Self-service bike sharing systems: simulation, repositioning, pricing

Daniel Chemla, Frédéric Meunier, Thomas Pradeau, Roberto Wolfler Calvo, Houssame Yahiaoui

SELF-SERVICE BIKE SHARING SYSTEMS: SIMULATION, REPOSITIONING, PRICING

DANIEL CHEMLA, FRÉDÉRIC MEUNIER, THOMAS PRADEAU, ROBERTO WOLFLER CALVO, AND HOUSSAME YAHIAOUI

Abstract. Self-service bike sharing systems experience often *imbalance* problems: in some stations there is a lack of bikes while in others there are too many bikes, leaving no empty rack for a user willing to park his bike. Imbalance problems can be partially settled by the actions of vehicles moving bikes during the night in order to prepare the forthcoming day. We speak then of *static regulation*, since bikes can be assumed to be not moving. These problems can also be partially countered by actions of the vehicles during the day. This is *dynamic regulation* and forms the topic of our present work.

We build a precise theoretical framework for studying this dynamic problem and discuss the kind of impact vehicles moving bikes can have on the system. We prove moreover the inherent hardness of the dynamic regulation problem and present some ways to circumvent it by the use of heuristics. A pricing technique, not involving vehicles, is also proposed. An open-source and versatile simulator is also briefly described and used to compare the aforementioned methods.

1. Introduction

1.1. **Context.** The launch of the celebrated Vélib' in Paris in 2007 has started a new trend in mobility: self-service bike sharing systems have widespread all over the world. The current largest one is in the city of Hangzhou, China, with about 2′400 stations and 60′000 bikes. If such systems cannot replace public transportation, they work as an incentive to stimulate people to change their ground transportation habits. Besides being a green transportation means, they offer a suitable response for part of inner-city transportation demand and counter the "last-kilometer problem". Several attempts had been tried before (see Shaheen *et al.* [1] and DeMaio [2] for a study on this means of transportation over time) but it is only recently that they have met success. We can rely this recent success on the development of new Information and Communication Technologies (ICT), which play an central role in their working.

A self-service bike sharing system is usually made of stations spread over a geographic area, in general a city or a town. In each station, there are racks at which bikes can be parked. To rent a bike, users have to identify themselves by using smart technologies such as mobile phone, membership card, or bank card – the list is not exhaustive. They are charged for the time they use the bike. If the bike is not parked at a rack after some while, a punitive cost is charged to the identified user.

Apart from vandalism, one of the main issues faced by operators of such systems is the maintaining of a right number of bikes at each station. Not enough bikes at a station may eventually lead to unsatisfied users, who would like to find a bike; too many bikes at a station may eventually lead to users not finding any available rack to park their bike. This issue is called the *imbalance*. In general, fleets of vehicles are used to move bikes from stations with an excess in bikes to stations with a default in bikes. However, there is a real need for improving the way these fleets are operated, see [3] or [4] for Vélib', or [5] for Villo (the system in Bruxelles). There is even a website dedicated to the imbalance of the Villo system in Bruxelles[1], Belgium.

1.2. **Objective and results of the present work.** The present paper aims to study ways for reducing the imbalance during the day. We call this reducing task the *dynamic regulation problem* because the impact of the users is not negligible, in contrary to what happens during the night and for which the corresponding problem is a *static regulation problem*. Note moreover that the congestion of the streets makes the dynamic regulation problem even more difficult in practice. Except if the operator decides to use a huge fleet of vehicles – a choice never met in concrete cases –, the impact of the regulation on the efficiency of the system

Key words and phrases. complexity; dynamic routing; pricing; self-sevice mobility system; simulation.

[1] `http://www.wheresmyvillo.be`

is necessarily limited. For instance, in Paris, there are currently about $110'000$ bikes daily moved by the users [6], a number which has to be compared with the 25 vehicles of the fleet dedicated to the regulation, each of these vehicles with a capacity of 20 bikes, except two with a capacity 62, which gives a total capacity of 584 bikes.

Our results are varied. As theoretical results, we show that it is an NP-hard problem to choose at any moment the best decision for the vehicles, see Section 3.1 for the formal framework and Section 3.2 for the proof. In Section 2.4, we discuss the impact of moving a bike: the gain in satisfied users gan be larger than 1. NP-hard problems to be solved in real-time are generally addressed by heuristics. We do not depart from this rule and propose some heuristics for the management of a unique vehicle, see Section 3.3. Indeed, we assume, as it is often the case in practice, that the city is divided into areas, each of these areas being assigned to a distinct vehicle. Such division into areas allows to work with each vehicle independently. It avoids coordination issues. We propose also a pricing strategy in Section 4 that makes possible to reduce the imbalance without the use of a fleet of vehicles moving the bikes. Note that such a pricing strategy could be particularly useful for self-service car sharing systems, as the Autolib' system launched recently in Paris. All these methods are evaluated in Section 6 with the help of a versatile simulator we developped with realistic assumptions on the user behaviors. This simulator described in Section 5 is open-source and allows rather easily the experiments of new fleet heuristics or pricing strategies. We conclude with open problems. We state also some concrete and elementary recommandations, which may seem natural, but which are confirmed by our study.

1.3. **Related works.** Bike sharing problems are relatively new, but there is already a substantial literature, addressing them from points of view varying from geography to operations research. We are not able to cite all of them here and we restrict our survey on papers dealing with optimization issues. The paper by Lathia, Ahmad and Capra [7], adopting a statistical approach to discuss the performances of existing systems, starts with a comprehensive survey of the literature on bike sharing systems.

Lin and Yang [8] addresses these systems from a strategic point of view: they propose a model determining the best locations for the stations, their capacities, and the total number of bikes, while taking into account the interests of both the users and the investors. George and Xia [9] model bike sharing systems as closed queuing networks and derive methods for determining the right number of bikes. Fricker and Gast [10] conduct an accurate analysis of the behavior of the closed network for a special case.

The static regulation problem is addressed by Raviv *et al.* [11]. Their objective is to find the best repositioning of the bikes that can be achieved by several vehicles within time limits. The satisfaction function introduced by Kolka and Raviv [12] is used to evaluate the quality of a repositioning. The same problem is also addressed by Benchimol *et al.* [13] and Chemla *et al.* [14]. They use more or less the same framework with only a single vehicle used for the regulation, the first paper being focused on theoretical questions such as polynomial approximation algorithm or special polynomial cases while the second one is focused on the computation of lower bounds and good feasible solutions via local search. Both aspects – service level requirement and bike repositioning – are combined in a recent paper by Schuijbroek *et al.* [15].

The dynamic regulation problem is addressed by Rousseau *et al.* [16]. They assumed that the future demands in bikes for each station is perfectly known over a short-term horizon and derive an exact model for minimizing the total imbalance of the system. Lower and upper bounds are computed via column generation and Bender's decomposition. Waserhole and Jost [17] develop a pricing strategy to capture the part of the demand leading to the best behavior for the system.

