A Novel Formulation for the Distributed Solution of Load Balancing Problems in Mobility On-Demand Systems

Francesco Acquaviva, Donato Di Paola, and Alessandro Rizzo

Abstract—In this paper, we present a novel optimization framework for the load balancing problem in mobility on-demand systems. The proposed approach aims to keep the system balanced and, at the same time, to maximize the number of accepted customers’ requests. Moreover, we propose the Distributed Load Balancing (DBL) algorithm that, given the customers’ requests, iterates towards feasible assignments that keep the system balanced. The performance of the proposed algorithm is assessed through an extensive simulation campaign in several operational conditions, by varying the number and locations both of the customers and of the vehicles, and considering different communication topologies among the pick-up and drop-off stations. Finally, a comparison of the performance of DBL with an optimal solver is provided.

I. INTRODUCTION

Recently, remarkable advancements in science and technology of intelligent transportation systems have raised the interest in the field of Mobility On-Demand (MOD) systems [1]. The principle underlying MODs is to provide customers with the opportunity of using a wide range of vehicles at any time, within an easy-to-use and affordable transportation service, through the sharing of electric or human-powered vehicles. The great success obtained by MODs resides in the possibility of obtaining the same benefits of privately-owned vehicles without requiring additional roads and parking spaces [2], in the advantage of paying only for the actual use of the car, without the fixed costs of maintenance, taxes and insurance related to car ownership, and in obtaining access to limited traffic zones. In many cities worldwide, bicycle and car sharing systems have been realized or designed [3].

Design, implementation, monitoring and control of MODs are indeed complex problems. Thus, to obtain an effective service for everyday mobility, a number of scientific and technological challenges must be addressed, involving all the actors in the system: vehicles, providers, customers, and not less important, the environment [4], [5]. Among the several issues related to MOD systems, one of the most important is the load balancing problem. Customers’ trips, by their nature, may yield some pick-up stations to run out available vehicles, while some drop-off stations may be full, and vehicles cannot be dropped at the desired location.

The remainder of this paper is organized as follows: in Section II the related work is reported. In Section III the mathematical formulation of the MOD system is provided. In Section IV the distributed optimization algorithm for load balancing is presented. Simulation results are provided and commented in Section V, and conclusions are summarized in Section VI.

II. RELATED WORK

Several MOD models oriented to the solution of the load balancing problem have been studied in the literature. In [1], a pricing strategy is proposed, through a solution that statically privileges paths that start from pick-up stations with low demand, close to the customer location, and end to drop-off stations with high-demand, close to the customer’s point of interest. The downside of this strategy lies in the fact that the pricing strategy is static, consequently, it does not take into account several factors related to urban mobility. In order to model a dynamical MOD system capable of modifying the pricing scheme based on external factors, in [6] a real-time pricing feedback control strategy, based on the main assumption that customers are sensitive to changes in the price of vehicles, is proposed. This strategy does not always guarantee system balance, since it does not take into account the possibility that the customer, motivated by the need to reach the desired destination, may not be sensitive to the increase of the cost of the vehicles. When a system becomes heavily unbalanced, i.e. there are several empty pick-up stations and/or full drop-off ones, it can be manually balanced by operators [7], or by self-driving vehicles, as suggested in [2]. However, these techniques tend to increase the overall cost of the MOD system infrastructure, without increasing neither the customers’ satisfaction, nor the providers’ profit.
III. Problem Formulation

A. Mobility-On-Demand System Model

We consider a set of \( m \) vehicles, with index set \( M = \{1, \ldots, m\} \), which can be driven by customers among \( n \) predefined stations, with index set \( V = \{1, \ldots, n\} \), located within a given urban area. Vehicles in \( M \) are distributed over stations in \( V \), i.e., a number \( b_i \in \mathbb{N}^+ \) vehicles (where \( \mathbb{N}^+ \) denotes the set of positive integers) are parked at each station \( i \in V \), where a maximum number of parking lots \( b_i^{\text{max}} \in \mathbb{N}^+ \) are available. We consider a set of \( w \) customers, with index set \( W = \{1, \ldots, w\} \). Each customer can pick up a vehicle from a departure station and can drop it off at a destination station. Thus, each customer \( k \in W \) is associated to a request that indicates the desired starting station and the destination station, \( i, j \in V \), respectively.

