Facility Location Models for Vehicle Sharing Systems

Alain Quilliot
LIMOS CNRS, Labex IMOB3
Université Blaise Pascal
63000 Clermont-Ferrand, France

Antoine Sarbinowski
LIMOS CNRS, Labex IMOB3
Université Blaise Pascal
63000 Clermont-Ferrand, France.

Email: alain.quilliot@isima.fr

Abstract—Designing a vehicle sharing system means locating stations which allow users to pick up and give back vehicles. One takes this strategic level decision while anticipating related rebalancing costs. We study here a strategic related bi-level Vehicle Sharing Station Location (VSSL) model, which involves as slave problem a static Vehicle Sharing Rebalancing (VSR) model.

I. INTRODUCTION

Vehicle Sharing systems (see [4]), involving bikes or electric cars, are among the systems which currently strive in order to find their place in the urban mobility landscape as a compromise between full individual transportation and rigid public transportation. They most often work as one-way systems: customers should be allowed to pick up a vehicle at any station and give it back at any other station. But the system may fast become unbalanced, with either empty or overfilled stations, making arise two decision problems:

- a strategic level problem (see [4]) about the way stations are located and capacitated.
- an operational (or tactical) level problem (see [2, 3, 5, 6]), about the way the Rebalancing Process is performed.

This contribution deals with the strategic level problem, which has been scarcely studied, and which we refer to as the Vehicle Sharing Station Location (VSSL). We link it with the operational level known as the Vehicle Sharing Rebalancing Problem (VSRP).

Efficiently locating the stations of the system means:

- locating the stations close to the origins and destinations of the users, in such a way that a global Access Demand be maximized, or at least some target value be reached;
- minimizing investment and infrastructure costs;
- making in such a way that the expected running costs due to the periodic Rebalancing Process be the smallest possible.

While the two first criteria yield standard Facility Location models, (see [9]), dealing with the last one leads us to explicit those expected running costs due to the periodic Rebalancing Process: This process consists in periodically picking up some vehicles at excess stations, that means stations which may be considered as containing more than enough vehicles, and move them to deficit stations, while using carriers (trucks, self-platoon convoys...). Optimizing this process gives rise to Vehicle Sharing Rebalancing (VSR) models. While on line VSR models received very little attention, being only handled through application of empirical decision rules (see [5, 7]), several static VSR models (see [2, 3, 7, 8]) have already been proposed and studied through heuristics and ILP models.

Our purpose is here to cast operational VSR as a slave sub-problem of a strategic Vehicle Sharing Station Location (VSSL) model. We first consider that the input data for VSSL problem mainly consists in an origin/destination matrix OD, and in additional information about demands and costs, and derive (Section II) a bi-level Vehicle Sharing Station Location (VSSL) whose master problem IS a Facility Location model and slave sub-model is some static VSR model. Next we propose (Section III) a related bi-level algorithmic resolution scheme which decomposes in turn the VSR model into a simple Min-Cost Assignment master model and a slave PDP: Pick up and Delivery model (see [1, 5]) model. We end by providing a VSR lower bound and performing numerical experiments (Section IV).

II. THE VSSL MODEL

A. Vehicle Sharing Station Location Instances

VSSL (Vehicle Sharing Station Location) input is a set VS of virtual stations, given together with:

- a demand matrix OD: for any x, y in VS, OD(x, y) means the access demand to the system in x, y, that means the number of vehicles which should be
picked up at station \( x \) and given back at station \( y \) by the users during a reference period \( P \).
- a distance matrix \( \text{DIST} : \text{DIST}(x, y) \) means the distance (time required) from \( x \) to \( y \).

**Demands and Costs:** Solving VSSL means computing a real station subset \( X \) of \( VS \) and its related capacity function \( C \) given a subset \( X \) of \( VS \) and a station \( u \) in \( VS \), we denote by \( \text{Prox}(u, X) \) the element \( x \) in \( X \) which is the closest to \( u \) then the Access Demand \( \text{Acc}(x, y, X) \) which is induced by \( X \) between two stations \( x \) and \( y \) of \( X \) is given by:

\[
\text{Acc}(x, y, X) = \sum_{u \in X} \Phi(\text{Dist}(u, \text{Prox}(u, X))) \Phi(\text{Dist}(x, \text{Prox}(x, X))),
\]

where \( \Phi \) is a decreasing \([0, 1]\)-valued function.

