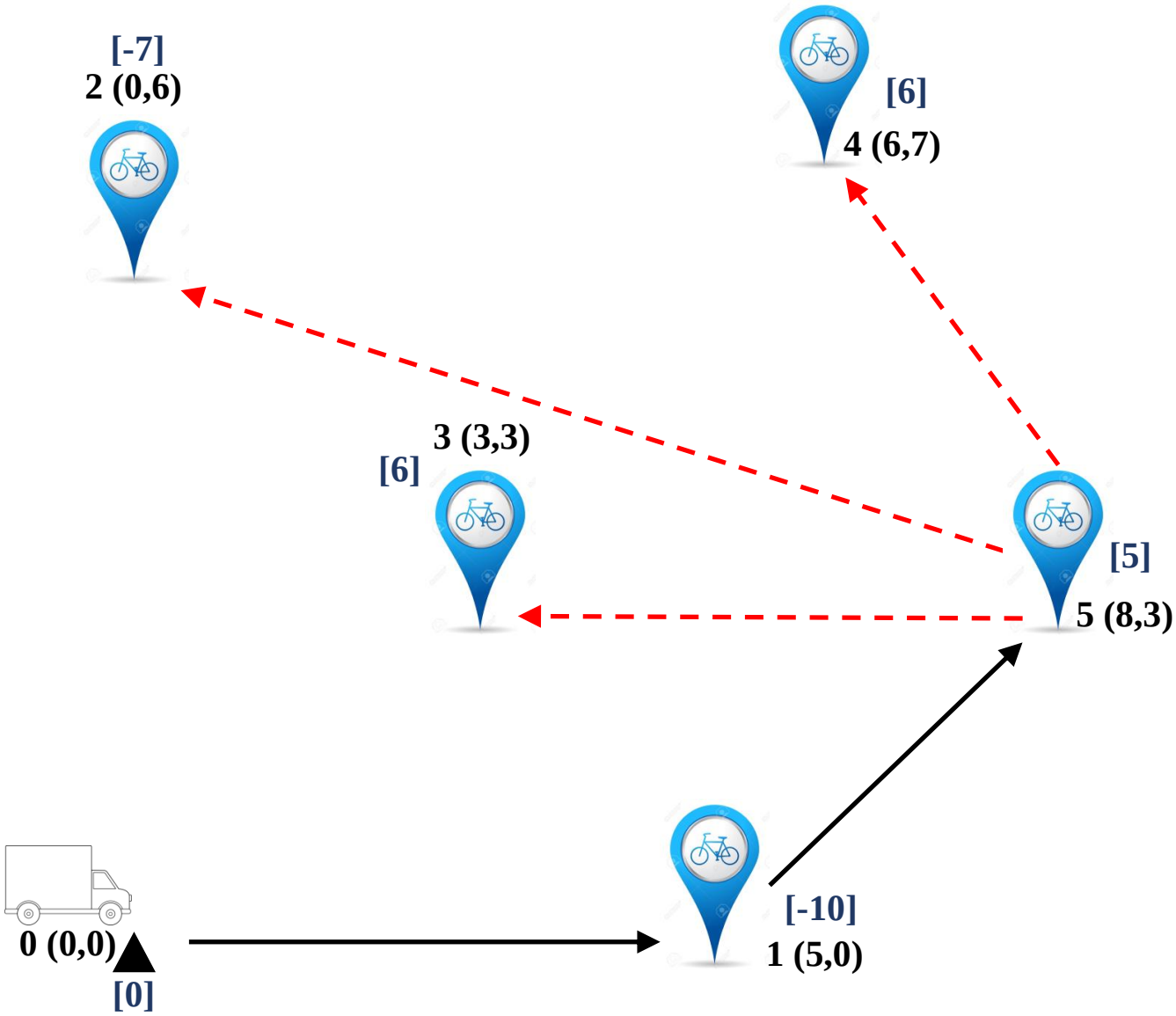
Pick up and Delivery TSP



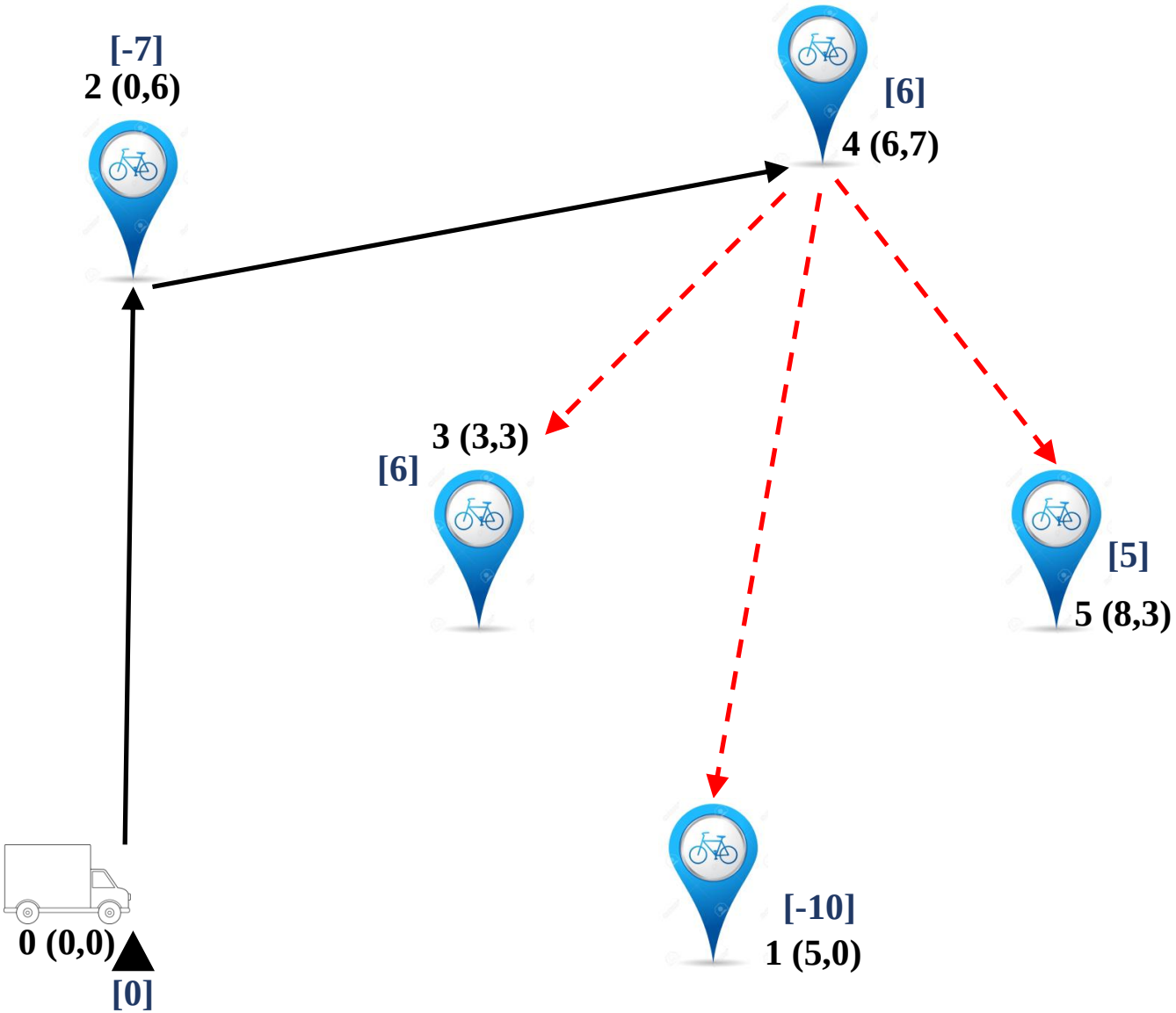