Assumptions III.1 (Request Assignment). We assume that each request from customers in \( W \), to travel from a station \( i \in V \) to a station \( j \in V \), has to be assigned univocally to one and only one vehicle \( l \in M \).

The core functionality of the proposed MOD system is to process the customer requests, while satisfying a set of given constraints, in order to maximize the revenue of the provider of the MOD service, yet satisfying the customer. Thus, a customer request can be accepted as it is, or modified and repurposed to the customer. In practical terms, a customer \( k \) who wants to travel from point A to point B selects a departure station \( i \), which is close to point A, and an arrival station \( j \), close to point B. Then, the MOD system may accept the request as it is or it can propose to modify the destination station from station \( j \) to station \( j^* \), close enough to the original destination station, offering a discount for the acceptance of the alternative trip. Moreover, we consider that customers are allowed to request two types of trip: one-way and round-trip. Round-trips evidently keep the system balanced, thus, one of the alternatives that the management system may propose to the customer is to accept a round-trip instead of a one-way one.

The assignment criteria of the MOD management system are based on the concept of profits, \( P_{lki} \in \mathbb{R} \), and costs, \( C_{lki} \in \mathbb{R} \), for both vehicles and customers.

Profits: We define the overall profit as \( P_{lki} = p_{li}^v + p_{ki}^c \), where \( p_{li}^v = f(b_i, b_j) \) represents the profit associated to vehicle \( l \in M \) that travels from station \( i \in V \) to station \( j \in V \), as a function of the number of vehicles in both pick-up and drop-off stations. The value \( p_{li}^v \) is higher when the vehicle executes trips that tend to balance the system. We define \( d(i, j) \) as the Euclidean distance between the pick-up station \( i \in V \) and the drop-off station \( j \in V \), and \( d_k(j^*, j) \) as the additional distance, for customer \( k \), between the drop-off station \( j^* \) in which he/she was routed by the system and his/her preferred station \( j \). Thus, \( p_{ki}^c = f(d_k(j^*, j)) \) represents the profit associated to customer \( k \in W \), which depends on the importance (in terms of the customer interest or his proximity to the point of major interest) of the visited drop-off station \( j \in V \) starting from the pick-up station \( i \in V \), i.e., if \( d_k = 0 \) the customers’ satisfaction is maximum.

Costs: We define the overall cost as \( C_{lki} = c_{li}^v + c_{ki}^c \), where \( c_{li}^v = f(d(i, j), \alpha, \chi) \) is the cost associated to vehicle \( l \in M \) traveling from station \( i \in V \) to station \( j \in V \). Cost \( c_{ki}^c \) is a function of the distance \( d(i, j) \) between the pickup and the drop-off station, of a variable quantifying traffic congestion along the selected route, \( \alpha \), and of a parameter that quantifies the vehicle characteristics, \( \chi \) [8]. On the other hand the cost \( c_{ki} = f(d(i, j)) \) associated to customer \( k \in W \) to travel from \( i \in V \) to station \( j \in V \) is a function of the distance between the stations \( i \) and \( j \).

Finally, we define the score, which is associated to customer \( k \) who selects vehicle \( l \) to travel from station \( i \) to station \( j \), as:

\[
S_{lki} = P_{lki} - C_{lki}. \tag{1}
\]

We consider the MOD system balanced when, at each station \( i \in V \), the number of vehicles \( b_i \) is kept between a select minimum \( b_i^{\text{min}} \) and the maximum \( b_i^{\text{max}} \). Given the aforementioned definitions related to the proposed MOD system, we state the main problem studied in this paper.

