Effective and Efficient Fleet Dispatching Strategies for Dynamically Matching AVs to Travelers in Large-scale Transportation Systems

Navjyoth J.S. Sarma, Daisik Nam, Michael F. Hyland (corresponding), Felipe de Souza, Dingtong Yang, Arash Ghaffar, I. Omer Verbas

Abstract— **This paper addresses the problem of dynamically matching automated vehicles (AVs) to open traveler requests in a large-scale automated-mobility-on-demand (AMOD) simulation framework. While optimization-based matching strategies based on the linear assignment problem formulation significantly outperform simple heuristic strategies (e.g. nearest neighbor), the scalability of the assignment problem limits its applicability to large problem instances. This study proposes a fleet dispatching strategy to dynamically assign AVs to travelers that involves the assignment problem formulation but restricts the decision space to reduce computational time. First, we significantly trim the decision space via only considering the *k*-closest open requests around each idle vehicle or *k*-closest idle vehicles around each open request. Second, we only calculate point-to-point shortest paths for vehicles and travelers that are close in spatial proximity. For vehicles and travelers that are not close in proximity, we use zone-to-zone travel time estimates. This study embeds the proposed AV fleet dispatching strategy within Polaris--an agent-based transportation simulation modeling framework. Within Polaris, the restricted fleet dispatching strategy proposed in this significantly outperforms (i) existing large-scale strategies in terms of fleet performance and (ii) the unrestricted assignment problem strategy in terms of computational performance.**

I. Introduction

The emergence and rapid growth of transportation network companies (TNCs, e.g. Uber, Lyft, Didi) have significantly disrupted passenger transportation systems in cities. TNC services offer significant benefits to travelers as a relatively affordable, high service quality travel mode that does not require purchasing or parking a vehicle. However, TNC services are also increasing vehicle miles traveled and traffic congestion due to various travel behavior impacts and operational inefficiencies.

Capturing the societal benefits and diminishing the negative outcomes of TNCs falls within the purview of transportation policymakers and planners. For example, there are plans to implement congestion pricing in Manhattan [1] and Manhattan already caps TNC vehicles [2]. Researchers have analyzed and compared other possible TNC regulations, including a driver minimum wage, a vehicle cap, and per-trip congestion tax [3].

Traditionally, transportation planning agencies, and to a lesser extent, policymakers have relied on models to understand the potential impacts of various technologies (e.g. electrification of vehicles), plans/designs (e.g. widening highways) and policies (e.g. congestion pricing and TNC vehicle caps) on transportation systems. The emergence and large presence of TNCs within transportation systems has presented a modeling challenge as traditional transportation system models only incorporated private vehicle and public transit modes.

Recently researchers have embedded fleet dispatching modules within transportation system models, including MATSim [4], Polaris [5] and SimMobility [6], to model the network and system impacts of TNCs as well as potential future automated-mobility-on-demand (AMOD) services, which are similar to TNC ridesourcing services except the vehicles are driverless, a fixed fleet size is typically assumed, and, the automated vehicles (AVs) are completely controlled by a central operator [7]. Polaris and MATSim both incorporate fleet dispatching modules and integrate them within a larger transportation system model that includes travel demand models and transportation network and control models.

Existing studies and transportation system models that incorporate fleet dispatching modules, and the fleet dispatching policies they employ, can be classified as either (i) large-scale simulations that employ simple heuristic strategies for dispatching [8], or (ii) small-medium-scale simulations that employ optimization-based strategies for dispatching [7].

The goal of this research study is to develop a computationally efficient yet effective (in terms of fleet performance) AV fleet operational strategy to assign AVs to traveler requests in large-scale transportation simulation models. The optimization-based strategy proposed in this study is based on the assignment problem formulation;

*Research supported by Argonne National Laboratory.

N.J.S. Sarma, D. Yang, and A. Ghaffar are PhD Students in Transportation Systems Engineering in the Civil and Environmental Engineering Department, University of California, Irvine, CA 92697 (email: nsarma.js@uci.edu, dingtony@uci.edu, ghaffaa1@uci.edu)

D. Nam is a postdoctoral researcher in the Civil and Environmental Engineering Department and the Institute of Transportation Studies, University of California, Irvine, CA 92697 USA (e-mail: daisikn@uci.edu).

M.F. Hyland is an Assistant Professor in the Civil and Environmental Engineering Department and the Institute of Transportation Studies, University of California, Irvine, CA 92697 USA (phone: 949-824-5084; e-mail: hylandm@uci.edu).