Pick up and Delivery TSP



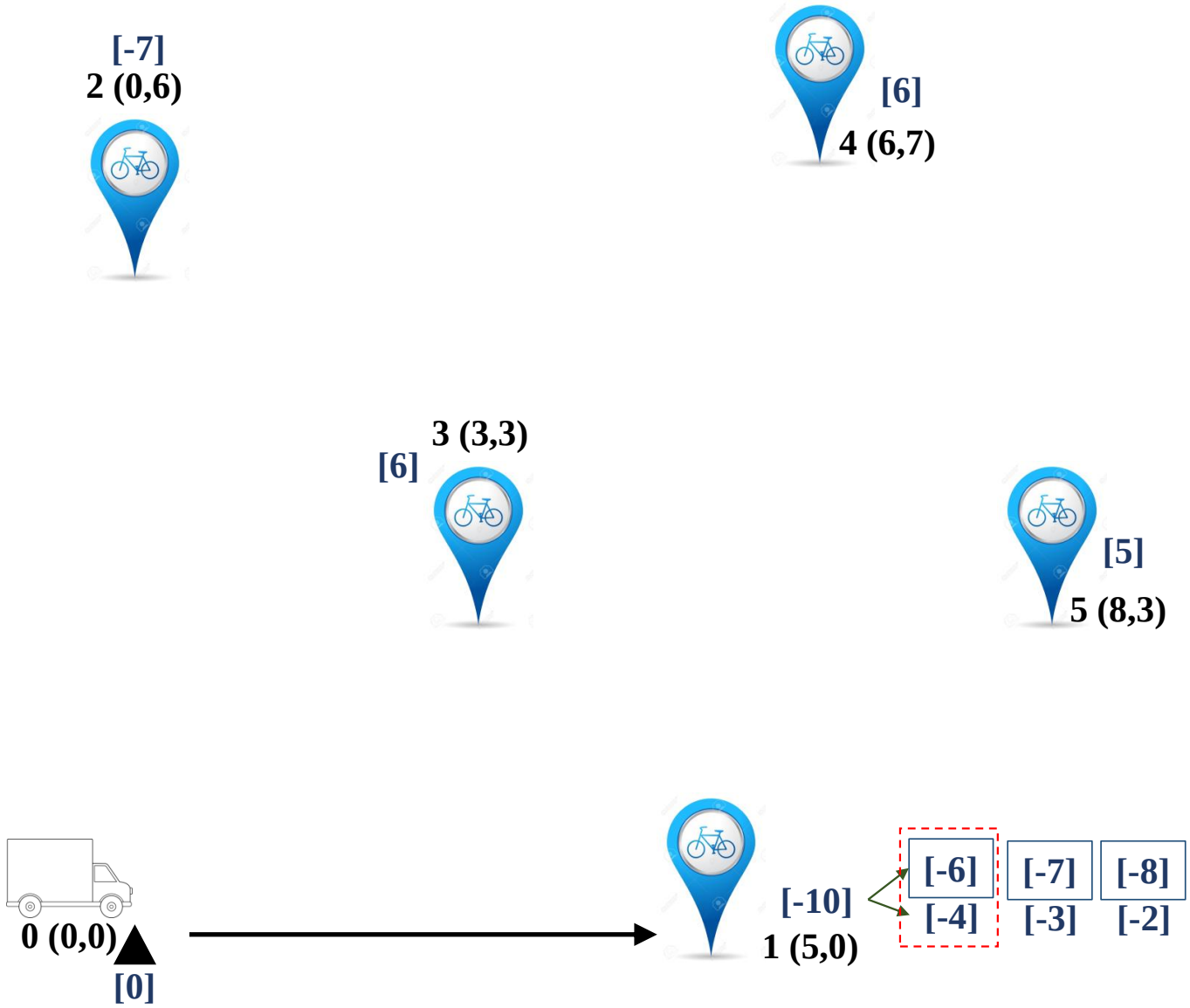
Pick up and Delivery TSP



