In *Proceedings of the 9th IFAC Symposium on Intelligent Autonomous Vehicles* (IAV 2016), Leipzig, Germany, June 2016.

Minimum Cost Matching for Autonomous Carsharing

Josiah P. Hanna [*] **Michael Albert** [*] **Donna Chen** [**] **Peter Stone** [*]

[*] *Computer Science Department, University of Texas at Austin, Austin, TX 78712 USA (e-mail: {jphanna,malbert,pstone}@cs.utexas.edu).*
[**] *Department of Civil & Environmental Engineering, University of Virginia, Charlottesville, VA 22904 USA (e-mail: author@snu.ac.kr)*

Abstract: Carsharing programs provide an alternative to private vehicle ownership. Combining carsharing programs with autonomous vehicles would improve user access to vehicles thereby removing one of the main challenges to widescale adoption of these programs. While the ability to easily move cars to meet demand would be significant for carsharing programs, if implemented incorrectly it could lead to worse system performance. In this paper, we seek to improve the performance of a fleet of shared autonomous vehicles through improved matching of vehicles to passengers requesting rides. We consider carsharing with autonomous vehicles as an assignment problem and examine four different methods for matching cars to users in a dynamic setting. We show how applying a recent algorithm (Scalable Collision-avoiding Role Assignment with Minimal-makespan or SCRAM) for minimizing the maximal edge in a perfect matching can result in a more efficient, reliable, and fair carsharing system. Our results highlight some of the problems with greedy or decentralized approaches. Introducing a centralized system creates the possibility for users to strategically mis-report their locations and improve their expected wait time so we provide a proof demonstrating that cancellation fees can be applied to eliminate the incentive to mis-report location.

Keywords: Autonomous vehicles, autonomous mobile robots, Minimal-makespan matching, Intelligent Transportation Systems, Multi-Vehicle Systems, SCRAM, Car-sharing

1. INTRODUCTION

Autonomous vehicle technology is becoming a reality. Driverless cars have the potential to make travel safer, increase fuel economy, and reduce pollution (Fagnant and Kockelman, 2014). However the potential of this technology does not have to be limited to a new way to move people from point A to point B. Autonomous vehicles have the potential to revolutionize modern transportation systems such as intersections (Dresner and Stone, 2008), dynamic lane openings and closings (Hausknecht et al., 2011), and, as we demonstrate, carsharing.

Carsharing is a transportation service where users can access a personal vehicle when desired without the need to own a private vehicle. A user reserves a vehicle and then goes to the vehicle's location where service begins. The user can then end his or her trip at any location within a given geofence. One problem with current systems is that vehicles may be moved from high demand to low demand areas. Then the vehicle is likely to remain unused for a long time between trips. Additionally, the nearest vehicle to a user may be too far to walk. Both of these problems have hindered the growth of carsharing programs.

Autonomous vehicles offer a promising solution to these problems. An autonomous vehicle can meet a user at a requested location and drop him or her off at the desired destination. When not in use, the vehicle can move to other users or strategically relocate to an area with high demand (Fagnant and Kockelman, 2014). Driverless cars can remain in service at all times (except for maintenance or refueling) so the carsharing fleet remains close to full strength throughout the day, and automated communication between cars can allow for coordination to improve system performance. We consider carsharing with autonomous vehicles as a multiagent system and look at how best to coordinate the assignment of vehicles to users to improve the system performance. Specifically, we view users and available vehicles as forming a bipartite graph. We examine and evaluate different maximal matching algorithms in the context of a carsharing simulation developed by Fagnant and Kockelman (2014).