F. de Souza and I.O. Verbas are a postdoctoral appointee and a computational transportation engineer, respectively, at Argonne National Laboratory. Lemont, IL 60439 USA (e-mail: fdesouza@anl.gov, omer@anl.gov)

however, the proposed approach significantly decreases the size of the decision space using knowledge of the spatial distribution of AVs and open requests. Moreover, the proposed dispatching strategy intelligently populates the cost function in the assignment problem via only calling the shortest path module when AVs and travelers are close in proximity. The restricted assignment problem strategy proposed in this study is compared against two existing dispatching strategies in the literature for dynamically matching AVs to traveler request. The first is a rule-based dispatching strategy [9] and the second involves solving an unrestricted assignment problem. For comparison, this study embeds each of these fleet dispatching strategies within the fleet dispatching module in Polaris—an agent-based integrated activity-based travel demand and dynamic network assignment modeling software [10].

The rest of the paper is organized as follows: Section II describes the proposed assignment method. Section III discusses the proposed dispatching strategy within POLARIS and describes the efficiency and effectiveness evaluation measures. Section IV compares the proposed dispatching strategy against other dispatching strategies in terms of computational and fleet performance. Section V concludes the study and discusses future research directions.

II. METHODOLOGY

A. Problem statement

Given a set of trip requests P and a fleet of AVs V in time step τ, the problem objective is to match the AVs to the trip requests as efficiently and effectively as possible, within an agent-based simulation module. In this study, efficiency is measured in terms of computational time and effectiveness is measured in terms of fleet productivity and average pickup distance across the entire simulation period, not just at the current time interval τ.

B. Problem Formulation

Finding the optimal assignment for a set of requests P and a set of vehicles V involves constructing $n_p \times n_v$ cost objects and solving an optimization problem with $n_p \times n_v$ decision variables, where n_p and n_v denote the number of requests and vehicles, respectively. Hence, as the number of vehicles and passengers increase, the size of the decision space rapidly increases, such that large problems (e.g. 10,000 vehicles and 8,000 requests) take too long to solve in real-time. To address the computational efficiency problem this paper proposes several methods to reduce the decision space, while not significantly impacting the optimal solution.

As mentioned in the problem statement, at each time interval in the agent-based simulation model, the fleet dispatching module needs to assign AVs to open requests. To do this, this study proposes repeatedly solving a modified assignment problem with a reduced decision space until all there are no more feasible AV-traveler matches. The objective function of the modified assignment problem is formulated as follows:

$$min \sum_{\forall c} (T_c - M) * X_c \quad (1)$$

where

- $c=(p,v)$ denotes a feasible combination of an unassigned person p and an vehicle v in the reduced decision space (explained further in sub-section C).
- T_c denotes the travel time from vehicle v to person p in the feasible combination c.
- X_c is a binary decision variable indicating whether the feasible combination c is chosen for matching or not.
- M is the maximum pickup travel time from an unassigned vehicle v to unassigned person p--a constant set by the TNC vehicle operator

In the modified assignment problem, each request can be assigned to at most one vehicle.

$$\sum_{\forall c} X_c * P_{pc} \leq 1 \quad \forall p \quad (2)$$

where $P_{pc} = 1 \; if \; p \in c$ and 0 otherwise.

Moreover, each vehicle can be assigned to at most one request.

$$\sum_{\forall c} X_c * V_{vc} \leq 1 \quad \forall v \quad (3)$$

where $V_{vc} = 1 \; if \; v \in c$ and 0 otherwise.

To prevent conflicting constraints that may arise based on the spatial distribution of vehicles and requests it is necessary to set both request and vehicle assignment constraints as inequality constraints. An objective function that minimizes total cost with inequality assignment constraints would result in no matches being made with an objective value of 0. The presence of the maximum pickup travel time term M in the objective function ensures that the solver assigns requests to vehicles that are within the maximum pickup time, even with inequality constraints for both requests and vehicles.