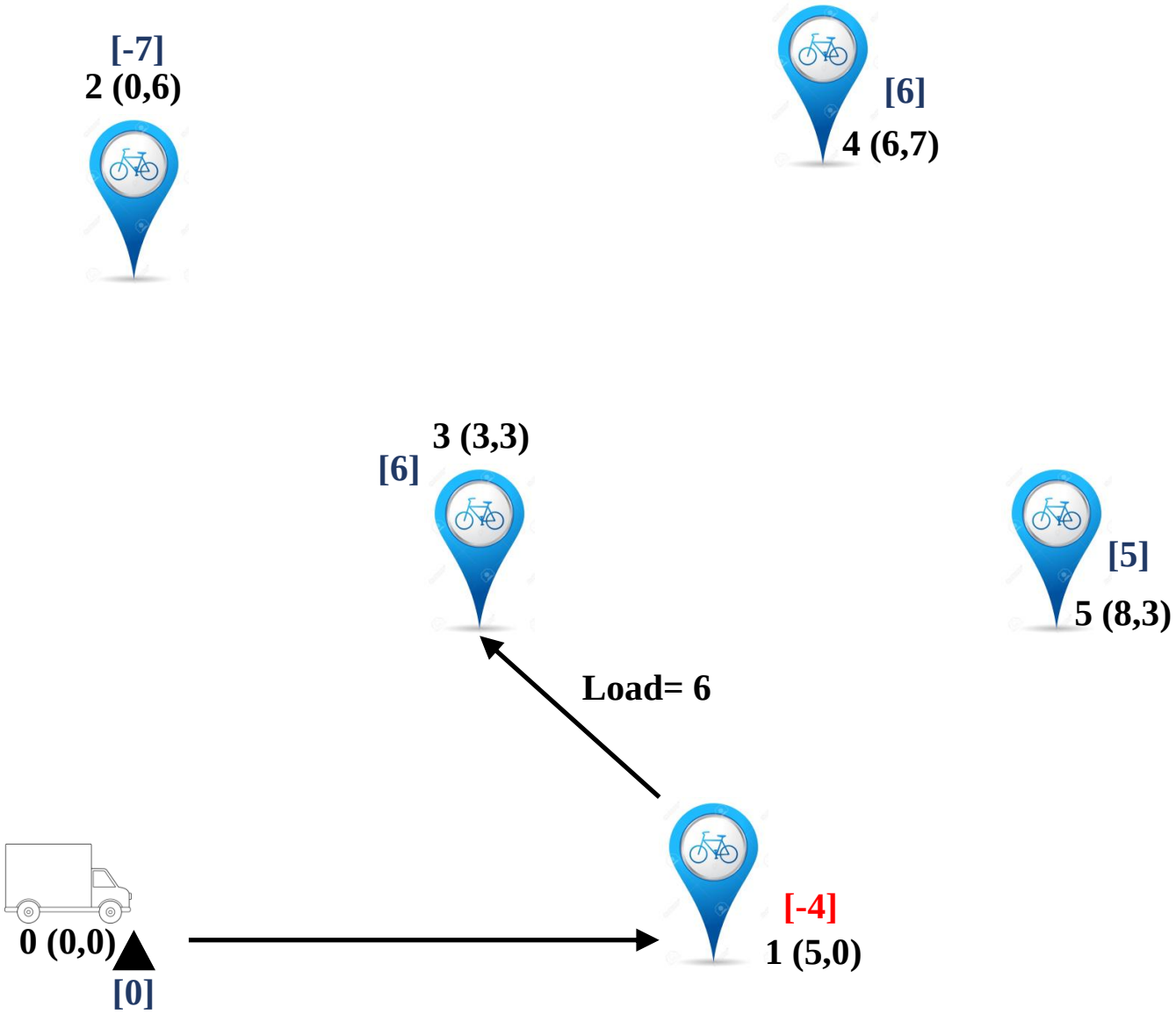
Pick up and Delivery TSP



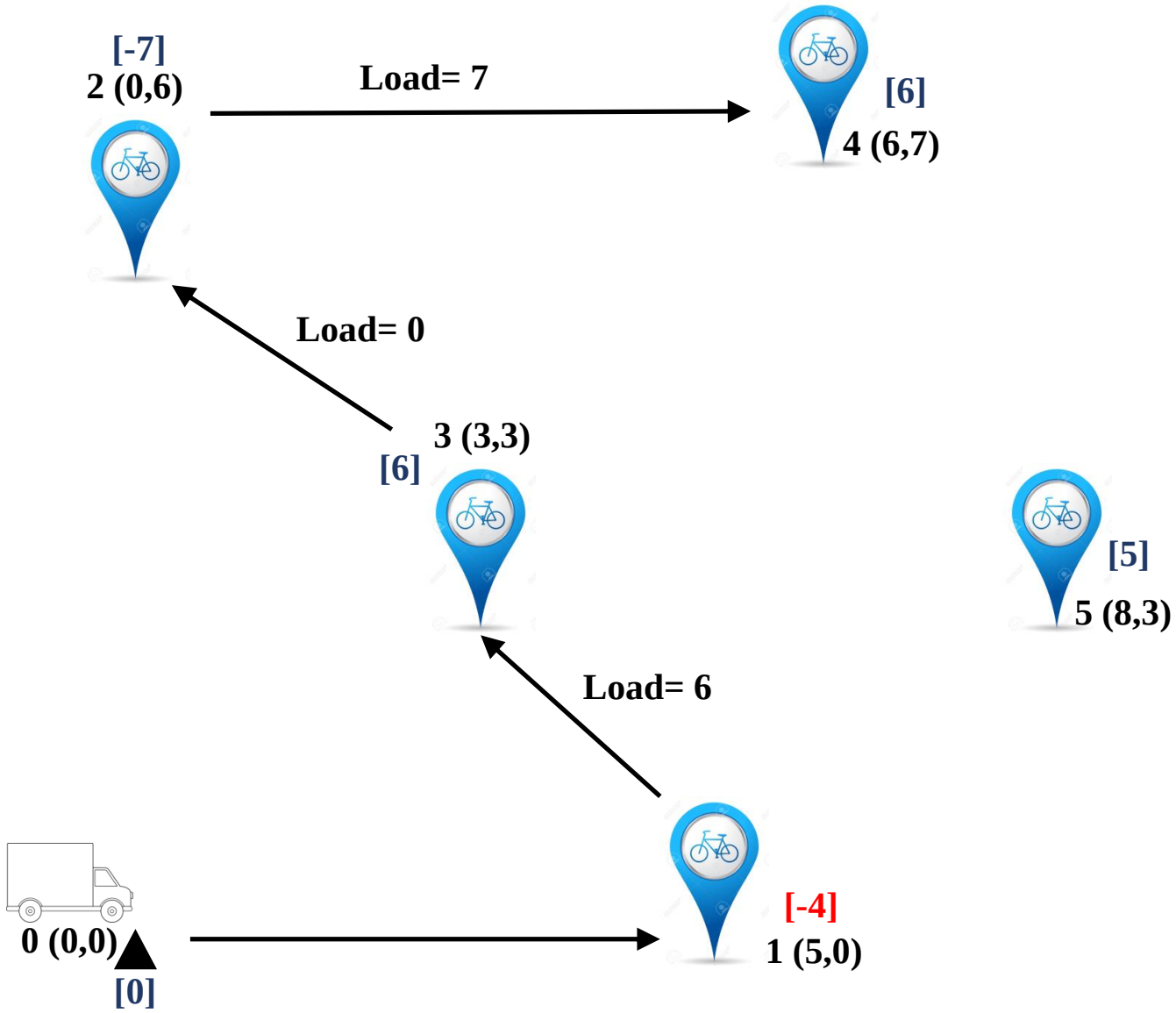
Pick up and Delivery