For an autonomous carsharing system to be practical it must meet three requirements; 1) the system must be optimal, meaning it provides users service in the shortest amount of time possible after they request a ride, 2) the system must be efficient, meaning distance traveled without passengers in the vehicle should be minimized as they are not directly serving users but still contribute to pollution and add fuel costs, and 3) the system must be "fair," meaning that a few users are not overly penalized for the sake of other users. A simple minimum cost maximal matching of the user-vehicle graph could address the requirements of optimality and efficiency, but it likely will leave some users with very long wait times in order to service a few users faster. To better address these issues we use a recent algorithm called SCRAM (Scalable Collision-avoiding Role Assignment with Minimal-makespan) from the class of minimal-makespan matching algorithms (MacAlpine et al., 2015). SCRAM finds a

[⋆] This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287), ONR (21C184-01), AFRL (FA8750-14-1-0070), AFOSR (FA9550-14-1-0087), and TxDOT (0-6838, 0-6847). Josiah is supported by an NSF Graduate Research Fellowship.

matching in a bipartite graph such that the maximal cost edge in the matching is minimized. Minimizing the makespan (the time it takes for all vehicles to reach their destination) ensures that we obtain an efficient allocation of cars to users subject to the constraint that the longest time any single user has to wait is minimized. In this sense, we view SCRAM as a more "fair" allocation of cars to users.

However, we also demonstrate that while centralized approaches to car assignment in a carsharing system can be more efficient and fair, they are subject to strategic manipulation by the users. We illustrate this problem with a situation under which a user achieves a lower expected wait time if he falsely reports his location. While, previous work has examined the incentive to strategically misreport the priority of trips (Kamar and Horvitz, 2009; Kleiner et al., 2011), we believe we are the first to demonstrate the feasibility of strategically misreporting one's location. We also show that this problem can be solved with an appropriate cancellation fee for certain centralized approaches.

2. RELATED WORK

The car-sharing system discussed in this paper is an example of an autonomous mobility-on-demand (AMoD) system (Zhang et al., 2015). Previous work has proposed various methods for optimizing the user-vehicle assignment problem in AMoD systems. Acquaviva et al. use a mixed integer linear program to optimize service (2015) while Zhang et al. propose solutions based on queueing networks (2016). These approaches do not consider "fairness" as an objective and assume cars begin serving users at fixed stations which is an assumption we relax here. As mentioned in Section 1, one problem with existing carsharing services is that vehicles may be moved out of areas of high demand and then remain unused. Autonomous vehicles do not face this problem since they can always reach users on their own; however, a strategic relocation strategy can reduce unoccupied vehicle distance travelled while ensuring enough vehicles to meet demand (Fagnant and Kockelman, 2014). Dynamic pricing of trips based on an auction mechanism is another way to ensure that vehicles are located where they are most needed by customers (Chow and Yu, 2015). In contrast to these two methods we use matching algorithms to efficiently get vehicles where they are needed without additional mechanisms for already having vehicles in the right place.

Another related body of work is the taxi-dispatch literature. Similar to future carsharing systems, systems of taxis and customers can also be modelled as multiagent systems (Chen and Cheng, 2010). While the distinction between taxis and shared autonomous cars may seem narrow, human taxi drivers bring their own preferences and utilities which need to be modelled to maximize performance (Seow et al., 2007). In contrast, autonomous vehicles can be operated purely for customer or system utility. Previous work has also considered what information should be made available to users and drivers in a taxi-dispatch system to facilitate service (Lee et al., 2013). We assume full knowledge (or knowledge of a distribution) of other agents' positions in our analysis of strategic agent behavior. While all of this prior work contributes to smarter transportation systems, this research is, to the best of our knowledge, the first to study in detail the impacts of different matching algorithms on autonomous carsharing.

3. ALGORITHMIC APPROACHES TO CAR ASSIGNMENT

The challenge we consider is how to assign vehicles to users in a setting where the number of available vehicles and users requesting rides is constantly changing. An obvious way to handle the assignment problem would be to let users request the nearest vehicle to him on a first come first served basis which is the approach adopted by current transportation and carsharing services such as Uber and Car2Go. Unfortunately such straightforward approaches are not likely to be optimal in terms of average user wait time or vehicle distance traveled in excess of what is needed to meet demand. In addition, we would like to have a system that is "fair" in that it does not achieve optimality by forcing a small subset of users to experience very long wait times.

