2017 IEEE International Conference on Service Operations and Logistics, and Informatics
September 18-20, 2017
Bari, Italy

Online Scheduling of a Fleet of Autonomous Vehicles Using Agent-Based Procurement Auctions

Meghana Madhyastha
Computer Science
IIITB
Bangalore, India
Email: MeghanaMadhyastha@iiitb.org

Sriveda Chevuru Reddy
Computer Science
IIITB
Bangalore, India
Email: SrivedaReddy.Chevuru@iiitb.org

Shrisha Rao
Computer Science
IIITB
Bangalore, India
Email: srao@iiitb.ac.in

***Abstract*—We propose a novel distributed approach for the problem of scheduling a fleet of autonomous vehicles. Our distributed system avoids a single point of failure, is scalable, fault tolerant and robust. We describe an agent-based distributed system to conduct a set of procurement auctions. The vehicles are the "sellers" and the passengers are the "buyers" in the auction. Each vehicle bids for the passengers with a bid value which is an inverse function of the time it would take the vehicle to reach the passenger. In our agent based architecture, the various software agents reside on different systems and we describe distributed algorithms for their communication. We have performed simulations of a vehicle fleet in two different locations (Bangalore, India and Tyson's Corner, Virginia, USA) and compare the maximum and average waiting times of the passenger of our algorithm with a FIFO algorithm. We also compute the ratio of the time in which the vehicle is servicing a passenger to the total time to compute the fuel wastage. The results show that our system improves the maximum and average waiting times of the passengers, as well as the fuel costs for the vehicle fleet. Furthermore, we show that such a distributed system reduces the time it would take on average to respond to customer requests, as compared to a system which is not distributed.**

I. Introduction

Scheduling of vehicles given their locations and passenger arrival times in an optimal manner is an NP-hard problem and it is called the Dial-a-Ride Problem (DARP). DARP consists of designing vehicle routes and schedules for n passengers who specify pickup and delivery requests between origins and destinations. The aim is to plan a set of m minimum cost vehicle routes capable of accommodating as many passengers as possible, under a set of constraints. Nevertheless, there are various heuristics to schedule vehicles in a near optimal manner. However, we address the problem of online vehicle scheduling where we are not given the arrival times of the passengers apriori. We model the passenger's arrival as a Poisson arrival process. Our task is to dynamically schedule the vehicles (assign a vehicle to the passenger) in an optimal, efficient way so as to minimize the maximum waiting time of the passengers. The model we propose is a distributed model which is explained further in the subsequent sections.

While problems such as refueling the vehicles as well as assigning routes to vehicles servicing passengers efficiently have been addressed, the problem of matching passenger requests to vehicles to minimise passenger waiting time and fuel costs has not been explored specifically in the context of autonomous vehicles. The previous work in this area require prior information of the passenger requests and the vehicles in order to schedule vehicles to passenger requests. VIPAFLEET attempts to carry out the scheduling of autonomous vehicles but does so in a limited industrial site where the kind of activities that are taking place. Soloman [1] discusses and analysis various heuristics and approximation algorithms for scheduling vehicles. However, most of the existing literature on this topic focus on the scheduling problem. They formulate the DARP as a mixed linear programming problem (LPP) by defining an objective function followed by a set of constraints [2], [3].

However, with the advent of self driving cars (autonomous vehicles), where the vehicles are induced with some amount of "intelligence" and processing power, it is an instructive challenge to approach this problem as a distributed problem where each vehicle has its own processor. Bsaybes *et al.* [4] propose a preliminary framework for fleet management of autonomous vehicles called VIPAFLEET which can operate in three different modes: Tram mode, elevator mode and taxi mode. Attanasio *et al.* [5] propose a method for parallel implementations of a Tabu search heuristic for scheduling the vehicles in the dynamic DARP.

Our purpose is to design a distributed system that scheduling autonomous vehicles to passenger requests using a bidding system in which vehicles in the vicinity of a passenger can offer to service that passenger by bidding for the passenger. Whichever vehicle has the highest bid is scheduled to that passenger. In order to incorporate this distributed auction bidding for the purpose of vehicle scheduling we referred to Badica *et al.* [6], who model an English auction where a seller initiates an auction with certain parameters such as the minimum number of people that must participate, starting price and time limit for the auction. Potential buyers can bid their price for the item to intermediate agents called as "Proxy Agents." These Proxy Agents filter out buyers based on the local maximum bid and then send only relevant buyers' bids to the central auction manager. This decreases the load on the auction manager and the system can be scaled by increasing the number of Proxy Agents if the number of buyers increase.