TSP with Split Demand



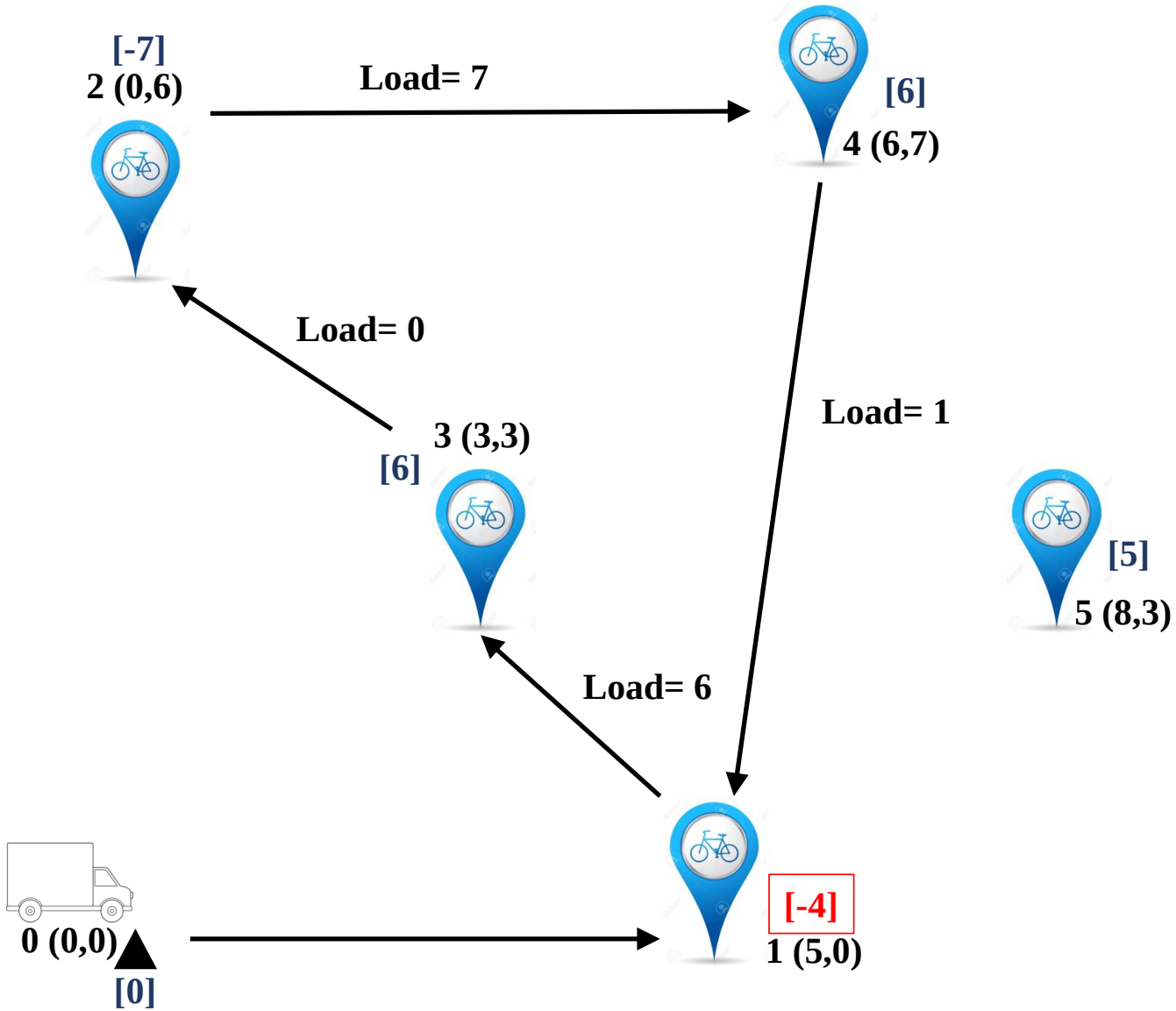
Pick up and Delivery TSP with Split Demand