2. Formalisation

2.1. **Transport system.** The system is modelled by a complete directed graph $D = (V, A)$, where $V = \{1, \ldots, n\}$ is the vertex set, the vertices being the stations. An arc (i, j) models the shortest path in the city from i to j. Each vertex i has a capacity C_i in bikes, which can be interpreted as the number of racks. The time needed by a bike (resp. by a vehicle, resp. by walking) to go from vertex i to vertex j is a random variable T^b_{ij} (resp. T^v_{ij}, resp. T^w_{ij}). The total number of bikes is denoted by N. A vehicle is assumed to have capacity Q. The time needed to load or unload bikes at a vertex is assumed to be negligible.

2.2. **Demand.** The users are assumed to arrive independently at a vertex i according to a Poisson process of parameter $\lambda_i \in \mathbb{R}_+$. Each user arriving at vertex i has a destination vertex j drawn at random with probability p_{ij}. Note that the matrix $(\lambda_i p_{ij})_{i,j\in V}$ can be interpreted as the usual O-D matrix.

If the user finds a bike at vertex i, he starts its trip from i to j. Otherwise, the user chooses to walk to a vertex k having bikes, assuming that once he is in the system, he knows perfectly the state of the whole system. Moreover, he chooses a k minimizing the following quantity

$$\textsc{priceWalk} \times \mathbb{E}(T^w_{ik}) + \textsc{priceBike} \times \mathbb{E}(T^b_{kj}),$$

where priceWalk is the disutility of walking during one unit of time, while priceBike is the disutility of biking during the same unit of time. Usually, these quantites are different. If the user does not eventually find a bike at vertex k when he arrives, he repeats the same process.

When a user arrives at his destination vertex j, he leaves his bike only if he is able to find an available rack. Otherwise, the user chooses to bike to another vertex with at least one available rack, and among such vertices, he chooses the vertex k minimizing

$$\textsc{priceBike} \times \mathbb{E}(T^b_{jk}) + \textsc{priceWalk} \times \mathbb{E}(T^w_{kj}).$$

To avoid having users endlessly roaming through the city, two bounds are given for these two exploration phases. When he looks for a bike, a user has a maximum number of vertices he is willing to explore and a limit on the time this exploration can last, after what he leaves the system unsatisfied. A similar two-bound limit is set for the exploration phase in the case he does not find an available rack. In the case a user leaves the system because he did not succeed in finding an available rack within the limits, the bike is "lost". These two bounds are introduced with the intention of having a realistic system. Assuming that the users leave the system after some while because they do not have find any bike is clearly realistic: in such a situation, users will opt for another transportation means. Lost bikes are not completely unrealistic either: for instance in Paris, none of the current available bikes are the original ones [18]. Moreover, these bounds make some users satisfied and some other not satisfied; they allow therefore to measure the efficiency of a system, see Section 2.3.

Additional behaviour assumptions when there is a pricing strategy. In the case when there is a pricing strategy, we have at time t a price $c_j(t)$ attached at each vertex j, which is paid by a user when he parks his bike at vertex j. When a user starts a trip from a vertex i to a destination vertex j at time t, he chooses an intermediate target vertex k that minimizes the quantity

$$\textsc{priceBike} \times \mathbb{E}(T^b_{ik}) + \textsc{priceWalk} \times \mathbb{E}(T^w_{kj}) + c_k \left(t + \mathbb{E}(T^b_{ik})\right). \tag{1}$$

In our work, we require that the prices remain constant over time windows, to avoid too many changes in the prices. Note that two users arriving during a same time window at a vertex pay the same amount of money, independently of their origin.

Practically, we can imagine that a user arriving in the system knows all the prices for all vertices in the forthcoming hour.

This framework is different from the one of Waserhole and Jost [17], in which the users do not change their destination but decide whether they realize the trip regarding its price.

2.3. **Quality of the management.** In the present subsection, we briefly describe how to evaluate and compare management strategies and algorithms. We emphasize that in our model, the demand is inelastic: the parameters λ_i and p_{ij} are independent of the quality of the management strategies. It is of course an approximation, which is classical in such contexts, especially in transportation science (for instance, basic trafic assignment models assume inelastic demand). Global modelling is more or less out of reach. Think for instance about the four steps model in transportation: the last one – traffic assignement – which has received a lot of attention – works often with a predetermined demand.

Roughly, we measure how many users have not found a bike, and how many users have not found an available rack to leave their bike. The sum of these two quantites is the number of *unsatisfied* users. The other users are *satisfied*. In other words, a user is said to be satisfied when he manages to rent a bike at a station and park it at a station. If he roams to find a bike or a parking within his patience limits, he is still considered as satisfied.

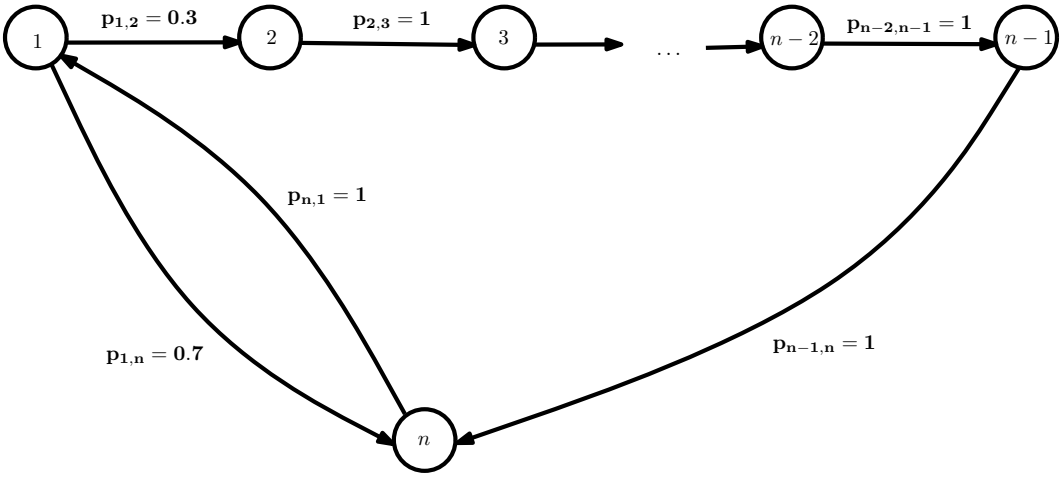

FIGURE 1. With $\lambda_i = 1$ for $i = 1, \dots, n-1$ and λ_n, moving a bike from vertex n to vertex 1 makes $0.3(n-1) + 0.7$ users satisfied

If there is a regulation strategy using vehicles, the efficiency of the strategy could be measured comparing the number of satisfied users obtained with this regulation with the number of them without regulation. The gain in satisfied users divided by the total number of bikes moved by the vehicles is the impact of moving a bike. Moving a bike in a right way may in general satisfy at least an additional user, who will be able to find a bike, and even two additional users if moving the bike vacates a rack in a full station. Moreover, the bike moved by the vehicle may eventually arrive in some part of the city where the demand is high and where it will satisfy many users. We see that the impact of moving a bike can be quite high and that the choice of the vertex on which the bike is taken can have a non-negligible impact. A further study of this point is provided in the next section.