The contributions of our paper are the following. First, while previous attempts formulate the problem as a linear programming problem which has a sub-exponential worst time message complexity, ours is a simple distributed approach that is scalable. Second, we extend the work of Badica *et al.* [6] and propose a novel, hierarchical representation of different agents including vehicles, proxy agent and auction manager agent to

978-1-5090-5847-1/17/$31.0

facilitate efficient communication between vehicles that have no knowledge of the fleet. We then propose a distributed multi-agent auction bidding architecture for the scheduling system.

The rest of the paper is structured as follows. In Section II, we explain some procurement auction theory that is used in the bidding of passengers. In Section III, we explain the system architecture for agent based procurement auctions. In Section IV, we explain our results and simulations. Finally, we conclude in Section V.

II. Procurement Auctions in Vehicle Bidding

Auction theory [7], [8] is a particular technique used in network economics [9], and finds applications in many domains, including electronic procurement [10] and cloud computing [11].

In a procurement auction the roles of the buyer and the sellers are reversed as the sellers compete to obtain business from the buyer. Unlike in a traditional auction, in a procurement auction (also called a reverse auction), it is the buyer who starts the auction [12], [13]. In our model, the "buyers" are the passenger and the "sellers" are the vehicles. A new auction takes place each time a passenger requests for a cab ride. The auction will take place for a stipulated time duration. The time duration is not necessarily a constant across auction but it depends on a variety of factors including the traffic conditions, the density of the traffic at the particular location, etc. Vehicles which are within a certain predefined radius of the passenger can "bid" for passengers. The bid value is a function of the estimated time it would take for the vehicle to reach the passenger. Furthermore, in our model, we divide the entire geographical area into different clusters. A vehicle can only participate in an auction in which it shares the same geographic cluster as the the passenger (the initiator of the auction). This means that a vehicle remains unoccupied until it wins an auction in its specific cluster. This is done to minimize the waiting time of the passengers.

One critical difference between the auction we model here and a traditional procurement auctions is that in a traditional procurement auction, the commodity has a monetary value and the bidders bid a price X currency to obtain the good. However, in this case, the vehicles bid for the product in terms of the estimated to reach the passenger. More specifically, let $\{V_1, V_2, ..., V_n\}$ be the set of vehicles contending for passenger P. The estimated waiting time of passenger p_i is denoted by T_i. We model our system as a first price sealed bid auction where the object is sold to the highest bidder at the highest bidder's price [14]. The value of the commodity (in this case the passenger) to V_i is denoted by $v_i = \frac{1}{T_i}$. We use the reciprocal of time to capture the intuition that the vehicle which takes the least time to reach the passenger should ideally win the auction. Say V_i bids b_i. We will derive the value of b_i so as to maximise the expected profit for the vehicle. We assume each vehicle bids with the same strategy so the system approaches Nash equilibrium.

$$Profit = (v_i - b_i)P(b_i) \tag{1}$$

where $P(b_i)$ is the probability that V_i wins the passenger if it bids b_i. $P(b_i)$ is an increasing function in b_i whereas $v_i - b_i$ is a decreasing function in b_i. So is a tradeoff. Let's assume that each vehicle V_i sees the other vehicles' values v_j as random variables drawn from the uniform distribution on the interval $[0, 1]$ which has a cumulative distribution function$F$ defined as follows.

$$F(x) = \begin{cases} 0 & \text{for } x < a \\ \frac{x-a}{b-a} & \text{for } a \leq x \leq b \\ 1 & \text{for } x > b \end{cases}$$

We first derive an expression for $P(b_i)$. Our derivation proceeds along the lines of Easley *et al.* in [14]. If there are n bidders, $P(b_i)$ is the probability that V_i is the highest bidder or the probability that all other vehicles bid a lower value that V_i's bid $= b_i$. Suppose each vehicle bid a fraction $1/c$ of V_i.