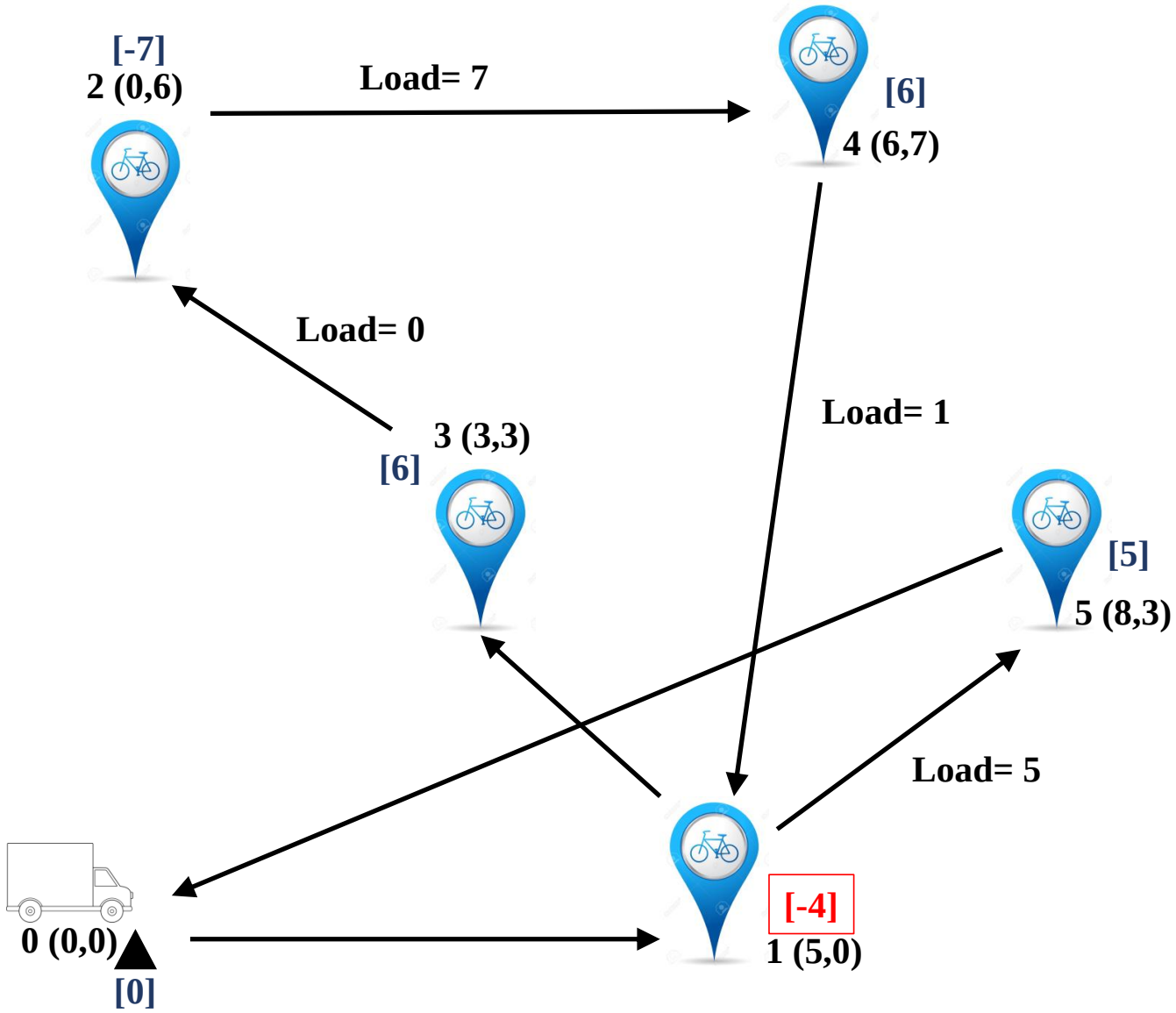
Pick up and Delivery TSP with Split Demand



Pick up and Delivery TSP with Split Demand



Pick up and Delivery TSP with Split Demand



Vehicle Routing Optimization in BSS - Literature Review

Author	Solution Strategy	Stations	PD	Split
Ho & Szeto (2017)	LNS + TS	518	✓	X
Schuijbroek et al. (2017)	CP, MIPs	30	✓	X
Dell'Amico et al. (2016)	DR + LS	564	✓	X
Szeto et al. (2016)	CRO	300	✓	X
Kadri et al. (2016)	BB	30	✓	X
Forma et al. (2015)	CB	200	✓	X
Brinkmann et al. (2015)	HC + SA	200	✓	X

LNS: Large Neighborhood Search, TS: Tabu Search,
CP: Constraint Programming, MIP: Mixed-Integer Programming,
DR: Destroy and Repair, LS: Local Search, BB: Branch and Bound,
CB: Cluster-based, HC: Hill Climbing, SA: Simulated Annealing,
CRO: Chemical Reaction Optimization

Mathematical Formulation for the TSP

- Sets

  $\mathcal{N}$: Set of stations

- Parameters

  c_{ij} : Cost of traveling from station i to station j

- Decision Variables
  - $y_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$
  - z_{ij} : Position of arc (i,j) in the solution

Mathematical Formulation for the TSP

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

subject to,

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

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

$$\sum_{k \in \mathcal{N}} z_{ki} - \sum_{j \in \mathcal{N}} z_{ij} = 1 \qquad \forall\ i \in \mathcal{N} \setminus \{0\}$$

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

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

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

Mathematical Formulation for the PDTSP

- Sets

  $\mathcal{N}$: Set of stations

- 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

Mathematical Formulation for the PDTSP

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

subject to,

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

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

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

$$\sum_{k \in \mathcal{N}} x_{ki} - \sum_{j \in \mathcal{N}} x_{ij} = q_i \qquad \forall\ i \in \mathcal{N}$$