We assume time is discretized and at every time step users can make requests for vehicles and vehicles can move a fixed amount. We also assume that once a vehicle is assigned to a user they will keep traveling to that user. This assignment problem can be represented as a sequence of graphs $G_t = (U_t, V_t, E_t)$ where U_t is the set of users requesting vehicles at time t and V_t is the set of available vehicles at time t. E_t is the set of edges $e = (u, v)$ representing the current distance from user u and vehicle v at time step t. For each edge, $e = (u, v)$, we use $e.dist$ to refer to the distance from u to v. Any user or vehicle not assigned at time step t will appear in graph G_{t+1}. At each time step the goal is to find an assignment, A_t, which is a set of edges matching users to vehicles. Formally, we want to maximize the size of A_t while minimizing the sum of distances over the edges in A_t and while minimizing the longest edge that occurs in A_t. This equates to serving as many users as possible, reducing user wait and excess distance traveled, and finding a "fair" solution. We next introduce a series of increasingly complex matching algorithms to address the assignment problem, and motivate their strengths and weaknesses.

3.1 Greedy Matching

The simplest matching algorithm is to allow users to request the nearest unoccupied vehicle. We refer to this method as the decentralized method because it could be implemented with direct communication between a user and a vehicle. For experiments, we implement the method by greedily matching users to vehicles in random order. At each time step we draw a user, u, uniformly at random from U_t and attempt to pair him with a vehicle, v, such that $(u, v).dist$ is minimized. If successful, u and v are removed from U_t and V_t respectively and (u, v) is added to the set of assignments, A_t. If unsuccessful u will be added to the set of unserved users, $\bar{A}_t$, which can be considered at the next time step. Pseudocode is shown in Algorithm 1. Figures 1a and 1b illustrate how this approach can result in sub-optimal matchings.

3.2 Centralized Greedy Matching

A natural improvement on the decentralized greedy approach is to provide centralized coordination to ensure that every car is matched with its closest user. The centralized greedy matching algorithm starts by iterating over each $u \in U_t$ and trying to find $v \in V_t$ such that $(u, v).dist \leq r$ where r is a fixed radius. For each (u, v) found, u and v are removed from U_t and V_t and the procedure repeats with r incremented. The loop terminates when r reaches a preset maximum value or U_t or V_t becomes empty. Full pseudocode is shown in Algorithm 2. The advantage of this approach is that a vehicle will not be assigned

Algorithm 1 Pseudocode for the decentralized greedy matching assignment algorithm. The subroutine **findClosestVehicle**(u, V) returns $v \in V$ that minimizes $(u, v).dist$ or **NULL** if one does not exist.

```
 1: function DECENTRALIZED($G_t = (U_t, V_t, E_t)$)
 2:     $A_t = \emptyset$
 3:     $\bar{A}_t = \emptyset$
 4:     for all $u \in U_t$ do
 5:         $v =$ findClosestVehicle$(u, V_t)$
 6:         if $v \neq$ NULL then
 7:             $A_t = A_t \cup \{(u, v)\}$
 8:         else
 9:             $\bar{A}_t = \bar{A}_t \cup \{u\}$
10:         end if
11:     end for
12:     return $A_t, \bar{A}_t$
13: end function
```

Algorithm 2 Pseudocode for the centralized greedy matching assignment algorithm. l is the maximum radius to search. The subroutine **searchForCarAtDistance**(u, V, r) returns a vehicle such that $(u, v).dist = r$ or **NULL** if one cannot be found.

```
 1: function GREEDYCENTRALIZED($G_t = (U_t, V_t, E_t), l$)
 2:     $A_t = \emptyset$
 3:     $r = 1$
 4:     while $r \leq limit$ and $U_t \neq \emptyset$ and $V_t \neq \emptyset$ do
 5:         for all $u \in U_t | u.assigned = False$ do
 6:             $v =$ searchForCarAtDistance$(u, V_t, r)$
 7:             if $v \neq$ NULL then
 8:                 $A_t = A_t \cup \{(u, v)\}$
 9:             end if
10:         end for
11:         $r = r + 1$
12:     end while
13:     $\bar{A}_t = \{u \in U_t | u.assigned = False\}$
14:     return $A_t, \bar{A}_t$
15: end function
```

to a user if it could have been assigned to another passenger with a smaller distance. The disadvantage is that there are no guarantees that the assignment will be optimal. Figures 1c and 1d illustrate a scenario where this approach fails to produce the optimal solution.