The probability that one of them bid less than b_i:

$$F\Big(\frac{v_i}{c}\Big) = \frac{v_i}{c} \tag{2}$$

The probability that all $n-1$ of them bid less than b_i:

$$F\Big(\frac{v_i}{c}\Big)^{n-1} = \Big(\frac{v_i}{c}\Big)^{n-1} \tag{3}$$

Substituting for $P(b_i)$ in (1) we get:

$$Profit = (v_i - b_i)\frac{v_i^{n-1}}{c^{n-1}}. \tag{4}$$

To maximise the expected profit we choose b_i such that $\frac{d}{db_i}((v_i - b_i)\frac{v_i^{n-1}}{c^{n-1}}) = 0$. We thus get

$$b_i = \frac{n-1}{n}v_i \tag{5}$$

Thus, each vehicle V_i must bid $\frac{n-1}{n}$ of its true value $v_i = \frac{1}{T_i}$. The variable bid in Algorithm 4 and Algorithm 5 stores this bid value and it is explained in detail in the next section.

III. System Architecture for Distributed Agent Based Procurement Auction

Mathematically, we model the road network as a graph $G(V, E)$ with a set of vertices V and edges E. The assumption we are making here are that the vertices represent the pick up and drop locations and the edges are the roads that connect them. We are disallowing pick ups and drops in the middle of edges. The passenger requests are modelled as tuples (A_t, S_v, D_v) where A_t is the arrival time, S_v is the source vertex and D_v is the destination vertex. We disallow ride sharing. Our aim is to schedule the vehicles in such a way so as to minimize the average waiting time of the customers.

We extend the work of Badica *et al.* [6] to design a distributed agent-based online system for procurement

auctions. The system consists of various types of software agents interacting with one another. A software agent can be defined as a software entity which has its own thread of control and can decide on the actions it wants to perform. It can be located on any arbitrary machine and it has an unique identifier. The various different agents we use in our system are $PersonalAgent$ ($VehiclePersonalAgent$ and $PassengerPersonalAgent$), $AuctionManager$, $MainManager$ and $Proxys$. The $VehiclePersonalAgent$ and $PassengerPersonalAgent$ represent a vehicle bidder and passenger respectively. The $AuctionManager$ is responsible for managing a single auction. The $MainManager$ has the overall responsibility of maintaining the auctions. The $Proxy$ agent serves as an intermediary agent between the $PersonalAgent$s and the $AuctionManager$.

We have divided our architecture into two different layers.

1) Interface Layer: The interface layer as the name suggests consists of all of the components that allows the user to participate with the main server where all the auction related processing is done. The following software agents are part of the interface layer.

- $PersonalAgent$**:**

  One $PersonalAgent$ is created for every user—vehicle or passenger. There is a one to one relationship between a vehicle/passenger and a $PersonalAgent$. A $PersonalAgent$ can take various roles. It can take the role of a vehicle participant who is essentially the seller in our procurement auction model. The agent in this case is called $VehiclePersonalAgent$. A personal agent can also play the role of the Initiator participant agent who in this case is the passenger and also the buyer in the auction.The Initiator participant agent is called the $PassengerPersonalAgent$. The $PassengerPersonalAgent$ initiates a travel request with certain parameters (location) to the $MainManager$ agent.

2) Core Layer: The core layer consists of $MainManager$, $AuctionManager$ and $Proxys$. This layer is responsible for the auctions management and for the coordination of the auction participants by implementing the rules that govern the auction.