2.4. **Optimal positioning of a bike.** We discuss a little bit further the possible impact of moving a bike with a vehicle. We start with an example.

Assume that we have the following system with one bike and one vehicle, see Figure 1. The vertex set is $V = \{1, \dots, n\}$, the parameters are $N = 1$, $\mathbb{E}(T^b_{ij}) \gg \mathbb{E}(T^v_{ij})$ for all $i, j \in V$, $\lambda_i = 1$ for $i = 1, \dots, n-1$ and $\lambda_n = 0$, and

$$(p_{ij})_{i,j\in V} = \begin{pmatrix} 0 & 0.3 & 0 & \cdots & 0 & 0.7 \\ \vdots & \ddots & 1 & \ddots & & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & & & & 0 & 1 \\ 1 & 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}.$$

There is no exploration process (all bounds for the exploration are set to 0).

Note that p_{n1} is set to 1 only to have $(p_{nj})_{j\in V}$ a true probability distribution over V. Since no user will ever arrive to n to take a bike, no user will make the trip n to 1.

The vehicle is initially at vertex n. As soon as the bike is arriving at vertex n, the vehicle brings it back to vertex 1, and comes back to vertex n. In average, when unloaded on vertex 1, the bike makes $0.3(n-1)+0.7$ users satisfied. The gain in satisfied users by the move of one bike can therefore be arbitrarily large.

Given a system with a network $D = (V, A)$, arrival rates $(\lambda_i)_{i\in V}$, and probabilities $(p_a)_{a\in A}$, a natural question is on which vertex the expected number of trips of a bike is the highest before being stucked in a

station. Consider $D' = (V, A')$ where

$$A' = \{(i,j) \in A : \lambda_i p_{ij} \neq 0\}.$$

If there is a vertex i such that there is no path in D' from i to a *sink* – a vertex j with no leaving arc –, then the bike will never stop moving, and the expected number of trips is infinite. Therefore the question makes sense only under the following assumption.

Assumption 1. In D', each vertex has a path from itself to a sink.

We answer completely the question in the following Theorem 2.2. The main objects for computing the expected number of trips are the matrix $Q = (q_{i,j})_{i,j\in V}$ and the vector $\boldsymbol{e} = (e_i)_{i\in V}$ defined by

$$q_{ij} = \begin{cases} p_{ij} & \text{if } \lambda_i \neq 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad e_i = \begin{cases} 1 & \text{if } \lambda_i \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Before stating and proving Theorem 2.2, we need the following technical lemma, whose proof is postponed at the end of the section. Denote by n the number of vertices and by I_n the $n \times n$ identity matrix.

Lemma 2.1. *Under Assumption 1, the matrix $I_n - Q$ is nonsingular.*

Now, we are in position to state and prove the main result of this subsection.

Theorem 2.2. *Under Assumption 1, the expected number of trips of a bike put on vertex i before being stucked in a sink is the ith component of $(I_n - Q)^{-1}\boldsymbol{e}$.*

The assumption on D' may seem to be too restrictive to model some realistic situation. However, if there are stations with very low arrival rate, they can be considered as sinks.

Proof of Theorem 2.2. Define the following sequence $(\boldsymbol{h}^{(k)})_{k\geq 0}$ of elements of $\mathbb{R}^V$ by

$$\boldsymbol{h}^{(0)} = \boldsymbol{e} \quad \text{and} \quad \boldsymbol{h}^{(k)} = \boldsymbol{e} + Q\boldsymbol{h}^{(k-1)} \text{ for } k \geq 1.$$

For a bike initially located at vertex i, the ith component $h_i^{(k)}$ of $\boldsymbol{h}^{(k)}$ can be interpreted has its expected number of trips if we stop the system as soon as the bike is stucked in a sink or has made $k+1$ trips. If the limit of $\boldsymbol{h}^{(k)}$ exists when k goes to infinity, the ith component of the limit is the expected number of trips when the bike is located initially at vertex i.

We have $\boldsymbol{h}^{(k)} = \sum_{\ell=0}^{k} Q^\ell \boldsymbol{e}$ and thus $\boldsymbol{h}^{(k)} - (I_n - Q)^{-1}\boldsymbol{e} = (Q - I_n)^{-1}Q^{k+1}\boldsymbol{e}$. Therefore, we have

$$\|\boldsymbol{h}^{(k)} - (I_n - Q)^{-1}\boldsymbol{e}\| \leq |\!|\!|(Q - I_n)^{-1}|\!|\!| \cdot |\!|\!|Q^{k+1}|\!|\!| \cdot \|\boldsymbol{e}\|$$

Since the Perron-Frobenius theorem ensures that $\lim_{k\to+\infty} Q^k = 0$, we get that $\boldsymbol{h}^{(k)}$ converges towards $(I_n - Q)^{-1}\boldsymbol{e}$. □

We can also formulate questions in which the time is taken into account. For instance, on which initial vertex the bike must be put in order to maximize asymptotically the mean number of trips by unit of time? The answer in this case is a direct consequence of the convergence property of Markov chains: on any irreductible part, the mean number of trips does not depend on the initial vertex.

Proof of Lemma 2.1. Suppose for a contradiction that there is a nonzero eigenvector $\boldsymbol{g}$ of Q with eigenvalue 1. Define $g_{\max}$ as the maximal component of $\boldsymbol{g}$. Without loss of generality, we can assume that $g_{\max} > 0$. Denote by J the indices i such that $g_i = g_{\max}$. The submatrix Q' of Q defined by $Q' = (q_{ij})_{i,j\in J}$ is stochastic.

Indeed, for $i \in J$, we have $\lambda_i > 0$ using the equality $g_i = \sum_{j=1}^{n} q_{ij}g_j > 0$ and the definition of q_{ij}. Since

$$g_i = \left(\sum_{j=1}^{n} p_{ij}\right) g_{\max} \geq \sum_{j=1}^{n} p_{ij}g_j = \sum_{j=1}^{n} q_{ij}g_j = g_i,$$

we have $p_{ij} = 0$ for j such that $g_j < g_{\max}$.

As Q' is stochastic, there is no path from any vertex in J to a sink in D'. It is in contradiction with Assumption 1. □

3. Operating with one vehicle

3.1. **Formal framework.** A theoretical discussion on the dynamic regulation problem with one vehicle requires a formal definition of problem. To fix a framework, we add the following constraint.

Constraint on vehicle behavior. *The vehicle can start a new action only when it is on a vertex. In other words, when it is traveling, no modification on its travel can be taken.*

With this additional constraint, the only thing missing in the model given in Section 2 is the exact expression of an objective function. To simplify the discussion, we can assume that the capacity Q of the vehicle and the capacities C_i of the vertices are all infinite, and that a user not finding a bike disappears without any roaming.

The most natural problem can then be formalized as follows.

1-vehicle dynamic regulation problem: best mean

Input. A complete directed graph $D = (V, A)$, arrival rates $(\lambda_i)_{i\in V}$, probabilities $(p_{ij})_{i,j\in V}$, exponential travel times $(T^b_{ij})_{i,j\in V}$ and $(T^v_{ij})_{i,j\in V}$, a total number N of bikes.