We set: \( \text{Global-Demand}(X) = \sum_{x \in X} \text{Acc}(x, y, X) \).
This Access Demand induces, for any station \( x \) in \( X \), a residual quantity \( \text{Res}(x, X) \):

\[
\text{Res}(x, X) = \sum_{y \in X} \text{Acc}(x, y, X) - \sum_{y \in X} \text{Acc}(y, x, X).
\]

This residual quantity means the number of vehicles which is likely to be in excess (\( \text{Res}(x, X) > 0 \)) or in deficit at station \( x \) at the end of the reference period \( P \).

The Top Demand in station \( x \in X \), i.e. the variation between the least and the largest numbers of vehicles in station \( x \) during period \( P \), is given by:

\[
\text{Top}(x, X) = \text{Q}(x, X) \text{H}(x, X) \text{with:} \quad \text{Q}(x, X) = \text{Sup}(\sum_{y \in X} \text{Acc}(y, x, X), \sum_{y \in X} \text{Acc}(x, y, X)) \quad \text{H}(x, X) = \Pi/|\text{Res}(x, X)|/|\text{Q}(x, X)|,
\]

\( \Pi \) being a decreasing \([0, 1]\)-valued function, \( \alpha > 0 \).

Setting a station at node \( x \) in \( VS \) with capacity \( C(x) \), has a fixed cost \( \text{Fix}(x) \), augmented with a flexible cost \( \text{Prop}(x) \), which linearly depends on \( C \).
Besides, since running the system defined by \( X \) and function \( C \) periodically requires relocating vehicles from excess stations to deficit stations, we denote by \( \text{Run-Cost}(X, C) \) the cost of this rebalancing process.

**Constraints:** \( X \subseteq VS \) and \( C \) are subject to:
- **Capacity Constraints:** for any \( x \in X \), \( \text{Top}(x, X) \leq C(x) \);
- **Demand Constraints:** Global Demand should be at least equal to some target level \( \text{Goal} \): \( \text{Global-Demand}(X) \geq \text{Goal} \).

Then the VSSL model comes as follows:

**VSSL Model** : Compute the subset \( X \), the Depot station \( D \), and the capacity function \( C \) such that the Capacity and Demand Constraints be satisfied and that:

\[
\text{Cost} = \sum_{x \in X} (\text{Fix}(x) + C(x) \text{Prop}(x)) + \text{Depot-Cost}(D) + \text{Run-Cost}(X, C, D) \text{ is minimal}.
\]

We denote by \( \text{Relax-VSSL} \) the restriction of VSSL which is obtained by removing the \( \text{Run-Cost} \) quantity.

**B. The Vehicle Sharing Rebalancing Problem: VSR**

Let us suppose that \( X \) and \( C \) are given, together with the Depot station \( D \). For any station \( x, \nu(x) = \text{Res}(x, X) \) vehicles are in excess at station \( x \): if \( \nu(x) < 0 \), we talk about deficit.
We suppose \( \sum_{x \in x} \nu(x) = 0 \), which means that \( D \) may bring additional vehicles to the system.
\( K-\text{Max} \) is the number of available carriers, all with capacity \( C \), and initially located at \( D \).
This defines the VSR instance \( (X, \nu, C, D, K-\text{Max}) \).

**VSR Feasible Solutions:** A VSR tour \( \gamma \) is a finite sequence \( \gamma_{\text{Route}} = \{x_0 = D, x_1, \ldots, x_{n(\gamma)} = D\} \) of stations, given together with a loading strategy, that means with 2 sequences \( \gamma_{\text{Load}} = \{L_0, L_1, \ldots, L_{n(\gamma)}\} \) and \( \gamma_{\text{Time}} = \{T_0 = 0, T_1, \ldots, T_{n(\gamma)}\} \) of coefficients whose meaning is: a carrier which follows the route \( \gamma \) loads, at time \( T_i \), \( L_i \) vehicles at station \( x_i \) (unloads in case \( L_i < 0 \)).
This VSR tour \( \gamma \) is feasible if:

- For any \( i = 0, \ldots, n(\gamma) - 1 \),
  \[
  T_{i+1} \geq T_i + \text{DIST}(x_i, x_{i+1});
  \]  
  \[
  \text{E1}
  \]
- For any \( i = 0, \ldots, n(\gamma) - 1 \),
  \[
  \text{L}_x^* = \sum_{j=0}^{n(\gamma)} \text{L}_j \leq \text{CAP};
  \]  
  \[
  \text{E2}
  \]
- For any \( j \) such that \( \nu(x_j) \geq 0, \nu(x_j) \geq \text{L}_j \geq 0, \)  
  \[
  \text{E4}
  \]
- For any \( j \) such that \( \nu(x_j) \leq 0, \nu(x_j) \leq \text{L}_j \leq 0, \)
  \[
  \text{E5}
  \]