- $MainManager$
  Each passenger request initiates the creation of a new auction. Clearly, this means that millions of auctions will take place at a given instance. $MainManager$ is responsible for managing all active auctions at a given instance across all possible clusters.
- $AuctionManager$**:**
  The $AuctionManager$ manages a single auction. Each auction has an $AuctionManger$. This agent is responsible for performing the following tasks.
  - creation and termination of the auction as well as recording the winner of the auction
  - decision on the duration of the auction
  - creation and destruction of proxy agents
  - assignment participant agents to the proxy agents
  - validation of participant eligibility based on their location i.e, if they are within a certain physical radius from the passenger
  - communication of the auction winners to the proxy agents
- $Proxy$**:**
  The participants (vehicles) across all auctions are split into disjoint groups. Each group is managed by a single $Proxy$ agent. $Proxy$ agents can be thought of as the intermediate layer between the auction manager and the participant agent. All communication between the participant agents and the auction managers happens through the $Proxy$ agent. The purpose for introducing these proxy agents is to reduce the load on the $AuctionManager$ by distributing the load across the $Proxy$ agents and also to avoid a single point of failure. The proxy agents manage a subset of vehicles and their bids. Their job is to filter our the bids received and forward the local highest bid to the auction manager. The auction manager then decides the highest bid globally and relay this information to the proxy agents who in turn informs all the participants part of their group. The participant agents self-destroy when the auction is over. It is up to the auction manager to assign the participants to the proxies to maintain the proxy rule.
  The usage of $ProxyAgents$ reduces the message complexity if distributed in an optimal manner. Assuming there are n participants in a single auction, it would require $O(n)$ number of messages for the auction manager to notify all the participants of any updates in the traditional centralized auction bidding model. However, in our distributed model, we can reduce this complexity. Assume there are p proxies and on average each proxy is managing k participants. Then clearly, $n = pxk$ and the message complexity is $O(p + k)$. This value can be optimized to $O(\sqrt{n})$ if we add the constraint that the number of participants handled by each proxy cannot outnumber the total number of participant agents. Essentially, this boils down to choosing p and k such that $p \approx k \approx \sqrt{n}$. The participant agents self-destroy when the auction is over. It is up to the auction manager to assign the participants to the proxies to maintain the proxy rule.

A. Agent Interactions

The $PassengerPersonalAgent$ sends a $createAuction$ request to the $MainManager$ which subsequently creates an $AuctionManager$ for the auction. When the vehicles are free and want to bid for a passenger, the $VehiclePersonalAgent$ sends a participate request to the $MainManager$ which then gets forwarded to the nearest $AuctionManager$. The $AuctionManager$ then checks if the vehicle belongs to its cluster and lets the $VehiclePersonalAgent$ know whether or not it is eligible to participate in the Auction. If the vehicle is allowed to participate then, $VehiclePersonalAgent$ is assigned to a proxy who then takes the bid. The $Proxy$ agent notifies the winning vehicle of the win and the vehicle subsequently proceeds to pick up the passenger. The auction