3.3 Hungarian Algorithm

The Hungarian algorithm (Kuhn, 1955), which is guaranteed to find a minimum cost perfect matching in a bipartite graph, is a natural fit for the goals of efficiency and optimality. Since the Hungarian algorithm requires a perfect matching, and in general $|U_t| \neq |V_t|$, we augment the smaller of the two sets with dummy vertices, u_d or v_d. For all edges of the form $e = (u, v_d)$ or $e = (u_d, v)$ we have $e.dist = 0$. When the Hungarian algorithm is run on this new graph, the assignment returned will be the maximal matching for G_t once edges that include dummy vertices are removed (MacAlpine et al., 2015). In the case that $|U_t| > |V_t|$ any user matched with a dummy vehicle is unable to be matched and must try again in the next time step. The Hungarian algorithm can be implemented to run in time $O(n^3)$, fast enough to compute in real time for problems of this scale. The Hungarian algorithm is well known so we omit pseudocode. Our implementation matches the input and output of the other algorithms.

Algorithm 3 Pseudocode for SCRAM. The subroutine **getMinimalMaxEdge** minimizes the longest edge in a perfect matching.

```
 1: function SCRAM($G_t = (U_t, V_t, E_t)$)
 2:     $A_t = \emptyset$
 3:     $e_{max} =$ getMinimalMaxEdge$(E_t)$
 4:     $E'_t = \{e \in E_t | e.dist \leq e_{max}.dist\}$
 5:     $A_t =$ hungarian$(E'_t)$
 6:     for all $e = (u, v) \in A_t$ do
 7:         $u.assigned = True$
 8:     end for
 9:     $\bar{A}_t = \{u \in U_t | u.assigned = False\}$
10:     return $A_t, \bar{A}_t$
11: end function
```

3.4 SCRAM for Minimal Makespan Matching

While, the Hungarian algorithm produces a minimum cost perfect matching, it does not consider the “fairness” of the system. Therefore, some users may face a very long wait time even though the average is minimized. To avoid this problem, we use a recent algorithm known as scalable collision-avoiding role assignment with minimal-makespan (SCRAM) to find an assignment that minimizes the longest distance that any vehicle must travel to a passenger. Specifically we implement the Minimum Maximal Distance + Minimum Sum Distance^2 (MMD+MSD^2) variant of SCRAM as proposed by MacAlpine et al. (2015). SCRAM has previously been applied to formational positioning in simulated robot soccer – a setting with a small, static number of agents (11 agents and 11 target destination) (MacAlpine et al., 2015). In constrast we apply the algorithm to a large, dynamic, real-world setting.

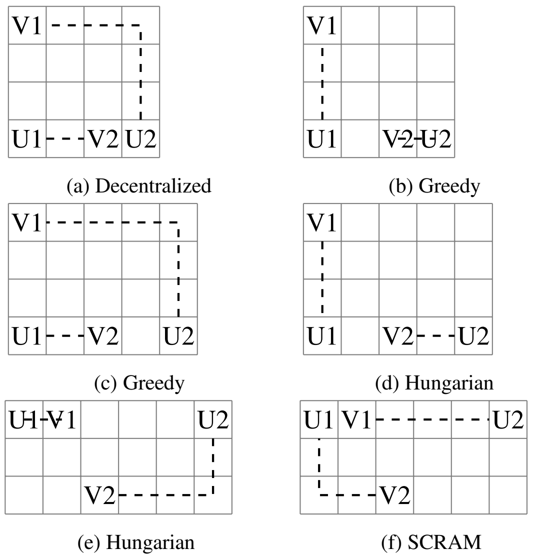
Given a bipartite graph, $G_t = (U_t, V_t, E_t)$, SCRAM first finds the minimum maximal edge, e_{max} in a perfect matching from U_t to V_t, which is the minimum makespan for a feasible matching (Algorithm 3, line 3). Then edges $e \in E_t$ such that $e.dist > e_{max}.dist$ are removed from E_t. Finally, the Hungarian algorithm is run on the reduced set E_t to find the minimum cost matching over these edges. The algorithm also runs in time $O(n^3)$. Similar to the Hungarian algorithm, SCRAM assumes $|U_t| = |V_t|$, so we make use of dummy vertices when this assumption does not hold. Pseudocode is provided in Algorithm 3.