Mathematical Formulation for the PDTSP

$$\sum_{k \in \mathcal{N}} z_{ki} - \sum_{j \in \mathcal{N}} z_{ij} = 1 \qquad \forall\ i \in \mathcal{N} \setminus \{0\}$$

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

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

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

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

Mathematical Formulation for the PDTSP with Split Demand

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

subject to,

$$\sum_{j \in \mathcal{N}, i \neq j} y_{ij} \geq \left\lceil \frac{|q_i|}{Q} \right\rceil \qquad \forall\ i \in \mathcal{N}$$

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

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

$$\sum_{k \in \mathcal{N}} x_{ki} - \sum_{j \in \mathcal{N}} x_{ij} = q_i \qquad \forall\ i \in \mathcal{N}$$

Mathematical Formulation for the PDTSP with Split Demand

$$\sum_{k \in \mathcal{N}} z_{ki} - \sum_{j \in \mathcal{N}} z_{ij} = 1 \qquad \forall\ i \in \mathcal{N} \setminus \{0\}$$

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

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

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

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

Solution Strategies

- Instances with up to 42 nodes
  Mathematical formulations
- 50 nodes and larger instances
  - Constructive heuristics
  - Greedy Randomized Adaptive Search Procedure (GRASP) + Variable Neighborhood Descent (VND)

Constructive Heuristics

- Nearest Neighbour for the TSP

Table: Distance Matrix

	0	1	2	3	4	5
0	0.0	5.0	6.0	4.2	9.2	8.5
1	5.0	0.0	7.8	3.6	7.1	4.2
2	6.0	7.8	0.0	4.2	6.1	8.5
3	4.2	3.6	4.2	0.0	5.0	5.0
4	9.2	7.1	6.1	5.0	0.0	4.5
5	8.5	4.2	8.5	5.0	4.5	0.0

TSP Solution:

0						

Constructive Heuristics

- Nearest Neighbour for the TSP

Table: Distance Matrix

	0	1	2	3	4	5
0	0.0	5.0	6.0	**4.2**	9.2	8.5
1	5.0	0.0	7.8	3.6	7.1	4.2
2	6.0	7.8	0.0	4.2	6.1	8.5
3	4.2	3.6	4.2	0.0	5.0	5.0
4	9.2	7.1	6.1	5.0	0.0	4.5
5	8.5	4.2	8.5	5.0	4.5	0.0

TSP Solution: 4.2

0	3					

Constructive Heuristics

- Nearest Neighbour for the TSP

Table: Distance Matrix

	0	1	2	3	4	5
0	0.0	5.0	6.0	**4.2**	9.2	8.5
1	5.0	0.0	7.8	3.6	7.1	4.2
2	6.0	7.8	0.0	4.2	6.1	8.5
3	4.2	**3.6**	4.2	0.0	5.0	5.0
4	9.2	7.1	6.1	5.0	0.0	4.5
5	8.5	4.2	8.5	5.0	4.5	0.0

TSP Solution: 7.8

0	3	1				

Constructive Heuristics

- Nearest Neighbour for the TSP

Table: Distance Matrix

	0	1	2	3	4	5
0	0.0	5.0	6.0	**4.2**	9.2	8.5
1	5.0	0.0	7.8	3.6	7.1	**4.2**
2	6.0	7.8	0.0	4.2	6.1	8.5
3	4.2	**3.6**	4.2	0.0	5.0	5.0
4	9.2	7.1	6.1	5.0	0.0	4.5
5	8.5	4.2	8.5	5.0	4.5	0.0

TSP Solution: 12

0	3	1	5			

Constructive Heuristics

- Nearest Neighbour for the TSP

Table: Distance Matrix

	0	1	2	3	4	5
0	0.0	5.0	6.0	**4.2**	9.2	8.5
1	5.0	0.0	7.8	3.6	7.1	**4.2**
2	6.0	7.8	0.0	4.2	6.1	8.5
3	4.2	**3.6**	4.2	0.0	5.0	5.0
4	9.2	7.1	6.1	5.0	0.0	4.5
5	8.5	4.2	8.5	5.0	**4.5**	0.0

TSP Solution: 16.5

0	3	1	5	4		

Constructive Heuristics

- Nearest Neighbour for the TSP

Table: Distance Matrix

	0	1	2	3	4	5
0	0.0	5.0	6.0	**4.2**	9.2	8.5
1	5.0	0.0	7.8	3.6	7.1	**4.2**
2	6.0	7.8	0.0	4.2	6.1	8.5
3	4.2	**3.6**	4.2	0.0	5.0	5.0
4	9.2	7.1	**6.1**	5.0	0.0	4.5
5	8.5	4.2	8.5	5.0	**4.5**	0.0

TSP Solution: 22.6

0	3	1	5	4	2	

Constructive Heuristics

- Nearest Neighbour for the TSP

Table: Distance Matrix

	0	1	2	3	4	5
0	0.0	5.0	6.0	**4.2**	9.2	8.5
1	5.0	0.0	7.8	3.6	7.1	**4.2**
2	**6.0**	7.8	0.0	4.2	6.1	8.5
3	4.2	**3.6**	4.2	0.0	5.0	5.0
4	9.2	7.1	**6.1**	5.0	0.0	4.5
5	8.5	4.2	8.5	5.0	**4.5**	0.0

TSP Solution: 28.6

0	3	1	5	4	2	0

Constructive Heuristics

- Nearest Neighbour for the PDTSP

Table: Distance Matrix

	0	1	2	3	4	5
0	0.0	5.0	6.0	4.2	9.2	8.5
1	5.0	0.0	7.8	3.6	7.1	4.2
2	6.0	7.8	0.0	4.2	6.1	8.5
3	4.2	3.6	4.2	0.0	5.0	5.0
4	9.2	7.1	6.1	5.0	0.0	4.5
5	8.5	4.2	8.5	5.0	4.5	0.0
q	0	-6	3	1	4	-2

