Proceedings of the Federated Conference on Computer Science and Information Systems pp. 605–608
DOI: 10.15439/2016F10
ACSIS, Vol. 8. ISSN 2300-5963

Facility Location Models forVehicle Sharing Systems

Alain Quilliot
LIMOS CNRS, Labex IMOBS3
Université Blaise Pascal
63000 Clermont-Ferrand, France

Antoine Sarbinowski
LIMOS CNRS, Labex IMOBS3
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 $DIST$: $DIST(x,y)$ 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 $Prox(u, X)$ the element x in X which is the closest to u Then the *Access Demand* $Acc(x, y, X)$ which is induced by X between two stations x and y of X is given by:

- $Acc(x, y, X) = \Sigma_{u,\, v \text{ in } VS \text{ such that } x = Prox(u, X),\, y = Prox(v, X)}$ $OD(u,v).\Phi(Dist(u, Prox(u, X)).\Phi(Dist(v, Prox(v, X))$, Φ being a decreasing [0, 1]-valued function.

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

- $Res(x, X) = \Sigma_y Acc(y, x, X) - \Sigma_y Acc(x, y, X)$.

This *residual* quantity means the number of vehicles which is likely to be in excess ($Res(x, X) > 0$) or in deficit at station x at the end of standard 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:

$Top(x, X) = Q(x, X).H(x, X)$ with :
$Q(x, X) = Sup(\Sigma_y Acc(y, x, X)), \Sigma_y Acc(x, y, X))$
$H(x, X) = \Pi(|Res(x, X)|/Q(x, X))$,

Π being a decreasing $[\alpha, 1]$-valued function, $\alpha > 0$.

Setting a station at node x in VS with capacity $C = C(x)$, has a fixed cost $Fix(x)$, augmented with a flexible cost $C.Prop(x)$, 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 $Run\text{-}Cost(X, C)$ the cost of this *rebalancing process*.

Constraints: $X \subseteq VS$ and C are subject to:

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

Then the VSSL model comes as follows:

VSSL Model : {Compute the subset X, the *Depot* station D, and the capacity function C in such a way that the *Capacity and Demand Constraint*s be satisfied and that:

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

We denote by *Relax*-VSSL the restriction of VSSL which is obtained by removing the *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, $v(x) = Res(x, X)$ *vehicles* are in *excess* at station x: if $v(x) < 0$, we talk about *deficit*. We suppose $\Sigma_{x \in X} v(x) = 0$, which means that D may bring additional *vehicles* to the system. *K-Max* is the number of available *carriers*, all with capacity *CAP* and initially located at D. This defines the *VSR* instance (X, v, C, D, *K-Max*).

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

- For any $i = 0, .., n(\gamma)\text{-}1$,
  $T_{i+1} \geq T_i + DIST(x_i, x_{i+1})$; (E1)
- For any $i = 0, .., n(\gamma)\text{-}1$,
  $L^*_i = \Sigma_{j = 0..i} L_j \leq CAP$; (E2)
- $\Sigma_{j = 0..n(\gamma)} L_j = 0$; (E3)
- For any j such that $v(x_j) \geq 0$, $v(x_j) \geq L_j \geq 0$; (E4)
- For any j such that $v(x_j) \leq 0$, $v(x_j) \leq L_j \leq 0$. (E5)

Then *a feasible solution* for the *VSR* instance (X, v, C, D, *K-Max*, *DIST*) is a collection $\Gamma = (\Gamma(k), k = 1..K \leq K\text{-}Max)$ of *feasible tours*, such that, for any station x: $\Sigma_k \Sigma_{i \text{ such that } x(k)i = x} L(k)_i = v(x)$. (E6)
The cost of Γ is given by: $R\text{-}Cost(\Gamma) = \alpha.K + \beta.\, Sup_k\, T(k)_{n(\Gamma(k))} + \chi.\, \Sigma_k T(k)_{n(\Gamma(k))} + \delta.(\Sigma_k \Sigma_j DIST(x(k)_j, x(k)_{j+1}).L^*_j)$,
where α, β, χ, δ 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 $Run\text{-}Cost(X, C, D)$ 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 *Relax*-VSSL;
Not *Stop*; While Not *Stop* do
Solve the slave VSR model induced by X; (*)
Derive an additional constraint $C\text{-}Aux(X)$, 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 -> $Top(x, X)$ in a straighforward 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 Splitable* 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 $\alpha..K + \beta.$ Sup $_k$ $Length(\Gamma(k)) + \gamma.\ \Sigma_k\ Length(\Gamma(k)) + \delta.\ \Sigma_j\ Ride(j)$: k denotes the vehicles, ($\Gamma(k)$) the related PDP tours and *Ride*(j) is the time spent by *Load*(j) inside a *truc*k.