Problem III.1 (MOD Balancing). Given a set of \( w \) customers’ requests, and \( m \) vehicles parked over \( n \) stations, determine the best assignment for the \( m \) vehicles in order to maximize the system score, serving as many requests as possible, and keeping the number of vehicles balanced over the system’s stations.

B. Optimal Load Balancing

In this section we propose a mathematical formulation to obtain an optimal assignment of vehicles to customers, as stated in Problem III.1. This formulation is inspired by the well-known Profitable Tour Problem (PTP) [9]. However here, multiple vehicles are considered and additional constraints are required.

We define \( \Lambda \in \{0,1\}^{w \times m} \) as the matrix of possible matches between customers and vehicles, i.e., \( \Lambda_{kl} = 1 \) when customer \( k \), waiting in station \( i \), can take vehicle \( l \), parked in \( i \), and 0 otherwise. Given the definition of the score in Eq. (1) and considering the binary decision variables \( x_{lki} \in \{0,1\} \), where \( x_{lki} = 1 \) if vehicle \( l \) is assigned to customer \( k \) to travel from station \( i \) to station \( j \) and 0 otherwise, we formulate Problem III.1 as the following binary integer programming problem:

\[
\max \sum_{l=1}^{m} \sum_{k=1}^{w} \sum_{i=1}^{n} \sum_{j=1}^{n} S_{lki} x_{lki} \tag{2}
\]

subject to:

\[
\sum_{j=1}^{n} x_{lki} \leq \Lambda_{kl} \quad \forall (l, k, i) \in M \times W \times V; \tag{3}
\]

\[
\sum_{k=1}^{w} \sum_{j=1}^{n} x_{lki} \leq \Lambda_{kl} \quad \forall (l, i) \in M \times V; \tag{4}
\]

\[
\sum_{l=1}^{m} \sum_{j=1}^{n} x_{lki} \leq \Lambda_{kl} \quad \forall (k, i) \in W \times V; \tag{5}
\]

\[
\sum_{l=1}^{m} \sum_{k=1}^{w} \sum_{j=1}^{n} x_{lki} \leq \min\{w_i, b_i\} \quad \forall i \in V; \tag{6}
\]
Assignment constraints are expressed in Eqs. (3)-(6). In particular, constraint (3) states that customer $k \in W$ in vehicle $l \in M$ can drive only on a route that departs from station $i \in V$ and arrives to station $j \in V$, constraint (4) states that each vehicle $l$ in station $i$ can be picked up only by a single customer $k$, constraint (5) states that customer $k$ in station $i$ must be associated to only a single vehicle $l$, and constraint (6) states that the sum of the possible routes starting from station $i$ must be equal to the number of possible matches between customers $k_i \in W_i$ waiting at station $i \in V$ and vehicles $l_i \in M_i$ parked in $i$, where $b_i$ and $w_i$ are the number of vehicles parked and that of waiting customers at station $i$, respectively. Finally, constraints (7) represent station capacity restrictions. The minimum number of vehicles allowed in the stations, $b_i^{\text{min}} \in \mathbb{N}^+$, that is set to 1, is imposed in (7a), whereas the maximum number of vehicles allowed in a generic stations, $b_i^{\text{max}}$, is imposed in constraint (7b).

In the problem previously defined, the number of assignable vehicles depends on the number of available vehicles and on that of the customers waiting at that station. In particular, if $b_i$ vehicles and $w_i$ waiting customers are located in station $i$, a number $\min\{w_i, b_i\}$ of vehicles can be assigned. Thus, the solution of problem (2)-(7) always consists in the assignment of $m^* = \sum_{i=1}^{n} \min\{w_i, b_i\}$ vehicles to $m^*$ customers.

It is known that the PTP is a NP-hard problem, since it is a generalization of the Travelling Salesman Problem [9]. As previously stated, problem (2)-(7) is a particular formulation of the PTP for multi-agents systems with balancing constraint, thus, it also belongs to the class of NP-hard problems. Therefore, an efficient approach to handle real-world scenarios without incurring in a heavy computational burden is needed.