PDTSP Solution: 35.1

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

Constructive Heuristics

- Nearest Neighbour for the PDTSP with Split Demand

Table: Distance Matrix

	0	1	2	3	4	5
0	0.0	5.0	6.0	4.2	9.2	8.5
1	5.0	0.0	7.8	3.6	7.1	4.2
2	6.0	7.8	0.0	4.2	6.1	8.5
3	4.2	3.6	4.2	0.0	5.0	5.0
4	9.2	7.1	6.1	5.0	0.0	4.5
5	8.5	4.2	8.5	5.0	4.5	0.0
q	0	-10	-7	6	6	5

PDTSP Solution: Infeasible

Solution for PDTSP-Split: 38.3 splitting demand (6=5+1) in station 4

Route	0	1	5	4	2	3	4	0
Load	0	10	5	0	7	1	0	

Greedy Randomized Adaptive Search Procedure (GRASP)

```
procedure GRASP(MaxIterations, Seed)
1.    Set f* ← ∞;
2.    for k = 1, ..., MaxIterations do
3.        S ← GreedyRandomizedAlgorithm(Seed);
4.        if S is not feasible then
5.            S ← RepairSolution(S);
6.        end;
7.        S ← LocalSearch(S);
8.        if f(S) < f* then
9.            S* ← S;
10.           f* ← f(S);
11.       end;
12.   end;
13.   return S*;
end.
```

Figure: GRASP Procedure (Resende & Ribeiro, 2010)

Greedy Randomized Adaptive Search Procedure (GRASP)

procedure `GreedyRandomizedAlgorithm(Seed)`

1. $S \leftarrow \emptyset$;
2. Initialize the candidate set: $C \leftarrow E$;
3. Evaluate the incremental cost $c(e)$ for all $e \in C$;
4. **while** $C \neq \emptyset$ **do**
5. Build a list with the candidate elements having the smallest incremental costs;
6. Select an element s from the restricted candidate list at random;
7. Incorporate s into the solution: $S \leftarrow S \cup \{s\}$;
8. Update the candidate set C;
9. Reevaluate the incremental cost $c(e)$ for all $e \in C$;
10. **end**;
11. **return** S;

end.

Figure: Greedy Randomized Algorithm (Resende & Ribeiro, 2010)

Greedy Randomized Adaptive Search Procedure (GRASP)

From the Greedy Randomized (constructive) phase, it is possible to find feasible solutions:

Route	0	5	1	3	2	4	0	n	s
q	0	-2	-6	1	3	4			
Load	0	2	8	7	4	0		0	0

and infeasible solutions:

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

n: number of infeasible loads

s: sum of infeasible loads

Greedy Randomized Adaptive Search Procedure (GRASP)

How to repair infeasible solutions?

Four neighborhoods (so far) within a Variable Neighborhood Descent (VND) method

- Forward insertion
- Backward insertion
- Swap
- 2-Opt

Greedy Randomized Adaptive Search Procedure (GRASP)

- Forward insertion

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

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

n: number of infeasible loads

s: sum of infeasible loads

Greedy Randomized Adaptive Search Procedure (GRASP)

- Backward insertion

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

Route	0	5	**1**	3	2	4	0	n	s
q	0	-2	**-6**	1	3	4			
Load	0	2	8	7	4	0		0	0

n: number of infeasible loads

s: sum of infeasible loads

Greedy Randomized Adaptive Search Procedure (GRASP)

- Swap

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

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

n: number of infeasible loads

s: sum of infeasible loads

Greedy Randomized Adaptive Search Procedure (GRASP)

- 2-Opt

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

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

n: number of infeasible loads

s: sum of infeasible loads

Greedy Randomized Adaptive Search Procedure (GRASP)

Is it possible to improve efficiency for neighborhood movements?

for $i = 1$ to *stations* **do**
  for $j = i$ to 1 step -1 **do**
    for $k = 1$ to j **do**
      Load$+=q_k$
      if feasible(Load)=**false**
      then
        Break
      end if
    end for
    improve(Solution)
  end for
end for

Greedy Randomized Adaptive Search Procedure (GRASP)

- Backward insertion

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

Route	0	5	**1**	3	2	4	0	n	s
q	0	-2	**-6**	1	3	4			
Load	0	2	8	7	4	0		0	0

n: number of infeasible loads

s: sum of infeasible loads

Greedy Randomized Adaptive Search Procedure (GRASP)

Is it possible to improve efficiency for neighborhood movements?

```
for i = 1 to stations do
    for j = i to 1 step −1 do
        for k = 1 to j do
            Load+=q_k
            if feasible(Load)=false
            then
                Break
            end if
        end for
        improve(Solution)
    end for
end for
```

```
for i = 1 to stations do
    for j = i to 1 step −1 do
        (Load_min,Load_max)=SetBounds()
        if feasible(Load,Load_min,Load_max)
        then
            Load+=q_j
            improve(Solution)
        end if
    end for
end for
```

Preliminary Results

Data Sets and Software

- Data Sets
  - Instances adapted from TSPLib Library (`elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html`)
  - Instances with 9, 14, 16, 22, 29, 42, 52, 130, 280 and 582 nodes were tested
- Software
  - All the algorithms and models were coded on C++
  - Mathematical models were solved using Gurobi Optimizer 7.1
  - Intel Core i7, 16Gb RAM. OS: Fedora 24(x86-64)

Mixed Integer Programming Models (MIPs)- Results

St	TSP		TSPPD		TSPPD -Split	
	Distance	Time(s)	Distance	Time(s)	Distance	Time(s)
9	28.60	0.03	38.15	0.02	32.13	0.04
14	30.87	0.04	36.01	0.75	35.55	1.58
16	50.47	0.21	84.84	0.57	83.58	0.69
22	75.30	0.42	95.84	5.85	95.05	6.57
29	9073.31	0.91	13 529.20	89.61	12 550.40	854.19
42	679.02	1.12	1446.33	1018.57	1394.46*	4366