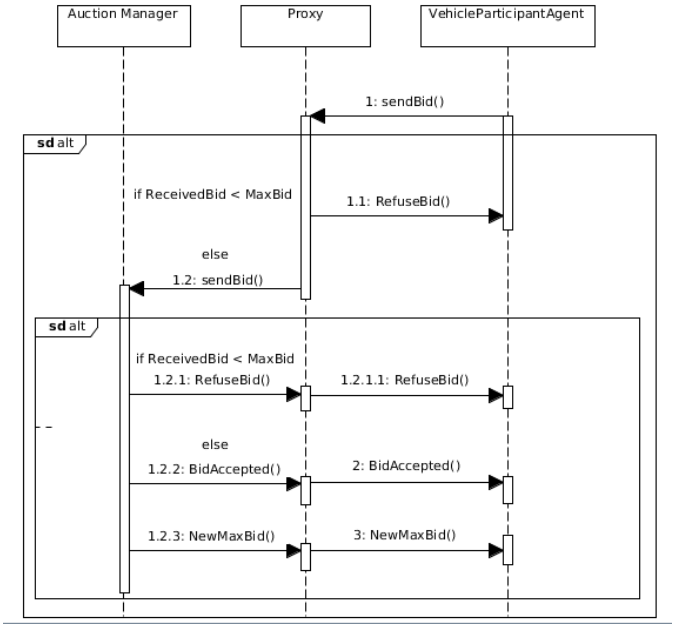

Fig. 1. Agent Interactions for the communication of bid values

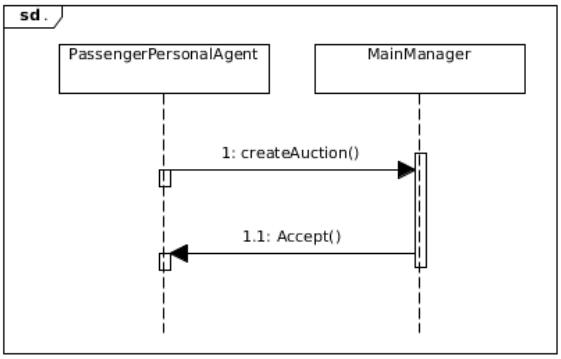

Fig. 2. Agent Interactions for the initiation of the auction

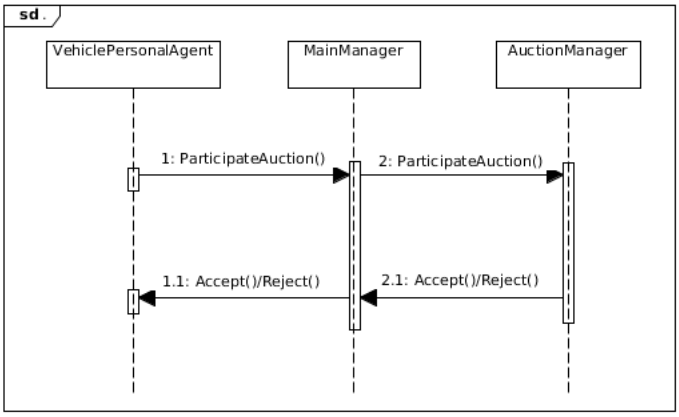

Fig. 3. Agent Interactions for participation in the auction

then ends and the agents self-destruct. The agent interactions are algorithmically given in the pseudocode (Algorithm 1-4).

. In Algorithm 1, the $PassengerPersonalAgent$ sends

Algorithm 1 Initiate Auction: $PassengerPersonalAgent$

1: send ⟨srcLocation, dstLocation, PassengerStartAuction ⟩ to $MainManager$
2: $AuctionManager$ gets created
3: $AuctionManager$ sends ⟨location, auctionNum⟩ to $Proxies$

a request to the $MainManager$ (line 1). This request results in the creation of $AuctionManager$ which implies that a new auction has been created. . Algorithm 2

Algorithm 2 Bid for Passenger: $VehiclePersonalAgent$

upon receiving ⟨location, Accept VehicleParticipateAuction⟩ :
1: bidvalue = generateBid()
2: send ⟨bidvalue, myid ⟩ to $Proxy$ upon receiving ⟨WINNER, V_{id} ⟩ :
3: **if** $V_{id} = myid$ **then**
4: pickup $Passenger$

explains the procedure for a vehicle bidding for a passenger. The $VehiclePersonalAgent$ first waits for the ⟨AcceptVehicleParticipateAuction⟩ from the $Proxy$ which means that it is now officially a part of the auction. After that, it generates its bid value = b_i as given in (5). If it gets a message form the $Proxy$ declaring it to be the winner of the auction, it proceeds to pick up the passenger.

.

Algorithm 3 $Proxy$

1: **for** i = 1 to $time$ **do**
2: upon receiving ⟨bid, V_{id} ⟩ from $VehiclePersonalAgent$:
3: **if** $bid > maxbid$ **then**
4: $maxbid \leftarrow bid$
5: send ⟨WINNER, V_{id} ⟩ to $MainManager$

Algorithm 3 explains the procedure the $Proxy$ has to follow. The $Proxy$ receives the bids from the $VehiclePersonalAgent$, filters them out to forward the locally maximum bid to the $AuctionManager$. .

Algorithm 4 $AuctionManager$

upon receiving ⟨location, VehicleParticipateAuction⟩:
1: **if** dist(location, proxyLocation)$< \delta$ **then**
2: send ⟨Accept VehicleParticipateAuction⟩ to $VehicleParticipantAgent$

3: $maxbid \leftarrow 0$
4: **for** i = 1 to $time$ **do**
5: upon receiving $\langle bid, V_{id} \rangle$ from $Proxy$:
6: **if** $bid > maxbid$ **then**
7: $maxbid \leftarrow bid$
8: send⟨ WINNER, V_{id}⟩ to $Proxy$

Algorithm 4 explains the role of the $AuctionManager$. The $AuctionManager$ has to first check whether the vehicle belongs to its cluster so it can participate in the auction. The $AuctionManager$ then receives the maximum bids from the $Proxy$ and relays the winner to the $Proxy$.