### IV. DISTRIBUTED LOAD BALANCING

#### A. Distributed MOD Infrastructure

The distributed implementation of the MOD management system requires that vehicles are able to exchange information through a communication network that connects vehicles and stations. We define the communication network among stations as the undirected graph $G = (V, A)$, where the set of nodes corresponds to all the stations in $V$ and the set of edges $A = \{(i, j) \mid i, j \in V\}$ represents the communication links between stations. Moreover, we assume that each vehicle is equipped with a wireless device that allows it to communicate with the station where it is parked and with all the vehicles parked in the same station. Hence, we define $F_i$ as the set of communication links that connect station $i \in V$ with the vehicles parked in there. Finally, we define the communication infrastructure as the graph $Q = (D, E)$, where the set of nodes $D = V \cup M$ corresponds to the union of the sets of stations and vehicles, and the set of edges is $E = A \cup F$, where $F = \bigcup_{i=1}^{n} F_i$ represents the communication links among the stations and all the vehicles. We also define the neighborhood of vehicle $l$ as $\mathcal{N}_l = \{i \mid (l, i) \in E\}$, which is the set of vehicles with which vehicle $l$ can communicate in one hop, through at least one connected station.

In the rest of the paper we assume that graph $Q$ is connected and that each station collects customer requests in a queue. When a predefined time interval elapses, each station communicates the customers’ requests to the parked vehicles and the assignment procedure starts.

#### B. The DLB Algorithm

The Distributed Load Balancing (DLB) is a two-phase iterative algorithm that aims to maximize the MOD score, as defined in problem (2)-(7), providing a balanced assignment of vehicles to customers. In Phase 1, called Local Request Evaluation, each vehicle, through a bidding strategy, tries to self-assign the most profitable request for both the customer and the provider, from the vehicle point of view, taking into account the balance state of the system based on the information obtained at the previous iteration. In Phase 2, called Balancing, after communicating with its neighbors, each vehicle verifies the feasibility of the previous local assignment. Thus, the assignment is cancelled if it leads to an unbalanced state of the MOD system, otherwise it is maintained. After Phase 2, the vehicle executes again Phase 1, starting a new iteration. We define with $t \in \mathbb{N}$ the generic iteration of the DLB algorithm, which consists of a single run of the two phases. The DLB algorithm terminates when the convergence on a balanced assignment is reached, i.e., if all the vehicles in $M$ own the same information regarding the customer-to-vehicle assignment and the corresponding score.

The definition of distributed termination strategies, for which several strategies exist [10], is out of the scope of this paper. Thus, here we detect the convergence of the algorithm in a centralized fashion.

In the following we describe the two phases in detail.

1) Phase 1 - [Local Request Evaluation]: The first phase of the algorithm is the local assignment process. Algorithm 1 shows Phase 1 of the DLB for vehicle $l \in M$ at iteration $t$. This phase is performed by all the vehicles at each iteration $t$. Hereinafter, we assume that all vehicles are synchronized.
by means of distributed synchronization protocols \[11\], to ensure that all vehicles execute the same phase of the algorithm at the same time. In real world scenario this constraint can be relaxed using a proper asynchronous adaptation of the DLB algorithm.