Output. A *policy*, which determines for each state of the system an action the vehicle has to perform, such that the mean number of users finding a bike per unit of time is maximal.

Note that since the travel times follow exponential laws, the system is Markovian.

By elementary properties of Markov Decision Process, there is no harm in assuming that the policy is deterministic. A state is fully described by the number of bikes in each vertex i, the number of bikes traveling from i to j for each pair of vertices i, j, the position of the vehicle, and the number of bikes it carries. We can associate to each such state an action (the *policy*) such that performing this action each time the vehicle arrives at a vertex maximizes the mean number of users finding a bike per unit of time.

The precise description of the policy seems to be out of reach. The heuristics described in Section 3.3 are a way to solve approximatively this problem. From a complexity point of view, it is not clear how to deal with such a problem since the policy itself can be an algorithm. However, we can propose a variant of it for which the complexity discussion is tractable.

1-vehicle dynamic regulation problem "catching first users"

Input. A complete directed graph $D = (V, A)$, arrival rates $(\lambda_i)_{i\in V}$, constant travel times $(t^b_{ij})_{i,j\in V}$ and $(t^v_{ij})_{i,j\in V}$, a number N_i of bikes in each vertex i, the vertex on which the vehicle is initially located, a number m.

Output. A *tour* of the vehicle that maximizes the probability that the first m users find a bike.

Here, a *tour* is the sequence of vertices the vehicle intends to visit, with the number of bikes loaded or unloaded at each visit, as well as possible updates when users take bikes.

3.2. **Complexity.**

Theorem 3.1. *The 1-vehicle dynamic regulation problem "catching first users" is NP-hard.*

Proof. Let $G = (V, E)$ be a graph, and let $n = |V|$. Add a new vertex o and link it to all vertices. It defines a new graph $G' = (V', E')$ with $V' = V \cup \{o\}$ and $E' = E \cup \{(o, v), v \in V\}$. Assume that the vehicle is initially located at o. We set $N_o = n$ and $N_i = 0$ for all $i \in V$. All λ_i are assumed to be equal to the same value λ, except λ_o equaling 0. It takes exactly one unit of time for the vehicle to traverse any edge of the graph G', i.e., $t^v_{ij} = 1$ for all $ij \in E'$. We consider the complete directed graph on the same set of vertex $D = (V, A)$, where the arcs model the shortest path in G'. We are going to show that if we were able to solve on D the 1-vehicle dynamic regulation problem "catching first users" in polynomial time for $m = 1$, we were able to decide in polynomial time whether G has an Hamiltonian path.

Suppose there is an Hamiltonian path in the graph G, then, following this path, the vehicle can unload 1 bike at each time step. The probability that the first arrival time τ is lower or equal to any $t \in \mathbb{R}_+$ is $\mathbb{P}[\tau \leq t] = 1 - e^{-n\lambda t}$. If the first user arrives before 1 ($\tau < 1$), he does not find any bike. If $1 \leq \tau < 2$, the vehicle has only time to visit one station. So in $n-1$ out of n cases, he does not find any bike. Repeating the argument, we obtain that the probability for the first user not finding a bike, while following the Hamiltonian chain, is

$$P(n, \lambda) = \sum_{k \geq 0}^{n-1} \alpha_k \beta_k \tag{2}$$

where

$$\alpha_k = \frac{n-k}{n} \quad \text{and} \quad \beta_k = e^{-n\lambda k} - e^{-n\lambda(k+1)} \quad \text{for all } k = 0, \ldots, n-1.$$

α_k is the ratio of unvisited vertex at time step k and β_k is the probability for the first arrival of a user to occur between k and $k+1$.

If the vehicle does not follow an Hamiltonian path, then the vehicle cannot visit a new vertex at each time step. With the same lines of reasoning, we get a probability $P'(n, \lambda) = \sum_{k=0}^{\infty} \alpha'_k \beta_k$ with $\alpha'_k \geq \alpha_k$ for all k and $\alpha'_{k_0} > \alpha_{k_0}$ for at least one k_0. Thus $P'(n, \lambda) > P(n, \lambda)$.

We have then proven that there is an Hamiltonian path in G if and only if the minimum value of the probability for the first user not finding a bike is equal to $P(n, \lambda)$. The conclusion follows, as the problem of deciding whether there is an Hamiltonian path in a graph is NP-complete. □

Theorem 3.1 shows that, when m is not fixed, the 1-vehicle dynamic regulation problem "catching first users" is a difficult problem. The proof shows actually that the case $m = 1$ is NP-hard. The proof can easily be adapted in order to get that the NP-hardness for any fixed m, since the best strategy is then again clearly for the vehicle to follow an Hamiltonian path and to unload m bikes at each vertex. Of course, we assume that the capacity of the vehicle is big enough.

If we want to deal with fixed capacities on the vertices or on the vehicle, things become more difficult. We were not able to prove some general result. The following is a first special case.

Proposition 3.2. *Assume that the vertices have unit capacities. Then the* 1*-vehicle dynamic regulation problem "catching first users" (with capacities) is NP-hard for $m = 1$ and for $m = 2$.*

We prove first a technical lemma. A *complete vertex* in a graph is a vertex having any other vertex as neighbor.

Lemma 3.3. *The problem of deciding whether a graph G has an Hamiltonian path with an endpoint being a complete vertex is an NP-complete problem.*

Proof. Add to G a new vertex v and link it to all vertices of G. It gives a new graph G' whose vertex v is complete. We check that G has an Hamiltonian path if and only if G' has an Hamiltonian path whose endpoint is a complete vertex.

If G has an Hamiltonian path, then clearly this Hamiltonian path followed by v is an Hamiltonian path of G' with an endpoint being a complete vertex. Conversely, suppose that G' has such an Hamiltonian path P and let u its complete endpoint. If P has v as an endpoint, then clearly $P \setminus v$ is an Hamiltonian path in G and we are done. If P has not v as an endpoint, one can delete from P an edge incident to v and add an edge incident to u making P again an Hamiltonian path. Call this new Hamiltonian path P'. The path $P' \setminus v$ is an Hamiltonian path of G.

The conclusion follows since it is NP-complete to decide whether there is an Hamiltonian path in a graph. □

Proof of Proposition 3.2. The case $m = 1$ is actually proven in the proof of Theorem 3.1. Let us now deal with the case $m = 2$. Take the same graphs G' and D as in the proof of Theorem 3.1. Assume that the vehicle, located at o, carries $n + 1$ bikes. If there is in G an Hamiltonian path P with an endpoint being a complete vertex, then the best strategy consists in following P. Indeed, the best the vehicle can do is to unload one bike at each time step at a new vertex. If the first user arrives at a vertex already visited by the vehicle, the condition on P ensures that there is an Hamiltonian path on the remaining vertices and the

now empty vertex. If the first user arrives after the vehicle has visited all the vertices, the vehicle stays at the last vertex, which is a complete vertex. Once the first user has shown up, the vehicle goes to the now empty vertex in one time step where it unloads a bike.