IV. Simulation and Results

We have simulated our model using the Mason and GeoMason Java Multi-agent Simulation toolkits. We have chosen two places, Tyson County, Virginia, USA and Bangalore, India, the former a small sparsely populated locality and the latter a densely populated city for our simulations. We ran a series of three experiments whose results are explained in subsections A, B and C.

The first experiment is to determine the average and maximum waiting times of the passengers when there are 100 vehicles in the fleet using our proposed distributed algorithm as well as with FIFO. The reason we chose to compare it with the FIFO is because cab service apps currently use a FIFO queue to decide who gets serviced first or service the nearest free vehicle. We vary the number of passengers from 0 to 2000 and plot the number of passengers versus the maximum and average waiting time for Bangalore and Virginia. Figures 1- 4 show these results.

The second experiment is to determine the average number of passengers served by a vehicle for every 100 kilometers travelled and compare the results in the case of the FIFO algorithm and the distributed algorithm. We then use this information to compute the percentage decrease in the cost spent on fuel per customer. We take into account census information to try to predict traffic conditions and allow the vehicles to move accordingly.

Finally, we perform experiments to determine the efficacy of the procurement auction system. We compute the latency of the system across different number of $Proxies$ and different number of passengers. Here, latency is defined as the average time it takes the system to answer a bid.

A. Analysis of Waiting Times of the Passenger

Figure 4 and Figure 5 respectively show how the Average and Maximum waiting times vary with the number of passengers for the FIFO algorithm and distributed scheduling algorithm in the simulation of Bangalore, India. We set the maximum speed limit to 50 kmph but the vehicles usually travel at a slower rate because of high traffic. Our analysis shows that our scheduling algorithm shows on average a 81 percent improvement in waiting times with the distributed scheduling algorithm.

Figure 6 and Figure 7 respectively show how the Average and Maximum waiting times vary with the number of passengers for the FIFO algorithm and distributed scheduling algorithm in the simulation of Tyson's District, Virginia, USA. We restrict the maximum speed limit to 72 kmph. A similar trend can be observed Our analysis shows that our scheduling algorithm shows on average a 73 percent improvement in waiting times with the distributed scheduling algorithm.