Figure 1f shows the solution SCRAM would find for a small two user and two vehicle examples. In contrast, the Hungarian algorithm, Figure 1e, is much less equitable.

4. SIMULATED CARSHARING MODEL

As a testbed we use the agent-based carsharing model proposed by Fagnant and Kockelman (2014). There are two types of agents in this simulator: vehicles and users. Vehicles can move north, south, east, or west to adjacent cells ($e.dist$ is Manhattan distance). Motion within the cells is not modeled and vehicles do not interfere with each other's path. At each time step users make requests to the system for a vehicle to take them from their current location to their destination. Time is discretized into five minute intervals for a total of 288 time steps per day ($0 \leq t < 288$). Each run of the simulation corresponds to one 24 hour day. The model represents a 10 mile by 10 mile city as a grid of 0.25 mile by 0.25 mile grid cells.

Users requesting vehicles are generated in each grid cell at a rate that decreases the farther a cell is from the city center. In



(a) Decentralized (b) Greedy (c) Greedy (d) Hungarian (e) Hungarian (f) SCRAM

Fig. 1. These simple examples illustrate the relative advantages of different algorithms used in the experiments. Each row depicts a fixed scenario. For each scenario, the subfigure on the left show a suboptimal solution with one algorithm and the subfigure on the right show a better solution from a different algorithm. The bottom row shows how SCRAM minimizes distance of the longer assignment but not the total distance traveled.

other words, the center of the city is an area with high user density and the density falls off towards the edges of the map. Full details on the process can be found in the paper introducing the simulator (Fagnant and Kockelman, 2014). Each day sees approximately 36,000 requests for vehicles.

At each time step t, a matching algorithm attempts to match users requesting vehicles (U_t) with available vehicles (V_t). If a user cannot be served they are added to the set of users making requests at the next time step (U_{t+1}). If a user cannot be served after 30 minutes of wait time (six consecutive requests) the user ceases requests and is counted as unserved. Vehicles that are low on fuel go to the nearest fueling station (assumed to be within the same grid cell) and are out of service for two time steps (10 minutes). Vehicles that are assigned then begin to move towards the passengers. Vehicles already with passengers continue to their user's destination. Each vehicle can move a fixed number of grid cells per time step depending on the maximum speed for its location in the grid.

The model has several limitations. The model approximates the effects of a carsharing system in a relatively small 10 mile by 10 mile urban area. Though car speed has small variations due to time of day and distance from the city center congestion is not extensively modelled. While the presence of other modes of transportation is approximated by the user generation rates and speed variations, the simulation only explicitly models vehicles and users of the carsharing service. However, the model has been used in prior carsharing studies (Fagnant and Kockelman, 2014; Fagnant et al., 2015) and is thus considered a sufficient simulation for this domain.

5. EXPERIMENTAL RESULTS

The experiments compare the performance of the above algorithms over 100 consecutive days of fleet use. Note the events of each day affect the next (e.g. vehicles starting position for

Fig. 2. Excess vehicle distance (miles travelled while the vehicle is unoccupied) for each assignment method. Error bars are for a 95% confidence level.

a given day is the previous day's ending position), so multiple days are needed to get an idea of system performance. In these experiments the fleet size is set to 1000 vehicles which is an experimentally determined value able to serve all users with any assignment algorithm. It is important that all users be served in order to make meaningful comparisons. Otherwise, an assignment strategy could minimize excess distance travelled by simply not serving passengers that are far from available vehicles. Initial locations of vehicles are chosen according to the distribution of user requests over the simulated area. We run 50 trials of 100 days each for each assignment algorithm.