If there is no such Hamiltonian path in G, following the same lines as in the proof of Theorem 3.1, we get a strictly higher probability for the first two users both not finding a bike. If we were able to maximize in polynomial time the probability for the two first users to find a bike, we would be able to decide in polynomial time whether there is in G an Hamiltonian path with an endpoint being a complete vertex. The conclusion follows from Lemma 3.3. □

We conjecture that Proposition 3.2 can be extended for any fixed $m > 2$, but we were not able to do it.

3.3. **Heuristics.** Heuristics for managing a vehicle are based on the following ideas. Each vertex i has a *target state* θ_i (number of bikes). Moreover, the vehicle gets as mission either a vertex to visit, or a pair of vertices to be visited in a given order, without any information about the number of bikes it has to load or unload. This number is decided by the vehicle when it arrives at a vertex, and it takes the "best local decision" with respect to its current load and the current load of the vertex. The "best local decision" consists in unloading or loading bikes in order to bring the vertex to the closest possible state to its target state.

Once a mission of the vehicle is finished, the heuristic is run again and the vehicle gets a new mission to perform.

The *vertex the most in excess* is denoted i^+ and is such that

$$i^+ = \text{argmax}_{i \in V}(x_i - \theta_i). \tag{3}$$

In case there are several such vertices, select the closest to the vehicle. The *vertex the most in default* is denoted i^- and is such that

$$i^- = \text{argmax}_{i \in V}(\theta_i - x_i). \tag{4}$$

In case there are several such vertices, select the closest to the vehicle.

The *one-step heuristic*, denoted 1SH, consists in sending the vehicle to i^+ when the load of the vehicle is under a threshold ρ, and to i^- when the load of the vehicle is above the threshold ρ. When the vehicle arrives at this vertex, it takes the best local decision.

The *two-step heuristic*, denoted 2SH, is similar, but instead of deciding only the next vertex to visit for the vehicle, it decides the two next. When the load of the vehicle is under the threshold ρ, the vehicle is first sent to i^+ and then to i^-. When the load of the vehicle is above the threshold, the vehicle is first sent to i^- and then to i^+.

These two heuristics do not use at all the informations given by the λ_i and the p_{ij}. They only use the current state of the system. It may be useful to try to take such informations into account. The following heuristics are built in this spirit. They replace the current state x_i by an estimated one, $\widetilde{x}_i$, computed as follows, where j is the current vertex of the vehicle.

$$\widetilde{x}_i = \min\left(C_i, \max\left(0, x_i + \alpha \mathbb{E}(T^v_{ji})(\mu_i - \lambda_i)\right)\right) \tag{5}$$

where $\mu_i = \sum_{k \in V} \lambda_k p_{ki}$. The quantity μ_i can be interpreted as an estimation of the mean number of bikes arriving to i per unit of time in the stationary state of the system. It is actually an approximation since bikes are leaving a vertex k in order to go to vertex i only if the vertex k is nonempty. Similarly, we can see λ_i as the mean number of bikes leaving i per unit of time in the stationary state of the system. Assuming these interpretations for μ_i and λ_i, the quantity $\widetilde{x}_i$ is a correction of the number of bikes x_i by taking into account the future arrivals and departures of bikes due to the activity of the users. The parameter α is used to weaken the weight of the future, since it is a very rough estimation.

The *one-step heuristic with forecast*, denoted 1SHF, is exactly the one-step heuristic above, 1SH, once the x_i have been replaced by the $\widetilde{x}_i$.

The *two-step heuristic with forecast*, denoted 2SHF, looks for the "best" ordered pair of distinct vertices (i_1, i_2). For each such ordered pair (i_1, i_2), both different of j, the heuristic computes an estimated total number of bikes moved by the vehicle. At each vertex, the vehicle is supposed to take the best local decision.

The estimated number of bikes in i_1 is $\widetilde{x}_{i_1}$, as computed by Equation (5). The estimated number of bikes in i_2 is $\widetilde{x}_{i_2}$, once T^v_{ji} has been replaced by $T^v_{ji_1} + T^v_{i_1 i_2}$.

The *two-step heuristic one-stop with forecast*, denoted 2S1SHF, selects exactly in the same way the pair (i_1, i_2) but the mission of the vehicle is reduced to i_1.

4. Operating through dynamic pricing

As for the vehicle strategy, we suppose that each vertex i has a target number of bikes θ_i.

Let us define

$$\ell_{ijk} := \text{priceBike} \times \mathbb{E}(T^b_{ik}) + \text{priceWalk} \times \mathbb{E}(T^w_{kj}) - \text{priceBike} \times \mathbb{E}(T^b_{ij}).$$

The quantity ℓ_{ijk} models the "effort" provided by a user choosing an intermediate target vertex k on which he leaves his bike instead of going directly from i to j. It assumes the behavior described in Section 2.2.

Note c_i the price paid by a user leaving his bike at vertex i. Then the numbers y_{ijk} of users per unit of time wishing to do a trip from i to j but prefering to leave their bikes at vertex k satisfy

$$\begin{cases} \displaystyle\sum_{k\in V} y_{ijk} = \lambda_i p_{ij} \quad i,j \in V \\ \qquad y_{ijk} \geq 0 \quad \text{for } i,j,k \in V \text{ such that } k = \operatorname{argmin}_{k'\in V} \ell_{ijk'} + c_{k'} \\ \qquad y_{ijk} = 0 \quad \text{for } i,j,k \in V \text{ such that } k \neq \operatorname{argmin}_{k'\in V} \ell_{ijk'} + c_{k'}. \end{cases}$$

The pricing problem consists in finding prices c_i for $i \in V$ such that a solution $(y_{ijk})_{i,j,k\in V}$ of this system satisfies $\sum_{i,j\in V} y_{ijk} = \theta_k - x_k$, where x_k is the current number of bikes at vertex k. Since we may have $x_k > \theta_k$, we replace $\theta_k - x_k$ by $\widetilde{\theta}_k$, defined by

$$\widetilde{\theta}_k = \frac{\sum_{i,j\in V} \lambda_i p_{ij}}{\sum_{j\in V} \theta_j} \max(0, \theta_k - x_k).$$

The coefficient $\frac{\sum_{i,j\in V} \lambda_i p_{ij}}{\sum_{j\in V} \theta_j}$ is only used for normalization.