Then a feasible solution for the VSR instance \( (X, \nu, C, D, K-\text{Max}, \text{DIST}) \) is a collection \( \Gamma = (\Gamma(k), k = 1..K \leq K-\text{Max}) \) of feasible tours, such that, for any station \( x \): \( \sum_{x \in X} \nu(x) = 0 \)

\[
\text{Sup}(\sum_{T(k)}(L_k^{(x)} + \chi \sum_{k} T(k) L_k^{(x)} + \delta(\sum_{x} \text{DIST}(x(k), x(k+1)) \text{L}_x^*))).
\]

where \( \alpha, \beta, \chi, \delta \) are some scaling coefficients.

We derive the following **VSR Model**: Compute a feasible VSR solution \( \Gamma = (\Gamma(k), k = 1..K) \) which minimizes the above quantity \( R-\text{Cost}(\Gamma) \).

**Remark 1:** The \( \text{Run-Cost}(X, C, D) \) quantity of the VSSL model is the optimal value of this VSR model.

### III. Algorithms

We deal with the VSSL model according to a GRASP hierarchical decomposition scheme:

**VSSL-GRASP Scheme**

Initialize \( X \) and \( C \) while solving \( \text{Relax-VSSL} \);
Not Stop; While Not Stop do

Solve the slave VSR model induced by \( X \), (*) Derive an additional constraint \( C-Acc(x) \), and update \( X, C \) through local search;
We implement the first instruction by adapting Facility Location algorithms (see [8]) into a Relax-VSSL procedure, while observing that the capacity function \( C \) derives from \( X \) and the Top Demand function \( x, X \rightarrow Top(x, X) \) in a straightforward way. The resulting procedure is a GRASP Algorithm, which involves local search operators Insert(\( x \)), Remove(\( x \)), Replace(\( x, y \)) and Merge(\( x, y \)).

\( X \) being given, let us now explain how we deal with the resulting VSR sub-problem ((*) instruction).

### A. Decomposing VSR into Min Cost Assignment and PDP: The Distance Strategy.

In case we could decide, for any pair \( (x, y) \), \( x \) excess, \( y \) deficit station, which quantity \( Q_{x,y} \) has to move from \( x \) to \( y \) in order to achieve the rebalancing process, then we derive a VSR solution by solving the Load Splittable PDP instance (see [1]) defined by:

- Requests correspond to the 3-uple \( (o(j) = x, d(j) = y, Load(j) = Q_{x,y} \neq 0) \)
- Minimize \( \alpha.K + \beta.\left( \sum k \right) \text{Length}(\Gamma(k)) + \gamma.\left( \sum j \right) \text{Length}(\Gamma(j)) + \delta.\sum j \text{Ride}(j) \); \( k \) denotes the vehicles, \( \Gamma(k) \) the related PDP tours and \( \text{Ride}(j) \) is the time spent by \( \text{Load}(j) \) inside a truck.

We check that:

**Theorem 1:** We may restrict ourselves to vectors \( Q = (Q_{x,y}, x \text{ excess station, } y \text{ deficit station}) \) which are vertices of the Assignment polyhedron \( P:\text{Assign} \):

\[ P:\text{Assign}: \{ Z = (Z_{x,y}, x \text{ excess, } y \text{ deficit}) | \text{such that:} \]

- For any excess station \( x \), \( \Sigma y \text{ deficit } Z_{x,y} = v(x) \);
- For any excess station \( x \), \( \Sigma x \text{ excess } Z_{x,y} = -v(x) \)

This leads us to handle VSR through the following decomposition scheme:

**VSR Assignment/PDP Decomposition Scheme:**

Initialize cost vector \( Q \); Not Stop;

While Not Stop do

- Derive \( Z \) and the Request set \( J = J(Z) \);
- Solve the Load Splittable PDP related instance;
- Update \( Q \).