5.1 Unoccupied Distance Traveled

The first comparison we make is on the average amount of unoccupied distance travelled by vehicles per day. This metric demonstrates the efficiency of a given assignment algorithm. Figure 2 shows the distance traveled (in miles) by unoccupied vehicles. The results show that the more sophisticated matching approaches, SCRAM and the Hungarian algorithm, are able to reduce this number by approximately 20 percent from the baseline decentralized approach and also improve over the centralized greedy baseline. The Hungarian algorithm performs better than SCRAM as it solely minimizes the sum of unoccupied distance traveled, while SCRAM is subject to the constraint that the solution has a minimum makespan as well. However, SCRAM only produces one percent more additional unoccupied distance traveled than the true minimum solution.

5.2 Passenger Wait Time

The next point of comparison is passenger wait time. Wait time is a combination of how long it takes users to be matched to a vehicle and how long it takes for the assigned vehicle to reach the user:

$$Wait_{total} = Wait_{matching} + Wait_{travel}.$$

For all methods the average wait was less than five minutes (i.e. on average all users could be assigned a vehicle the first time they requested one) which means $Wait_{matching}$ is zero for most users. The average wait time shows the same relationship among methods as unoccupied distance traveled so we do not show results.

While a minimal average wait time is good, we also want the service to be reliable and predictable for the users. To characterize reliability we look at variance in the wait time and the number of users that have to wait longer than given time thresholds. Results are shown in Figures 3 and 4. First, the variance results show that both the Hungarian algorithm and SCRAM reduce variance in the wait time compared to the greedy approaches. However, SCRAM outperforms the Hungarian method since minimizing the makespan results in a smaller range in which user-vehicle distances can fall. Second, Figure 4 shows that SCRAM is able to minimize the number of users on average that must wait longer than certain time thresholds. While SCRAM's solution has a slightly longer average wait than the Hungarian algorithm, it spreads that total wait time out among all of the users instead of concentrating it among a subset.

Fig. 3. Variance of user wait time for each assignment method. Error bars are for a 95% confidence level.

Fig. 4. The number of users waiting longer than 5, 10, and 15 minutes for each assignment method. Error bars are for a 95% confidence level.

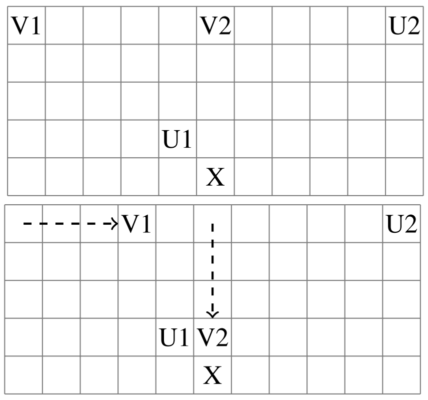


Fig. 5. A setting for which $U2$ can benefit by reporting his location as X. The optimal solution under SCRAM or the Hungarian algorithm assigns $V1$ to $U1$ and $V2$ to $U2$. This results in a total distance of 12 and a makespan of 7. If $U1$ reports position X then the assignment is reversed. After three steps $U2$ can report his true location and maintain his preferred assignment. This results in a cost of 14 and a makespan of 10 for both algorithms.

6. STRATEGIC MISREPORTING OF USER LOCATIONS

While all three centralized algorithms (centralized greedy, the Hungarian algorithm, and SCRAM) are significant improvements to the decentralized greedy matching algorithm in terms of efficiency and fairness, the additional complexity that allows for these improvements also comes at a cost of strategic manipulability. Given that any centralized algorithm may potentially not assign the closest vehicle to a user in order to maintain system wide efficiency, a user may be able to strategically misreport his location in order to draw a vehicle to him. Specifically, he can influence the assignment of vehicles by claiming to be at a location that both increases the likelihood of his preferred vehicle being assigned to him and having a route that passes near his true location. Once the vehicle begins to move towards the false location, the user can cancel the initial request, freeing up his preferred vehicle, and requesting a new pickup at their true location.[1] Assuming that the user strategically cancels their initial reservation, when the assignment is computed again the user will be assigned the vehicle that has now moved closer to him.