For each vehicle \( t \), we define the customer-to-vehicle assignment vector as \( \gamma_t(t) \in (W \cup \{\emptyset\})^n \), where the null element \( \emptyset \) indicates that the vehicle is not assigned to any customer. We also define the set \( W^i(t) = \{ t \} \subseteq W \) that contains the customers waiting for the assignment at station \( i \). At iteration \( t \), the set \( W^i(t) \) is populated with the customers who have not been assigned a vehicle in the previous iteration \( t - 1 \) (line 1 in Algorithm 1). Hence, for each customer \( k \in W^i(t) \) (line 2), vehicle \( l \) evaluates what is the most profitable destination among all the stations (lines 3-5). For each destination station \( j \in V \), vehicle \( l \) checks the station capacity vector \( b_l(t) \in N^n \), updated at the previous iteration \( t - 1 \). Hence, vehicle \( l \) stores in the element \( b_{l,j}(t) \), the number of vehicles parked in station \( j \) on the basis of its knowledge at iteration \( t \). If the number of vehicles at the selected destination station \( j \) does not exceed the maximum number of parking lots \( b_j^{\text{max}} \), then the Boolean function \( I() \) returns 1, otherwise it returns 0 (line 3). The obtained values are stored in the balancing vector \( \beta_l(t) \in \{0, 1\}^n \), where \( \beta_{l,j}(t) = 1 \) if station \( j \) is balanced, on the basis of the knowledge that vehicle \( l \) has at iteration \( t \), and 0 otherwise. Given the balancing vector \( \beta_l(t) \), the destination station with the maximum score \( \delta_{l,i}^{\text{max}} \) is selected for customer \( k \). For each vehicle \( l \), we define the temporary destination vector \( \delta_l^t(t) \in V^w \) and the temporary score vector \( \sigma_l^t(t) \in R^w \) as the vectors in which the maximum score \( \delta_l^t_{l,k}(t) \) (line 4), and the corresponding preferred destination \( \sigma_l^t_{l,k}(t) \) (line 5) for each customer \( k \), are stored, respectively. In the case of two or more destinations having the same score value, a tie-break rule is needed. We choose to assign the vehicle to the request asking for the destination with fewer vehicles, in order to improve the system balance. Hence, vehicle \( l \) updates vector \( \gamma_l(t) \) with the index of the customer with the maximum score \( \sigma_l^t \) (line 7), and the corresponding maximum score value is stored in the score vector \( \sigma_l(t) \in R^m \) (line 8). Finally, the destination vector \( \delta_l(t) \in V^m \) (line 9) and the starting vector \( \rho_l(t) \in V^m \) (line 10) are updated with the destination of the selected customer and with the departure station of vehicle \( l \), respectively. The last three vectors, defined for each vehicle \( l \), contain respectively the score value, the destination and the departure stations obtained by each vehicle at iteration \( t \). Also in this case, two or more requests can result in the same score value, thus we assume that vehicle \( l \) is assigned to the first request received (First-Come First-Served policy).

Once vehicle \( l \) is assigned to the most profitable request among those arrived from customers waiting at station \( i \), the vehicle sends and receives the vectors \( \gamma_l(t), \sigma_l(t), \delta_l(t), \rho_l(t) \) to/from its neighbors \( N_i \), then moves to Phase 2.

2) Phase 2 - [Balancing]: The second phase of the algorithm is the balancing process. In this phase each vehicle performs a conflict resolution procedure to avoid that the requests selected in Phase 1 yield an unbalanced system. The vehicle’s conflict resolution takes place synchronously in order to obtain a common conflict-free assignment. Algorithm 2 shows Phase 2 of DLB. This phase is repeated for each vehicle \( l \in M \) at each iteration \( t \). At first, vehicle \( l \) evaluates if the requests received by the customers waiting at its departure station \( l \) leads the system to overload the drop-off stations or to make the pick-up stations empty, respectively (line 1 of Algorithm 2). In the case of prospective unbalanced scenarios, vehicle \( l \) starts to scan all possible destinations \( j \in V \) (lines 2-8), then all possible pick-up stations \( i \in V \) (lines 10-17) to try to set an alternative balanced assignment. To this aim, requests that have station \( j \in V \) as a destination are identified and stored in the set \( D(l) \) (line 3). Then, the current number of vehicles parked in \( j \) and the number of vehicles assigned to a customer, and then stored in the set \( D(l) \), at iteration \( t \) are summed and stored in the variable \( b^D(t) \) (line 4). If the number of requests is greater than the maximum capacity \( b_j^{\text{max}} \) (line 5), the \textsc{update} procedure is triggered (line 6 of Algorithm 2).