The Distance Strategy: Initializing \( Q \) comes in a natural way by setting: for any \( x, y, \) \( x \) excess, \( y \) deficit stations, \( Q_{x,y} = \text{DIST}(x, y) \). We call this strategy, the Distance Strategy. We may state:

**Theorem 2:** If \( K \) is fixed and \( \alpha, \beta, \delta \) equal to 0 (we minimize the carrier riding time), then the Distance strategy induces a VSR approximation ratio of \((1+\text{CAP})\). This is the best possible ratio.

**Theorem 3:** If \( K \) is fixed and \( \alpha, \beta, \delta \) equal to 0 (we minimize the makespan), then the Distance Strategy induces a VSR approximation ratio of \((1+K\cdot\text{CAP})\). This is the best possible ratio.

### B. VSR-Assignment/PDP Algorithm

We follow the guideline of the previously described hierarchical decomposition scheme. As a matter of fact, we revisit it as follows

**VSR-Assignment/PDP Algorithm:**

Initialize cost vector \( Q \); Derive \( Z \) and the Request set \( J = J(Z) \); Solve the Load Splittable PDP related instance through some generic Insertion algorithm and get a current VSR solution \( \Gamma \); Not Stop;

While Not Stop do

- Update cost vector \( Q \) and the Request set \( J = J(Z) \); Let \( J_0 \) the set of formerly existing requests which have been removed from \( J \) and \( J_1 \) the set of newly created requests; Remove \( J_0 \) and next Reinsert \( J_1 \), in the sense of the PDP Insertion algorithm, into current solution \( \Gamma \);

Cost vector \( Q \) and related Request set \( J \) are updated by:

1. **1st Step:** Identify a subset \( J_0 \subseteq J \) of poorly inserted requests (those with a large gap between cost \( Q_{x,y} \) and mean riding time \( R_{x,y} \));

2. **2nd step:** Set, for any \( x, y \) involved into \( J_0 \), \( x \) excess, \( y \) deficit, \( Q_{x,y} = (Q_{x,y} + R_{x,y})/2 \).

**C. Retrieving Sensitivity Constraint C-Aux(X)**

A key instruction inside the main loop of the VSSL-GRASP algorithm is the following:

“Derive an additional constraint C-Aux(X)…”

We implement it while using the dual solution \( \lambda \), \( x \in VS \) of the Min-Cost Assignment problem related to current vector \( Q \), as a sub-gradient vector and derive the following Bender’s like constraint C-Aux(X):

\[ \Sigma x \in VS \text{Res}(x, X), \lambda_x \leq \Sigma x \in VS \text{Res}(x, X), \lambda_x \]

### D. A Lower Bound for the VSR model

We get a VSR lower bound \( LB \) by introducing (see [8]) a network with time indexed nodes and turning Preemptive VSR (carriers may exchange vehicles while performing the Rebalancing process) into a network flow model, which involves an integral carrier flow vector dominating some rational vehicle flow vector. Practically, we compute \( LB \) while using an ILP solver and applying some rounding process when the size of \( G \) is too large.
IV. NUMERICAL EXPERIMENTS

Since we can’t provide exact reference values for the VSSL model, we separately evaluate the distinct components of the VSSL-GRASP Algorithm.

A. Testing VSR-Assignment/PDP and Relax-VSSL

A VSR instance is identified by the numbers \( n, nd, K_{-Max} \), by the matrix \( DIST \), and by function \( v \). \( T_{-Max} \) is set to 480. We compute, for any instance:

- the value \( LB \) of the lower bound of Section III;
- the value \( V-NP_{-Dist} (V-NP) \) of the solution related to the Distance strategy (VSR-Assignment/PDP) and its related CPU time;
- Assignment/PDP ) and its related CPU time;

We get (on PC AMD Opteron 2.1GHz, while using gcc 4.1 compiler and the CPLEX12 library):