*Feasible solution with GAP $= 2.14\%$

Nearest Neighbour (NN)- Results

Stations	TSPPD			TSPPD Split		
	Distance	GAP	Time(s)	Distance	GAP	Time(s)
9	38.15	0.00	2.0E-06	39.67	0.23	5.8E-05
14	44.32	0.23	8.0E-06	44.32	0.25	4.0E-06
16	100.94	0.19	1.7E-05	100.94	0.21	1.1E-04
22	141.63	0.48	3.0E-05	137.75	0.45	2.1E-05
29	19 688.60	0.46	7.9E-05	17 379.80	0.38	2.1E-05
42	1889.76	0.31	1.0E-04	2157.73	0.55*	1.3E-04

$$GAP = 1 - \frac{Distance_{NN}}{Distance_{MIP}}$$

*Based on feasible solution reported by MIP

Nearest Neighbour (NN) - Results

	TSPPD		TSPPD - Split			
Stations	Distance	Time (s)	Distance	Time (s)	Nodes w. split	Visits x node
52	15 141.90	2.1E-04	14 256.1	7.20E-05	1	2
130	13 433.10	7.72E-04	13 599.2	1.53E-04	2	2
280	*	*	6362.99	5.62E-04	3	2
535	7229.68	1.35E-02	8209.22	2.16E-03	13	2

* NN does not return a feasible solution in less than 30 s.

GRASP + VND - Results

An example of the experiments with GRASP (280 stations)

Iterations	RCL	Distance	n	s	Time (s)
50	2	9311.26	0	0	0.06
50	3	10 149.20	2	4	0.10
50	4	11 312.40	1	3	0.14
50	5	12 277.90	1	8	0.19
100	2	8573.76	0	0	0.12
100	3	9898.20	0	0	0.19
100	4	10 516.10	2	11	0.27
100	5	11 239.70	2	5	0.37
500	2	8629.99	0	0	0.57
500	3	10 031.00	0	0	0.95
500	4	10 618.00	0	0	1.38
500	5	12 149.50	0	0	1.84

n: number of infeasible loads (in best solution)

s: sum of infeasible loads (in best solution)

GRASP + VND - Results

	MIP	GRASP + VND			
Stations	Distance	Distance	Feasible?	GAP	Time (s)
9	38.15	38.15	✓	0.00	0.001
14	36.02	38.24	✓	0.06	0.002
16	84.84	96.94	✓	0.12	0.002
22	95.84	107.85	✓	0.11	0.005
29	13,529.20	15,424.10	✓	0.12	0.002
42	1446.33	1631.60	✓	0.11	0.017

$$GAP = 1 - \frac{Distance_{MIP}}{Distance_{GRASP}}$$

GRASP + VND - Results

Stations	% Feasible solutions	Average time (s)
9	100.00%	0.001
14	100.00%	0.002
16	100.00%	0.003
22	100.00%	0.005
29	100.00%	0.008
42	91.67%	0.052
52	100.00%	0.022
130	79.17%	0.12
280	58.33%	0.516
535	66.67%	2.025

Current and Future Work

- Design exact or heuristic strategies able to repair infeasible solutions in 100% of the instances.
- Include inventory constraints in TSPPD model.
- Include stochastic demands for each station.
- Design exact and heuristic strategies able to include synchronization features in several routes.

ASOCIO 2017


UNIVERSIDAD EAFIT
Vigilada Mineducación
ASOCIO
Asociación Colombiana de Investigación de Operaciones
II Congreso Colombiano de
Investigación Operativa
ASOCIO 2017

References I

Brinkmann, J., Ulmer, M. W., & Mattfeld, D. C. (2015). Inventory Routing for Bike Sharing Systems. *Transportation Research Procedia*, *19*(June), 1–22.
URL http://dx.doi.org/10.1016/j.trpro.2016.12.091

Dell'Amico, M., Iori, M., Novellani, S., & Stützle, T. (2016). A destroy and repair algorithm for the Bike sharing Rebalancing Problem. *Computers and Operations Research*, *71*, 149–162.
URL http://dx.doi.org/10.1016/j.cor.2016.01.011

Forma, I. A., Raviv, T., & Tzur, M. (2015). A 3-step math heuristic for the static repositioning problem in bike-sharing systems. *Transportation Research Part B: Methodological*, *71*, 230–247.
URL http://dx.doi.org/10.1016/j.trb.2014.10.003

References II

Ho, S. C., & Szeto, W. Y. (2017). A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem. *Transportation Research Part B: Methodological*, *95*, 340–363. URL `http://dx.doi.org/10.1016/j.trb.2016.11.003`

Kadri, A. A., Kacem, I., & Labadi, K. (2016). A branch-and-bound algorithm for solving the static rebalancing problem in bicycle-sharing systems. *Computers and Industrial Engineering*, *95*, 41–52. URL `http://dx.doi.org/10.1016/j.cie.2016.02.002`

Resende, M. G., & Ribeiro, C. C. (2010). Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In *Handbook of metaheuristics*, (pp. 283–319). Springer.

Schuijbroek, J., Hampshire, R. C., & van Hoeve, W. J. (2017). Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, *257*(3), 992–1004. URL `http://dx.doi.org/10.1016/j.ejor.2016.08.029`

References III

Szeto, W. Y., Liu, Y., & Ho, S. C. (2016). Chemical reaction optimization for solving a static bike repositioning problem. *Transportation Research Part D: Transport and Environment*, *47*, 104–135.
URL `http://dx.doi.org/10.1016/j.trd.2016.05.005`

Vehicle Routing Optimization in Bicycle-sharing Systems

Juan David Palacio Domínguez
jpalac26@eafit.edu.co

Juan Carlos Rivera Agudelo
jrivera6@eafit.edu.co