The \textsc{update} procedure, given a set of vehicles \( X(t) \), the score vector \( \sigma_l(t) \), and the number of exceeding requests \( X^*(t) \), returns the updated set of vehicles \( X(t) \) and the vectors \( \gamma_l(t), \sigma_l(t), \delta_l(t), \rho_l(t) \), where the requests with the lowest scores that exceed the station capacity, are rejected.

Given the set \( D(l) \), returned from the \textsc{update} procedure, the station capacity \( b_{l,j}(t) \) is then updated only with the requests that maintain the destination stations balanced (line 7 of Algorithm 2). Subsequently, the pick-up stations are analyzed (line 10 of Algorithm 2). First, the set \( S(t) \), which contains the vehicles that have station \( i \) as origin, is populated (line 11). We remark that only one-way routes are considered in \( S(t) \), i.e., routes with destination station different from the origin one, since round-trip routes do not affect the system balance. Then, the capacity of station \( i \) is updated evaluating the number of vehicles in \( S(t) \) and the current capacity \( b_{i,j}(t) \) (line 12). If the number of requests for station \( i \in V \) is less than \( b_{i,j}^{\text{min}} \) (line 13), then the least profitable requests, equal in number to those that exceed the capacity of station \( i \), are rejected using the aforementioned \textsc{update} procedure (line 14).

**Algorithm 2 - DLB: Phase 2 for vehicle \( l \) at iteration \( t \)**

**Input:** \( \gamma_l(t), \sigma_l(t), \delta_l(t), \rho_l(t), b_l(t) \)

**Output:** \( b_{l,i}(t) \)

1. While \( b_{l,h}(t) > b_{l}^{\text{max}} \lor b_{l,h}(t) < b_{l}^{\text{min}} \) \( \forall h \in V \) do
2. For \( j \in V \) do
3. \( D_l(t) = \{ l \mid \delta_{i,l}(t) = j \} \)
4. \( b^D(t) = b_{l,j}(t) + |D(l)| \)
5. If \( b^D(t) > b_{l,j}^{\text{max}} \) then
6. Update \( b^D(t), D_l(t), \sigma_l(t) \)
7. \( b_{l,i}(t) = b_{l,i}(t) + |D(l)| \)
8. End if
9. End for
10. For \( i \in V \) do
11. \( S_l(t) = \{ l \mid \rho_{i,l}(t) = i, \delta_{i,l}(t) \neq i \} \)
12. \( b^S(t) = b_{l,i}(t) - |S_l(t)| \)
13. If \( b^S(t) < b_{l,j}^{\text{min}} \) then
14. Update \( b^S(t), S_l(t), \sigma_l(t) \)
15. \( b_{l,j}(t) = b_{l,j}(t) - |S_l(t)| \)
16. End if
17. End for
18. End while
paired with a finite number of rejected requests. Thus, vehicle l,1, is performed until all stations are balanced.

We observe that at the end of Phase 2, the assignment obtained for vehicle l in Phase 1 could result within the set of rejected requests. Thus, vehicle l will be able to self-assign another request at the next iteration t + 1.

V. NUMERICAL RESULTS

In this section the outcome of a Monte Carlo simulation campaign to assess the performance of the DLB algorithm is presented. This analysis has been performed through a simulator implemented in MATLAB. Simulations were run on a workstation with a 2.67 GHz quad-core processor with 4 GB of RAM.

Since the main purpose of this paper is to effectively present the proposed load balancing framework and the associated distributed strategy, we made the following simplifying assumptions on the simulated scenarios. The number of stations ranges from 5 to 20. Each station is randomly placed in a simulated square environment of 3x3 km, and paired with a finite number \( b_{i}^{\text{max}} = 5 \) \( \forall i \in V \) of maximum available parking lots. The simulator starts by placing one vehicle in each station and the remaining \( m - n \) vehicles are randomly distributed over stations with available parking lots. Finally, a finite number of customers that range from 10 to three times the number of available vehicles are randomly distributed over the stations. In the simulation campaign the number both of vehicles and customers for each trial is increased by steps of 5 units.

For each experiment, the results are averaged over 30 independent trials. Figures 1 - 2 show the averaged results, where error bars indicate the 95% confidence interval.