Figure 5 shows a scenario where, under both the Hungarian algorithm and SCRAM, if an agent ($U2$) reports a false location (X) he will receive a shorter expected wait time at the expense of system efficiency. While not shown, it is trivial to construct examples for the centralized greedy algorithm where strategic misreporting is beneficial as well. This is likely to be a fundamental feature of any centralized assignment algorithm, and therefore, should be considered in the future design of optimal carsharing systems. Note that the decentralized algorithm is robust to manipulative behavior because users are guaranteed, on a first come first served basis, to be assigned the closest available vehicle. In the language of game theory, a user truthfully reporting his location is a dominant strategy – optimal no matter what other users may report.

6.1 *Restoring Strategy-Proofness with Cancellation Fees*

We propose that by introducing cancellation fees, even centralized mechanisms such as the Hungarian algorithm and SCRAM can be designed to ensure mis-reporting is not optimal. It is necessary to assign a value of time to users in order to penalize mis-reporting with a monetary payment. In this discussion, we will assume that all users have the same value of time of one unit of time for one monetary unit although we could also use a reasonable upper bound.

When considering cancellation fees, it is important to keep two points in mind. First, for an infinite grid, no fixed finite cancellation fee can guarantee optimality, because we can always increase the benefit of misreporting by moving users farther apart. Second, the cancellation fee should be minimal in case a user has a legitimate need to cancel their request.

We propose setting the cancellation fee equal to Vickery-Clarkes-Groves (VCG) payments (Vickrey, 1961). VCG payments, loosely defined (see reference for an in depth discussion), are the difference between the total social welfare not considering the cancelled user's utility, and the total social welfare assuming that the cancelled user had never made his initial request. Assume that there are n users, and the time, in a given assignment, for a vehicle to reach each user $i \in \{1, ..., n\}$ is t_i^c for the case where user j cancels; t_i^* if user j had never reported; and t_i if user j had reported truthfully. Then total social welfare, not including j, is $-\sum_{i \neq j} t_i$, when j reports and cancels, and $-\sum_{i \neq j} t_i^*$ if j had never reported. The VCG cancellation fee would thus be $\sum_{i \neq j} t_i - \sum_{i \neq j} t_i^*$ for user j.

Theorem 1. If there is a VCG cancellation fee, for the Hungarian assignment algorithm, no user ever has an incentive to misreport and cancel.

Proof. Note that the Hungarian algorithm always maximizes social welfare, $-\sum_i t_i$, so for user j, if he reports truthfully, he will receive utility $-t_j$. If he lies he will receive a utility of $-t_j^r$ instead. Therefore, his benefit from cancelling is $t_j - t_j^r$. His cancellation fee will be $\sum_{i \neq j} t_i^c - \sum_{i \neq j} t_i^*$. Note that $-\sum_i t_i \geq -t_j^r - \sum_{i \neq j} t_i^c$, otherwise the Hungarian algorithm would have chosen the assignment reached by j mis-reporting and cancelling. Also, note that $-\sum_{i \neq j} t_i^* \geq -\sum_{i \neq j} t_i$, for the same reason. Therefore, $\sum_{i \neq j} t_i^c - \sum_{i \neq j} t_i^* \geq \sum_i t_i - t_j^r - \sum_{i \neq j} t_i^* \geq t_j - t_j^r + \sum_{i \neq j} t_i - \sum_{i \neq j} t_i^* \geq t_i - t_j^r$. Since the cancellation fee is greater than or equal to his benefit, no agent will have an incentive to mis-report.

Note that because SCRAM does not minimize travel time for all users the fee is not guaranteed to be incentive compatible with SCRAM. Therefore, we leave a strategy-proof cancellation fee for SCRAM to future work.