We check that:

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

P-Assign: {Z = ($Z_{x,y}$, x excess, y deficit) such that:

- *For any excess station x, $\Sigma_{y\ deficit} Z_{x,y} = v(x)$;*
- *For any excess station x, $\Sigma_{x\ excess} Z_{x,y} = - v(x)$}*

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 Splitable* 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, $Q_{x,y} = DIST(x, y)$. We call this strategy, the *Distance Strategy*. We may state:

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

Theorem 3: *If K is fixed and χ, δ equal to 0 (we minimize the makespan), then the Distance Strategy induces a VSR approximation ratio of (1+K.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 Splitable* PDP related instance through some generic *Insertion* algorithm and get a current VSR solution Γ; 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 Γ;

Cost vector Q and related *Request* set J are updated by:

- 1 th 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 th 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 λ_x, $x \in VS$ 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):

$$\Sigma_{x \in VS} \mathrm{Res}(x, X).\ \lambda_x \le \Sigma_{x \in VS} \mathrm{Res}(x, X_0).\ \lambda_x .$$

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 I:
Testing *VSR-Assignment/PDP*

n , n_A, K-Max	*V-NP-Dist/LB(%)*	*V-NP-Dist-CPU(s)*	*VNP/LB (%)*	*NP-CPU*
48, 20, 3	19.6	4.6	14.6	17.6
48, 30, 3	24.0	6.7	16.2	20.5
48, 40, 3	12.4	8.9	10.4	29.0
96, 20, 5	13.8	7.1	10.8	30.5
96, 60, 5	22.6	10.3	14.8	48.8
148, 40, 7	11.8	10.1	8.5	20.2
148, 80, 7	15.7	18.3	14.7	48.4

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 Φ and Π 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, while always setting *S* to 10:

Table II:
Testing *Relax-VSSL*

n , M	*G (%)*	*T-ILP (s)*	*T-Rel*
10, 5	1.1	29.6	4.1
15, 5	0	126	6.9
15, 10	0	*	14.0
20, 5	2.8	1588	10.7
20, 10	0	*	21.2

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 the initial cost obtained through *Relax-VSSL* and the final cost obtained through *VSSL-GRASP*, together with related CPU times *T0* and *T1*. Then we get:

Table III:
Testing *VSSL-GRASP*

n , S, q (%) M, L	*G (%)*	*T0(s)*	*T1*
20, 10, 25%, 20, 50	8.5	42.5	288.4
20, 20, 50%, 20, 50	17.6	43.1	360.8
20, 30, 75%, 5, 10	30.6	11.2	152.6
20, 30, 75%, 20, 50	35.9	43.6	506.2
50, 10, 25%, 5, 20	5.0	29.6	304.0
50, 10, 50%, 5, 20	12.6	29.6	334.8
50, 10, 75%, 5, 20	28.4	29.6	350.6
50, 10, 75%, 10, 50	31.3	58.8	1482.0

Comment: Computing costs increase with the *S* value.

References

[1] C. Archetti, M. Speranza: Vehicle routing problems with split deliveries; p 3-22, ITOR, (2012). http://dx.doi.org/10.1111/j.1475-3995.2011.00811.x

[2] M. Benchimol, P. Benchimol, B. Chappert, A. De la Taille, F. Laroche, F. Meunier, L. Robinet: Balancing the stations of a self service bike hiring systems, *RAIRO-RO* 45, p 37-61, (2011). http://dx.doi.org/10.1051/ro/2011102

[3] D. Chemla, F. Meunier, R. Wolfler Calvo: Bike sharing systems: solving the static rebalancing problem; *Discrete Optimization* 10 (2), p. 120-146, (2013). http://dx.doi.org/10.1016/j.disopt.2012.11.005

[4] D. Gavalas, C. Konstantopoulos, G. Pantziou: Design and management of vehicle sharing systems: a survey of algorithmic approaches; *ArXiv e-prints*, October 2015. https://arxiv.org/abs/1510.01158v1

[5] G. H. Kek, R. L. Cheu, Q. Meng, C. Ha Fung: A study on the vehicle size and transfer policy for car rental problems *Transportation Res. E: Logistics and Transp. Review* 64 (1), p 110-121, (2014), http://dx.doi.org/10.1016/j.tre.2014.01.007

[6] M. Nourinedjad, M. J. Roorda: A dynamic carsharing decision support system; *Transp. Res. E*, 66, p 36-50, 2014). http://dx.doi.org/10.1016/j.tre.2014.03.003

[7] B. Bernay, S. Deleplanque, A. Quilliot: Routing in Dynamic Networks: Grasp Versus Genetics, *7 th WCO Workshop*, FEDCIS Conf, Warwsaw, p 487, 492, (2014) http://dx.doi.org/10.15439/978-83-60810-58-3

[8] A. Klose, A. Drexl: "Facility location models for distribution systems"; *EJOR* 162, p 429-449, 2005. http://dx.doi.org/10.1016/j.ejor.2003.10.031.