Therefore, the pricing problem amounts to find a solution $(y_{ijk})_{i,j,k\in V}$ of the following system.

$$\begin{cases} \displaystyle\sum_{i,j\in V} y_{ijk} = \widetilde{\theta}_k \quad k \in V \\ \displaystyle\sum_{k\in V} y_{ijk} = \lambda_i p_{ij} \quad i,j \in V \\ \qquad y_{ijk} \geq 0 \quad \text{for } i,j,k \in V \text{ such that } k = \operatorname{argmin}_{k'\in V} \ell_{ijk'} + c_{k'} \\ \qquad y_{ijk} = 0 \quad \text{for } i,j,k \in V \text{ such that } k \neq \operatorname{argmin}_{k'\in V} \ell_{ijk'} + c_{k'}. \end{cases} \tag{6}$$

Consider the following linear program

$$\begin{array}{ll} \max & \displaystyle\sum_{i,j\in V} \lambda_i p_{ij} \omega_{ij} + \sum_{k\in V} \widetilde{\theta}_k \mu_k \\ \text{s.t.} & \mu_k + \omega_{ij} \leq \ell_{ijk} \quad i,j,k \in V \end{array} \tag{7}$$

Proposition 4.1. *Let ω^*_{ij} et μ^*_k be the optimal solutions of the linear program* (7)*. Setting $c_k = -\mu^*_k$ for each vertex k ensures that the system of equations* (6) *has a solution.*

Proof. We want to prove that for such c_k's, program (6) has a solution. We consider the dual of program (7).

$$\begin{array}{ll} \min & \displaystyle\sum_{i,j,k\in V} \ell_{ijk} y_{ijk} \\ \text{s.t.} & \displaystyle\sum_{i,j\in V} y_{ijk} = \widetilde{\theta}_k \quad k \in V \\ & \displaystyle\sum_{k\in V} y_{ijk} = \lambda_i p_{ij} \quad i,j \in V \\ & y_{ijk} \geq 0 \quad \text{for } i,j,k \in V \end{array} \tag{8}$$

We will prove that the optimal solution y^*_{ijk} of program (8) is a solution of system (6).

The y^*_{ijk}'s satisfy all equations of the system (6), except maybe the last one.

Consider a pair $i, j \in V$. If $\lambda_i p_{ij} = 0$, then for all k we have $y^*_{ijk} = 0$, and all such y_{ijk}'s satisfy also the last equation. If $\lambda_i p_{ij} > 0$, then $y^*_{ijk} > 0$ for at least one k. By the complementary slackness, we have for such a k the equality $\mu^*_k + \omega^*_{ij} = \ell_{ijk}$. Hence $\omega^*_{ij} \leq \ell_{ijk} - \mu^*_k$ with equality when k is equal to $\arg\min_{k' \in V} \ell_{ijk'} - \mu^*_{k'}$, i.e. $\omega^*_{ij} = \min_{k' \in V} \ell_{ijk'} - \mu^*_{k'}$. Now, take a k such that $\ell_{ijk} - \mu^*_k$ is not minimum. Then $\omega^*_{ij} < \ell_{ijk} - \mu^*_k$ and still by complementary slackness, we have $y^*_{ijk} = 0$.

Therefore, setting $c_k = -\mu^*_k$ for each vertex k makes y^*_{ijk} a solution of the system (6). □

Note that the value ω^*_{ij} in the proof above is the cost experienced by users wishing to do a trip from i to j when the prices are set to $-\mu^*_k$.

5. OADLIBSim: a versatile simulator

In order to evaluate and compare different heuristics for managing the fleet regulating self-service bike sharing systems, we developped an open-source simulator. The source is available at the following address:

http://cermics.enpc.fr/∼meuniefr/home.html/OADLIBSim_Site

OADLIBSim is programmed in C++ using the discrete-events simulation framework OMNet++. It simulates exactly the system described in Section 2. All parameters can be freely set. Moreover, users are described by a common vector

[maxStBike,maxStPrk,maxTimBike,maxTimPrk,priceBike,priceWalk]

where

- maxStBike is the maximum number of vertices a user is willing to visit in addition to his origin vertex in order to find a bike before leaving the system.
- maxStPrk is the maximum number of vertices a user is willing to visit in addition to his destination vertex in order to find a place to leave its bike before leaving the system. If a user does not find a place, the bike disappears from the system.
- maxTimBike is the maximum time a user is willing to spend in order to find a bike before leaving the system.
- maxTimPrk is the maximum time a user is willing to spend in order to find a place to leave its bike before leaving the system.
- priceBike is the price of 1 second spent by a user with its bike. By default, this value is 1, which means that the reference is the time spent with a bike.
- priceWalk is the price of 1 second spent by a user by walking. For instance, if we set this value to 5, it means that a user experiences the same disutility by spending 5 seconds with a bike or 1 second by walking.

When a user explores vertices to find a bike, he stops as soon as one of the limit given by maxStBike and maxTimBike is reached. The same holds for maxStPrk and maxTimPrk.

If the maximum number of vertices is reached before, the user leaves the system without going up to this time limit. Again, if a user does not find a place, the bike disappears from the system. If the maximum number of vertices is reached before, the user leaves the system without going up to this time limit.

It is easy to add to the source a new method for managing the fleet: it is enough to implement a unique function computing the mission of a vehicle, taking as an input its C++-reference (in the case there are many vehicles). The function is included in the source, and any information about the current state of the system can be easily read with existing functions. Similarly, it is easy to add a new pricing strategy: it is enough to implement a unique fonction giving the prices for each vertex. In this function, one can again read informations on the state of the system. Moreover, the computation time of the function is simulated: the system is evolving during the time needed for each execution of the function.

An illustration of the interface is given on Figure 2.

At the end of a simulation, almost all possible informations we may like to have are available. It is for instance possible to know how many users needed to visit a given number of vertices $\leq$ maxStBike to find a bike, how many users needed to visit a given number of vertices $\leq$ maxStPrk to leave the bike, how long they take, ...

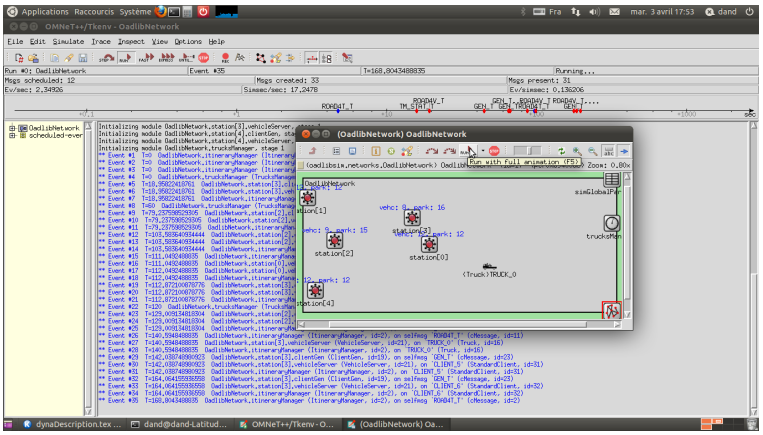

FIGURE 2. Interface: regulation using vehicles

6. EXPERIMENTAL RESULTS

6.1. **Instances.** In this section, the results obtained using all the former methods are given. The instances we used are available on the website whose address is given Section 5. They are chosen in order to have a good repartition of the vertices over the city space. The O-D matrix is built with respect to a gravity model, see Chapter 5 of the book by Ortúzar and Willumsen [19]. Four sizes are tested – with $20, 50, 100$, and 250 vertices. Moreover, in all cases, three different types of demand have been tested.