<table>
<thead>
<tr>
<th>( n,n_{k} )</th>
<th>( K_{-Max} )</th>
<th>( V-NP_{-Dist}/LB(%) )</th>
<th>( V-NP_{-Dist}/CPU(%) )</th>
<th>( VNP/L_{-B}(%) )</th>
<th>( NP_{-CPU} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>48, 20, 3</td>
<td>19.6</td>
<td>4.6</td>
<td>14.6</td>
<td>17.6</td>
<td></td>
</tr>
<tr>
<td>48, 30, 3</td>
<td>24.0</td>
<td>6.7</td>
<td>16.2</td>
<td>20.5</td>
<td></td>
</tr>
<tr>
<td>48, 40, 3</td>
<td>12.4</td>
<td>8.9</td>
<td>10.4</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>96, 20, 5</td>
<td>13.8</td>
<td>7.1</td>
<td>10.8</td>
<td>30.5</td>
<td></td>
</tr>
<tr>
<td>96, 60, 5</td>
<td>22.6</td>
<td>10.3</td>
<td>14.8</td>
<td>48.8</td>
<td></td>
</tr>
<tr>
<td>148, 40, 7</td>
<td>11.8</td>
<td>10.1</td>
<td>8.5</td>
<td>20.2</td>
<td></td>
</tr>
<tr>
<td>148, 80, 7</td>
<td>15.7</td>
<td>18.3</td>
<td>14.7</td>
<td>48.4</td>
<td></td>
</tr>
</tbody>
</table>

**Comment:** The LB value provides us with a rather good approximation. Though the Distance strategy is rather efficient, we improve V-NP values in a significant way by fully performing local search.

In order to test Relax-VSSL, we generate a set \( VS \) of \( n \) points of the Euclidian space \( R^{2} \), (so \( DIST \) means the Euclidian distance), and an origin/destination matrix \( OD \), with all values \( OD(x, y) \) between 0 and a given parameter \( S \), and uniformly distributed. Functions \( \Phi \) and \( \Pi \) are piecewise linear. We compute, for every instance, the gap \( G \) between the CPLEX optimal solution and the of Relax-VSSL together with related CPU times \( T_{-ILP} \) and \( T_{-Rel} \). Then we get:

<table>
<thead>
<tr>
<th>( n ), ( S ), ( q ) (%)</th>
<th>( G(%) )</th>
<th>( T_{0}(s) )</th>
<th>( T_{1}(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20, 10, 25%, 20, 50</td>
<td>8.5</td>
<td>42.5</td>
<td>288.4</td>
</tr>
<tr>
<td>20, 20, 50%, 20, 50</td>
<td>17.6</td>
<td>43.1</td>
<td>360.8</td>
</tr>
<tr>
<td>20, 30, 75%, 5, 10</td>
<td>30.6</td>
<td>11.2</td>
<td>152.6</td>
</tr>
<tr>
<td>20, 30, 75%, 20, 50</td>
<td>35.9</td>
<td>43.6</td>
<td>506.2</td>
</tr>
<tr>
<td>50, 10, 25%, 5, 20</td>
<td>5.0</td>
<td>29.6</td>
<td>304.0</td>
</tr>
<tr>
<td>50, 10, 50%, 5, 20</td>
<td>12.6</td>
<td>29.6</td>
<td>334.8</td>
</tr>
<tr>
<td>50, 10, 75%, 5, 20</td>
<td>28.4</td>
<td>29.6</td>
<td>350.6</td>
</tr>
<tr>
<td>50, 10, 75%, 10, 50</td>
<td>31.3</td>
<td>58.8</td>
<td>1482.0</td>
</tr>
</tbody>
</table>

**Comment:** Computing costs increase with the \( S \) value.

B. Testing VSSL-GRASP

A VSSL test is identified here by:

- the coefficients \( n \) (cardinality of \( VS \), \( S \) (top \( OD(x, y) \) value), \( q \) (relative weight of Run-Cost inside the global cost of a solution);
- the number \( M \) of replications of the VSSL-GRASP scheme;
- the length \( L \) of the main loop of VSSL-GRASP.

We compute, for any instance, the gap \( G \) between obtained through Relax-VSSL and the final cost obtained through VSSL-GRASP, together with related CPU times \( T_{0} \) and \( T_{1} \). Then we get:

<table>
<thead>
<tr>
<th>( n ), ( M ), ( L )</th>
<th>( G(%) )</th>
<th>( T_{0}(s) )</th>
<th>( T_{1}(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10, 5</td>
<td>1.1</td>
<td>29.6</td>
<td>4.1</td>
</tr>
<tr>
<td>15, 5</td>
<td>0</td>
<td>126</td>
<td>6.9</td>
</tr>
<tr>
<td>15, 10</td>
<td>0</td>
<td>*</td>
<td>14.0</td>
</tr>
<tr>
<td>20, 5</td>
<td>2.8</td>
<td>1588</td>
<td>10.7</td>
</tr>
<tr>
<td>20, 10</td>
<td>0</td>
<td>*</td>
<td>21.2</td>
</tr>
</tbody>
</table>

**REFERENCES**