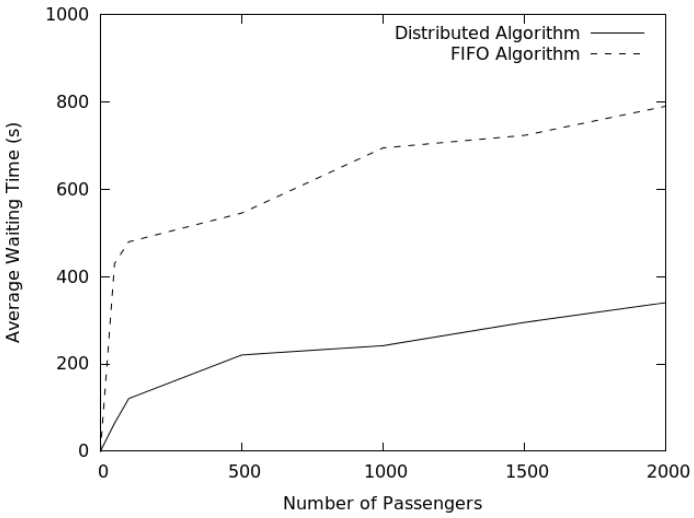

Fig. 4. Number of Passengers vs. Average Waiting Time(s) for Bangalore, India

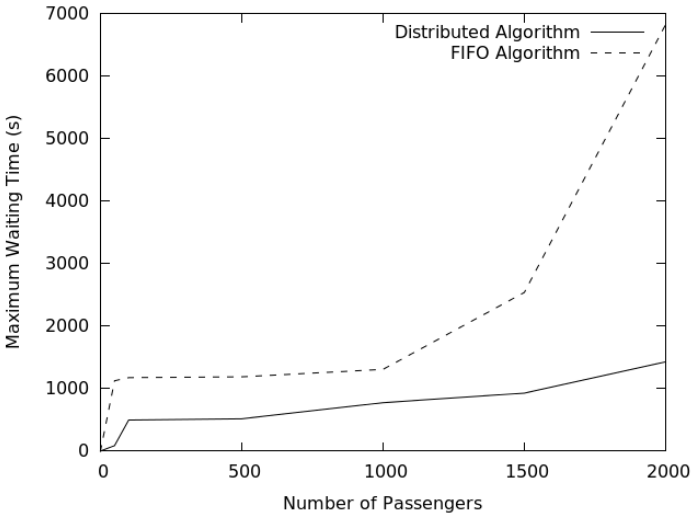

Fig. 5. Number of Passengers vs. Maximum Waiting Time(s) for Bangalore, India

B. Throughput and Cost Implication for the Owners of the Fleet of Vehicles

Place	FIFO Algorithm	Distributed Algorithm
Bangalore, India	0.32	0.11
Tyson's Corner, Virginia	0.14	0.08

TABLE I. Ratio of distance travelled with no passenger to the total distance travelled

Table I contains the average ratio of the distance travelled by a vehicle when it is not servicing a passenger to the total distance travelled by the vehicle in the case of the FIFO Algorithm as well as the case of the Distributed Algorithm. This has some important implications in terms of cost. Firstly, we would ideally want to minimize the time spent by the vehicle travelling on the road without a passenger because it

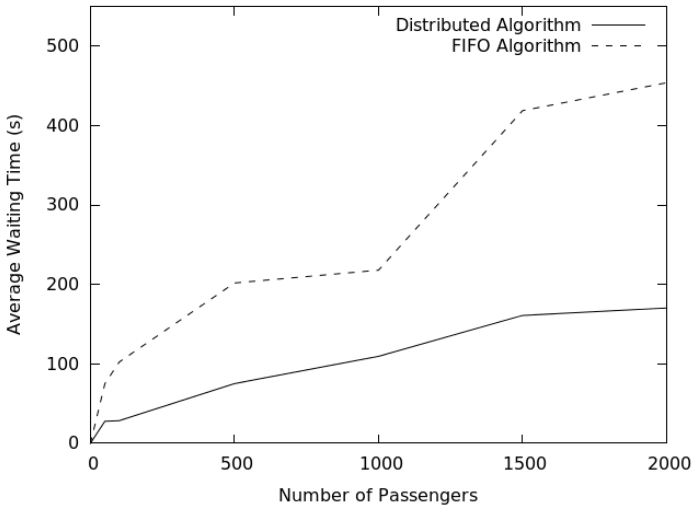

Fig. 6. Number of Passengers vs. Average Waiting Time(s) for Virginia, USA

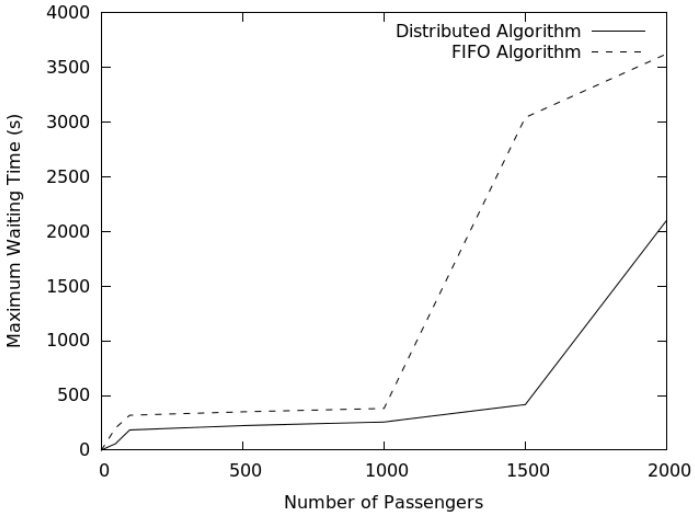

Fig. 7. Number of Passengers vs. Maximum Waiting Time(s) for Virginia, USA

would unnecessarily waste fuel. The global average cost of fuel per litre is $\$1.02/L$. If we consider a car with mileage $60kmpl$, this means that the cost incurred per km travelled is $\$0.017/km$. For example, if the vehicle travelled 1000 km in Bangalore, a ratio of 0.32 indicates that 320 km were travelled without a passenger in the car which amounts to a wastage of $\$5.44$ in fuel costs. We can observe from the graph that the ratios in the case of the Distributed Algorithm are lower than FIFO algorithm which implies lower fuel costs.