A. Optimal Balancing vs Distributed Balancing

Due the high computational burden of the optimization problem, we compare optimal solution (OPT), obtained by using the MATLAB solver bintprog, with one computed using the DLB algorithm only for MOD systems with a limited number of stations, ranging in particular from 5 to 10.

Figure 1 displays the comparison between OPT- and DLB-based assignment scores for scenarios with 5 and 6 stations. In particular, Figs. 1(a)-(c) show the trend of the assignment score with 10 customers and different numbers of vehicles for the aforementioned networks of stations. We observe that DLB tends to favor the balancing of the MOD system rather than the maximization of the score value. Thus, when the number of vehicles is comparable to the number of stations, i.e., 5 and 6 respectively, or to the number of available parking lots, i.e., 25 and 30 respectively, the difference in score between OPT and DLB is high. On the other hand, we observe that in the cases in which the number of vehicles is lower, i.e., 15 vehicles in Fig. 1(a) and 16 vehicles in Fig. 1(c), the DLB is able to explore more alternative assignments to find a better solution that reduces the difference with the OPT score.

Furthermore, simulations are also performed varying the number of customers. Figures 1(b)-(d) show a comparison of OPT and DLB scores for networks with 5 stations and 10 vehicles, and 6 stations and 11 vehicles, respectively. We observe that the difference between OPT and DLB score is minimized when the number of customers, keeping the number of vehicles constant, is increased. This behavior is due to the fact that with a greater number of customers, more possible assignments that allow the DLB to find a better balanced assignment that maximizes the score exist. This observation let us conclude, since it shows that the DLB tends to approach the OPT performance for small-size systems. Thus, for medium- and large-size MOD systems (i.e. with 20-50 vehicles and more than 50 vehicles, respectively) the DLB clearly provides a suboptimal solution. However, this solution yields a balanced system and is achieved in a reasonable computation time.

B. Distributed Load Balancing performances

In the analysis of the DLB algorithm, we consider the following performance indices [12]. The Mean Balancing Error (MBE) is a metric of the balancing status, \( \text{MBE} = \frac{1}{n} - \sum_{i=1}^{n} |b_i - \frac{m}{n}| \). If MBE = 0, then the MOD system is balanced, that is to say, vehicles are uniformly distributed over the stations, while MBE = 1, if the system is heavily unbalanced. In addition, to assess the satisfaction of the customers, we analyze the gap between the DLB solution and the original customers’ requests by defining the Additional Walking Distance (AWD), as \( \text{AWD} = \sum_{k=1}^{m} d_k(j^*, j) \). We recall that \( d_k(j^*, j) \) represents the sum of the additional distances for all customers assigned to vehicles.

Figure 2 shows the performance of the DLB algorithm for scenarios with 15 stations. In particular, Fig. 2(a) shows the assignment score for 4 sets of vehicles of different size.
The performance of the DLB algorithm has been assessed through an extensive Monte Carlo simulation campaign. We comment that the DLB algorithm tends to privilege the system balancing over the system score. This implies that, when the system is closed to a strongly unbalanced configuration, the optimization of the score is sacrificed in favor of the system rebalancing. In these cases, in fact, the achieved score results far from the optimal solution. This is due to the fact that DLB must respect the balancing constraints, even in spite of assignments that tend to decrease the overall score. On the other hand, when multiple feasible assignment options are available, the DLB algorithm succeeds in assigning vehicles to customer while keeping the system balanced and achieving a score close to the optimal one.

VI. CONCLUSIONS

In this paper, a novel optimization-based framework for Mobility-On-Demand systems has been presented. This formulation solves the load balancing problem taking into account the needs of both customers and vehicles. Furthermore, a distributed strategy able to produce a sub-optimal solution of the load balancing problem has been proposed.

The performance of the DLB algorithm has been assessed through extensive Monte Carlo simulation campaigns. We comment that the DLB algorithm tends to privilege the system balancing over the system score. This implies that, when

REFERENCES