- In the "low demand case" a user is showing up at each station in average every 5 minutes
- In the "medium demand case" a user is showing up at each station in average every 2.5 minutes
- In the "high demand case" a user is showing up at each station in average every 1.5 minutes

When the system starts, all the bikes are parked at vertices. Parameters $T^b_{ij}, T^v_{ij}, T^w_{ij}$ are deterministic and such that

$$T^w_{ij} = 2.4T^b_{ij} \quad \text{and} \quad T^b_{ij} = 2.5T^v_{i,j} \quad \text{for all } i, j \in V.$$

There is only one vehicle of capacity 20, and all stations have capacity 20. We simulate only one profile of users such that

$$\text{MAXSTBIKE} = \text{MAXSTPRK} = 1, \text{MAXTIMBIKE} = 600 \text{ s}, \text{MAXTIMPRK} = 900 \text{ s},$$

$$\text{PRICEBIKE} = 1, \text{PRICEWALK} = 2.1.$$

Even if in a real system users would be willing to explore more vertices, especially to find a parking place, these limits are taken small in order to be able to compare the results of the different methods. When the simulation starts, no bike is in use. Four hours of the system are simulated. The regulation method starts after 30 minutes to evaluate its performances on a running system. Each simulation is replicated 10 times with different seeds.

All target states are set to $\theta_i = 14$. This number of bikes is also placed on each vertex, giving in total $N = |V| \times \theta_i$ bikes. The threshold ρ for the vehicle is set to 14 and the parameter α in Equation (5) is set to 0.1. In the pricing method, prices are updated every 15 minutes.

6.2. **Results.** Table 1 gathers results for the "low demand case", Table 2 for the "medium demand case", and Table 3 for the "high demand case". Each table is divided into four parts, each of them corresponding to a distinct number of vertices for D. Due to the size of the linear programs, we were not able to simulate the pricing method for 250 vertices.

The row "Satisfied" counts the percentage of satisfied users (see Section 2.3). The row "No bike" counts users having not found any bike. The row "No parking" counts users having not found any available rack for their bike (it leads to lost bikes). The row "Rejection" makes sense only when there is a pricing method. It stands for the users prefering to walk instead of taking a bike because of the prices.

The "Empty" column shows the results of the system without any regulation. "1SH" column stands for the one-step heuristic method. "1SHF" column gives the results of the one-step heuristic with forecast. "2SH" column corresponds to the results obtained by the two-step heuristic. "2SHF" and "2S1SHF"columns respectively stand for the two-step heuristic with forecast and the two-step one-stop heuristic with forecast. The last column shows the performances of the pricing method – without vehicle. Table 4 shows the ratio measuring the average number of satisfied users gained per bike moved.

Size	Status	Empty	1SH	1SHF	2SH	2SHF	2S1SHF	Pricing
	Satisfied	70	99	99	98	95	98	84
20	No bike	12	1	1	0	3	1	0
	No parking	18	0	0	2	2	1	0
	Rejection							16
	Satisfied	80	92	94	90	88	92	91
50	No bike	13	5	4	5	9	5	0
	No parking	7	3	2	5	3	3	2
	Rejection							7
	Satisfied	62	67	67	66	66	67	81
100	No bike	26	23	23	23	23	22	10
	No parking	12	10	10	11	11	11	0
	Rejection							9
	Satisfied	63	65	65	65	65	65	
250	No bike	26	25	25	25	25	25	
	No parking	11	10	10	10	10	10	

TABLE 1. Comparison between performances of the different methods in the low case demand

Size	Status	Empty	1SH	1SHF	2SH	2SHF	2S1SHF	Pricing
	Satisfied	59	91	93	85	82	88	80
20	No bike	31	7	5	10	15	10	4
	No parking	10	2	2	5	3	2	1
	Rejection							15
	Satisfied	75	84	86	82	81	83	86
50	No bike	19	13	11	14	16	14	7
	No parking	6	3	3	4	4	3	0
	Rejection							7
	Satisfied	50	54	55	54	54	54	69
100	No bike	41	38	37	38	38	38	22
	No parking	9	8	8	8	8	8	1
	Rejection							8
	Satisfied	46	47	47	47	47	48	
250	No bike	47	46	46	46	46	45	
	No parking	7	7	7	7	7	7	

TABLE 2. Comparison between performances of the different methods in the medium case demand

The largest ranges for the prices in the pricing method that are observed are

€7 for 20 vertices, €10 for 50 vertices, and €16 for 100 vertices.

To make the conversion, we have taken €8 = 1 hour. It is a reasonable conversion for the value of travel time in a Western city (see [20]).

Size	Status	Empty	1SH	1SHF	2SH	2SHF	2S1SHF	Pricing
	Satisfied	46	80	82	72	71	76	72
20	No bike	47	18	16	24	26	21	13
	No parking	7	2	2	4	3	3	1
	Rejection							14
	Satisfied	68	77	78	75	73	76	78
50	No bike	27	21	20	22	24	22	16
	No parking	5	2	2	3	3	2	0
	Rejection							6
	Satisfied	39	43	42	42	42	43	57
100	No bike	55	51	52	52	52	51	35
	No parking	6	6	6	6	6	5	1
	Rejection							7
	Satisfied	32	33	33	33	33	33	
250	No bike	63	63	63	63	63	63	
	No parking	5	4	4	4	4	4	

TABLE 3. Comparison between performances of the different methods in the high case demand

Size	Demand	1SH	1SHF	2SH	2SHF	2SH1SF
	Low	1.17	1.17	1.33	1.13	1.16
20	Medium	1.85	1.93	2.22	1.88	1.87
	High	3.11	3.22	3.58	3.30	3.08
	Low	1.23	1.24	1.38	1.03	1.27
50	Medium	1.80	2.20	1.94	1.83	1.77
	High	2.75	3.31	3.15	3.03	2.01
	Low	1.69	2.20	1.77	1.60	1.67
100	Medium	2.62	3.20	2.76	2.65	2.73
	High	4.00	4.08	4.07	3.78	3.90
	Low	2.31	2.92	2.37	1.89	2.28
250	Medium	1.50	4.30	3.35	3.11	3.17
	High	3.57	4.65	3.69	3.48	3.43

TABLE 4. Ratio users gained per bike moved for the different methods run on all instances

Remark on the "Rejection" row for the pricing method. In the simulations, we decide to use Equation (1) even for $k = i$. Indeed, the prices computed by Proposition 4.1 are determined, give or take an identical constant. Adding a same number to all μ_k^* still give an optimal solution of program (7). To decide a concrete price requires to consider the largest system including other means of transportation, the city and so on. This topic is out of the scope of this work, and would involve socio-economics technics. Using Equation (1) for $k = i$ as well leads to simulations that are independent of the choice of the constant.

6.3. **Discussion.** The results show that with no doubt adding regulation improve the system level of service. The method using a vehicle that obtains the best results is the one-step heuristic with forecast 1SHF. Let us go into details.