C. Analysis of Performance of the Procurement Auction

Number of Passengers	No Proxy	1 Proxy	2 Proxy
50	7304	5295	4060
100	7692	5966	4301
1000	8739	6022	4824

TABLE II. LATENCY (MS)

In a distributed environment with multiple vehicles bidding concurrently, a vehicle might end up getting outbid by another vehicle which submitted its bid almost simultaneously. We use the following performance metrics for determining the way the system handles a large number of bids. We vary the number of vehicles and the number of proxies and compute the latencies. Table II shows the results of the experiment. As we increase the number of $Proxies$ the performance becomes better as there is more parallelization so the latency reduces. As we increase the number of vehicles, the latency increases as the system has to handle a larger number of bids.

V. CONCLUSION

In this paper, we proposed a novel distributed multi-agent procurement auction system as well as a scheduling algorithm to assign vehicles to passengers as they arrive dynamically. Our approach is a simple distributed approach that is scalable. We explained the roles of the different software agents to model the vehicles and passengers in the multi-agent auction bidding architecture and to facilitate efficient communication between vehicles that have no knowledge of the fleet. Finally, we proposed a scheduling algorithm which incorporates traffic information in order to ensure that the vehicle that reaches the passenger the fastest is assigned to the passenger. The experimental results show that there is a significant improvement from the traditional FIFO queue methods and furthermore that the addition of $Proxy$ agents in our systems has significantly reduced the response time for answering a bid.

REFERENCES

[1] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research*, vol. 35, no. 2, pp. 254–265, 1987.

[2] A. Y. Lam, Y.-W. Leung, and X. Chu, "Autonomous-vehicle public transportation system: Scheduling and admission control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1210–1226, 2016.

[3] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem: models and algorithms," *Annals of Operations Research*, vol. 153, no. 1, pp. 29–46, 2007.

[4] S. Bsaybes, A. Quilliot, and A. K. Wagler, "Fleet management for autonomous vehicles," *CoRR*, vol. abs/1609.01634, 2016. [Online]. Available: http://arxiv.org/abs/1609.01634

[5] A. Attanasio, J.-F. Cordeau, G. Ghiani, and G. Laporte, "Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem," *Parallel Computing*, vol. 30, no. 3, pp. 377–387, 2004.

[6] C. Badica, S. Ilie, A. Muscar, A. Badica, L. Sandu, R. Sbora, M. Ganzha, and M. Paprzycki, "Distributed agent-based online auction system." *Computing and Informatics*, vol. 33, no. 3, pp. 518–552, 2014.

[7] R. P. McAfee and J. McMillan, "Auctions and bidding," *Journal of Economic Literature*, vol. 25, no. 2, pp. 699–738, Jun. 1987.

[8] S. Parsons, J. A. Rodriguez-Aguilar, and M. Klein, "Auctions and bidding: A guide for computer scientists," *ACM Computing Surveys*, vol. 43, no. 2, Jan. 2011, doi:10.1145/1883612.1883617.

[9] Y. Narahari, D. Garg, R. Narayanam, and H. Prakash, *Game Theoretic Problems in Network Economics and Mechanism Design Solutions*. Springer, 2009.

[10] T. S. Chandrashekar, Y. Narahari, C. H. Rosa, D. M. Kulkarni, J. D. Tew, and P. Dayama, "Auction-based mechanisms for electronic procurement," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, pp. 297–321, 2007.

[11] A. S. Prasad and S. Rao, "A mechanism design approach to resource procurement in cloud computing," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 17–30, Jan. 2014, doi:10.1109/TC.2013.106.

[12] A. Lunander, "Procurement bidding in first-price and second-price, sealed-bid auctions within the common-value paradigm," *Computational Economics*, vol. 19, no. 2, pp. 227–244, 2002.

[13] R. B. Myerson, "Optimal auction design," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 58–73, 1981.

[14] D. Easley and J. Kleinberg, *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.