7. CONCLUSION AND FUTURE WORK

We have presented a comparison of methods for assigning vehicles to users for carsharing with autonomous vehicles. Our experiments show that employing an optimal perfect matching algorithm can improve performance of a system in terms of reducing unoccupied travel, decreasing customer wait times,

[1] We assume that any practical system must allow for cancellation due to changing one's mind or circumstances changing.

and decreasing the variance in expected wait time. As expected, the Hungarian algorithm, by ensuring minimum cost matching, minimizes total wait and unoccupied distance traveled across the entire system. However, SCRAM, by minimizing the makespan, reduces variance in the user wait time and the number of users who have significant waits, making the system more "fair." Compared to the two simpler greedy approaches, both the Hungarian algorithm and SCRAM are significant improvements. While a centralized assignment algorithm optimizes the system, strategic users can still manipulate the system. We have provided a cancellation fee that provably removes incentive to manipulate the system for one algorithm, but more work is needed for other methods.

The current approach is myopic in that it only considers matching at the current timestep. Any of the presented methods could be adapted to consider vehicles that will soon be available. One straightforward method would be to represent the vehicle's distance to a passenger as the distance from the passenger to the vehicle destination plus the vehicle's remaining distance left to travel. In situations where the fleet size is limited this could allow the system to directly tell a user if it will be possible to serve him and how long they would have to wait. Integrating knowledge of trip completion times with predictions of demand could further optimize long-term performance of the system. Finally, prior work has considered rebalancing vehicles not in use. A limitation of the matching algorithm approach is that it ignores the need to maintain vehicle balance. We wish to extend the presented matching algorithms to maintain a supply of vehicles close to areas likely to have future user requests.

REFERENCES

Acquaviva, F., Nunez, A., Di Paola, D., Rizzo, A., and De Schutter, B. (2015). Customer-oriented optimal vehicle assignment in mobility-on-demand systems. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, 2849–2854. IEEE.

Chen, B. and Cheng, H.H. (2010). A review of the applications of agent technology in traffic and transportation systems. *Intelligent Transportation Systems, IEEE Transactions on*, 11(2), 485–497.

Chow, Y. and Yu, J.Y. (2015). Real-time bidding based vehicle sharing. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*.

Dresner, K. and Stone, P. (2008). A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31, 591–656.

Fagnant, D. and Kockelman, K. (2014). The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transportation Research Board Part C*, 40, 1–13.

Fagnant, D., Kockelman, K., and Bansal, P. (2015). Operations of a shared autonomous vehicle fleet for the austin, texas market. *Transportation Research Board*.

Hausknecht, M., Au, T.C., Stone, P., Fajardo, D., and Waller, T. (2011). Dynamic lane reversal in traffic management. In *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC)*.

Kamar, E. and Horvitz, E. (2009). Collaboration and shared plans in the open world: Studies of ridesharing. In *20th International Joint Conference on Artificial Intelligence (IJCAI)*.

Kleiner, A., Nebel, B., and Ziparo, V. (2011). A mechanism for dynamic ride sharing based on parallel auctions. In *22th International Joint Conference on Artificial Intelligence (IJCAI)*, 266–272.

Kuhn, H.W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2), 83–97.

Lee, D.H., Wu, X., and Sun, L. (2013). Limited information-sharing strategy for taxi-customer searching problem in nonbooking taxi service. *Transportation Research Record: Journal of the Transportation Research Board*, (2333), 46–54.

MacAlpine, P., Price, E., and Stone, P. (2015). Scram: Scalable collision-avoiding role assignment with minimal-makespan for formational positioning. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Seow, K.T., Dang, N.H., and Lee, D.H. (2007). Towards an automated multiagent taxi-dispatch system. In *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*, 1045–1050. IEEE.

Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1), 8–37.

Zhang, R. and Pavone, M. (2016). Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *The International Journal of Robotics Research*, 35(1-3), 186–203.

Zhang, R., Spieser, K., Frazzoli, E., and Pavone, M. (2015). Models, algorithms, and evaluation for autonomous mobility-on-demand systems. In *American Control Conference (ACC), 2015*, 2573–2587. IEEE.