We compare the three heuristics 2SH, 2SHF, 2S1SHF. We see that 2SH is always better than 2SHF. It seems that taking into account the future is overwhelmed by the negative impact of fixing the next two vertices for the vehicle. This conclusion is consistent with the result for 2S1SHF, which is the best of the three. It fixes only the next vertex for the vehicle. When we look to the heuristics 1SH and 1SHF, and compare them to 2SH, 2SHF, 2S1SHF, we get a similar conclusion: very short-term decisions seem to be more efficient. However, taking into account the impact of the users during the moves of the vehicle, that is the informations provided by the λ_i, p_{ij}, and travel times, is positive.

The advantage 2S1SHF on 2SH is probably due to a higher number of bikes moved by the vehicle. Indeed, Table 4 shows that 2S1SHF has a ratio worst than 2SH while satisfying more users. This undelines the greater reactivity of 2S1SHF over 2SH.

Finally, we should tell that in the first version of our heuristics, the missions of the vehicle were described in terms of vertices to visit, and also in terms of number of bikes to load and unload. The results were always worse than the heuristics described only in terms of visited vertices, as done in the present work. It shows the importance of the short-term decisions (and maybe also of the weakness of the way we take into account the future).

When the network is too big, one vehicle is clearly not enough, especially in the case of high demand. This enhances the idea of having the city divided into clusters with a reasonable number of vertices in each of them, and having a vehicle devoted to each cluster. For such a division, the way of dealing the exchanges of bikes between the cluster has to be studied.

The pricing method seems to be particularly promising. First, when the size becomes larger, this method is more and more efficient when compared to the regulation with a vehicle, and becomes clearly better when we deal with 100 vertices. Second, it is reasonable to take the "No parking" numbers as the main objective for such a system, or at least as an important component in the evaluation: The consequence of not finding a rack to leave his bike has clearly more negative consequence than not finding a bike. The pricing method seems to be efficient to maintain the "No parking" number at a low level.

7. Conclusion

We can derive concrete recommendations from this work.

- Short-term methods are more efficient than medium-term methods. This is due to the limited impact of the vehicle.
- Regulation can be improved by knowing the parameters of the system (λ_i, p_{ij}, T^b_{ij}, T^v_{ij}): the heuristics taking into account their values have better impact than the others (2S1SHF is better than 2SH; 1SHF is better than 1SH). In practice, these parameters have to be estimated. It underlines the importance of having good statistics.
- The pricing strategy seems to be promising with respect to its results. However, it still needs to be addressed with socio-economical features. Moreover, operators attach importance to the *KISS – Keep It Simple & Stupid –* principle. To a certain extend, having prices which change regularly could be seen as not respecting this principle.

Several questions deserve future works.

The first one is the theoretical one concerning the complexity of the 1-vehicle dynamic regulation problem "catching first users" (with capacities) when the number of first users m is larger than or equal to 3, see Section 3.2. Its NP-hardness seems to be more or less obvious. Nevertheless, we were not able to write a proof of it.

For the heuristics with the vehicle, we see that there are many parameters that were arbitrarily set: θ_i (target state for vertex i), ρ (threshold for the vehicle used in some heuristics), and α, the weight of the future in the heuristics with forecast. Actually, we made some experiments with various values, but we were not able to find better values that the ones proposed in Section 6.1. However, it should be noted that our trials were more or less random since there is a lack of a practical model allowing to compute such parameters. Following the same lines, Equation (5) could be improved by making a better estimation of the number of bikes arriving and leaving a vertex per unit of time. In particular, it should take into account the "availability at station" i as defined by George and Xia [9], which is the percentage of time during which there is at least one bike at i.

Another open question is whether we can have efficient and robust heuristics for the multiple vehicle case that are not based on some *a priori* clustering.

References

[1] S. A. Shaheen, S. Guzman, H. Zhang, Bike sharing in Europe, the Americas, and Asia: Past, Present, and Future, Transportation Research Record 2143 (2010) 159 – 167.

[2] P. deMaio, Bike-sharing: History, Impacts, Models of Provision, and Future, Journal of Public Transportation (2009) 41–56.

[3] E. Bambaron, Vélib' peine à trouver un second souffle, Le Figaro.

[4] La gestion de vélib' a besoin d'une révision, Le Parisien.

[5] R.M., A la recherche désepérée de villo, La Libre Belgique.

[6] Vélib' et moi, http://blog.velib.paris.fr/blog/2011/06/30/la-regulation-succes-et-ambitions (Jun. 2011).

[7] N. Lathia, S. Ahmad, L. Capra, Measuring the impact of opening the London shared bicycle scheme to casual users.

[8] J.-R. Lin, T.-H. Yang, Strategic design of public bicycle sharing systems with service level constraints, Transportation Research Part E: Logistics and Transportation Review 47 (2011) 284 –294.

[9] D. K. George, C. H. Xia, Fleet-sizing and service availability for a vehicle rental system via closed queueing network, European Journal of Operational Research 211 (2011) 198–207.

[10] C. Fricker, N. Gast, Incentives and regulations in bike-sharing systems with stations of finite capacity, Tech. rep. (2012).

[11] T. Raviv, M. Tzur, I. A. Forma, Static repositioning in a bike-sharing system: Models and solution approaches, in: ODYSSEUS, 2009.

[12] T. Raviv, O. Kolka, Optimal inventory management of a bike-sharing station, Tech. rep. (2011).

[13] M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, L. Robinet, Balancing the stations of a self-service bike hire system, RAIRO-Operations Research 45 (2011) 37–61.

[14] D. Chemla, F. Meunier, R. Wolfler Calvo, Bike sharing systems: Solving the static rebalancing problem, Discrete Optimization.

[15] J. Schuijbroek, R. Hampshire, W.-J. van Hoeve, Inventory rebalancing and vehicle routing in bike sharing systems, Tech. rep. (2013).

[16] C. Contardo, C. Morency, L.-M. Rousseau, Balancing a Dynamic Public Bike-Sharing System, Tech. rep. (March 2012).

[17] A. Waserhole, V. Jost, Vehicle sharing system pricing regulation: Transit optimization of intractable queuing network, Tech. rep. (2012).

[18] P. Fay, Vélib', un système qui peut coûter cher, Europe 1.

[19] J. de Dios Ortúzar, L. G. Willumsen, Modelling Transport, Third Edition, Wiley, 2001.

[20] Y. Crozet, Le temps et le transport de voyageurs, Tech. rep., European Conference of Ministers of Transport (2005).

E-mail address: daniel.chemla@smart-flows.fr

E-mail address: frederic.meunier@cermics.enpc.fr

E-mail address: thomas.pradeau@enpc.fr

E-mail address: roberto.wolfler@lipn.univ-paris13.fr

E-mail address: houssame.yahiaoui@smart-flows.fr

D. Chemla and H. Yahiaoui, Smart Flows, 10 rue des Filles du Calvaire, 75003 Paris, France

F. Meunier and T. Pradeau, Université Paris Est, CERMICS, 6-8 avenue Blaise Pascal, Cité Descartes, 77455 Marne-la-Vallée, Cedex 2, France

R. Wolfler Calvo, Université Paris 13, Villetaneuse, France