Figure 1 shows an overview of the proposed vehicle-request matching algorithm executed at each time interval in the agent-based simulation model. The algorithm is divided into the following steps:

i. Input Pre-processing: This step involves finding the list of vehicles and requests eligible for assignment in each zone, and computing zone-zone cost matrices. Vehicles and requests that are unassigned at the beginning of a time step are eligible for assignment.

ii. Assignment Method Selection: This step involves choosing the method to build the restricted search space of feasible person-vehicle combinations to pass to the optimization problem. Based on the number of unassigned vehicles and requests (from Step i) The choice is between building combinations of either k nearest vehicles to all requests or k nearest requests to all vehicles.

iii. Optimization Pre-processing: This step involves finding the k nearest vehicles or requests based on the method chosen in the previous step and constructing cost objects for all unique person-vehicle combinations in the k nearest set. The travel cost between vehicles and requests is obtained by either indexing from a Zone

to Zone skim matrix, or by computing the point-to-point shortest paths between vehicles and travelers.

iv. Solving the Optimization Problem: This step involves solving one stage of the modified assignment problem with the reduce decision space for the feasible vehicle-traveler combinations from Step iii.

v. Post-processing: This step involves updating the statuses of assigned requests and vehicles, and updating the list of eligible requests and vehicles in each zone. If eligible vehicle-request combinations still exist, then go to Step i. If not, then the process is complete.

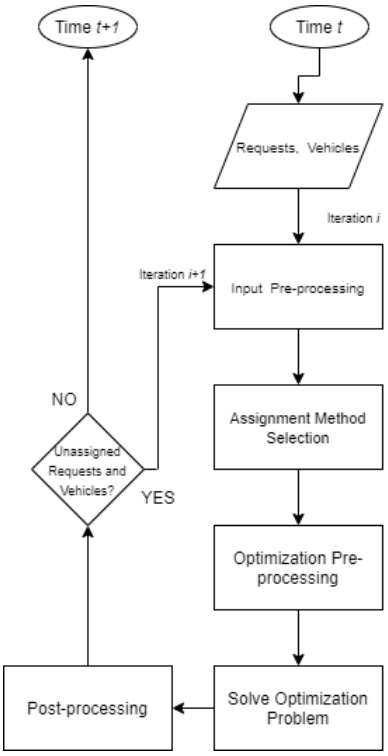

Figure 1: Overview of Request-Vehicle Matching Algorithm

C. Computation Time Reduction

Solving a matching problem with a reduced decision space of *k* nearest vehicles to all requests or *k* nearest requests to all vehicles reduces the number of decision variables in the optimization problem and hence the computation time and memory requirements. The overall computation time of the algorithm is influenced by the following three factors.

The first is the choice of Assignment Method. The number of requests and vehicles determine the entity (person or vehicle) around which *k* nearest search is to be performed to reduce the feasible decision space. Performing the *k* nearest search around the entity which is smaller in size results in fewer feasible person-vehicle combinations and hence takes less computation time and memory. Thus, if the number of requests is less than number of vehicles, a reduced decision space is constructed by finding the k nearest vehicles associated with each requests.

The second is building the heap of *k* nearest requests/vehicles. Finding *k* nearest requests to a vehicle (or vice versa), involves building a min heap from the request to all vehicles and popping the root of the heap *k* times. Building a heap of *n* objects takes *O(n)* time complexity. Each time the root of a heap is popped, it takes *O(log n)* time complexity to re-adjust the min heap to set a new root. Thus, finding *k* nearest vehicles to a single request takes a time complexity of $O(n+klogn)$.

In general, building a min heap for a request to all vehicles is a computationally intensive process especially when the number of vehicles is high. The computation time is reduced by incrementally constructing a min-heap from the request to all eligible vehicles within the same spatial zone and expanding the search to the next closest zone until a min heap with at least k vehicles to the request is built. This significantly reduces the time to find the *k* nearest entity to complexity $O(K+klogK)$, where *k* is a whole number typically in the range of 1 to 20 based on the search space, and *K* is the total number of eligible vehicles in each searched zone, starting from the request zone, $k<=K<=n$. This procedure requires zone-wise lists of eligible persons and vehicles to be initialized during pre-processing and updated at the end of each iteration of solving the optimization problem.

The third is building the vehicle-request cost objects. The cost term T_c in the objective function (Eqn. 1) can be found between each request and vehicle by either using zonal cost skims or by computing a point to point shortest path from the vehicle to the request. Zonal cost skims are pre-estimated and stored, hence accessing them is quite fast. However, using zonal cost skims may result in poor matches especially for vehicle-request assignments within the same zone or adjacent zones. Finding point to point costs for each feasible person-vehicle combination is computationally intensive, especially to build paths between persons and vehicles that are not close to each other. Hence, the proposed algorithm determines point to point costs only for feasible combinations that are within a zonal cost threshold. For all feasible combinations beyond the threshold, the inter-zonal cost is used to approximate the cost of travel between the vehicle and request. This proposed hybrid approach improves fleet performances by making better matches of feasible combinations that are close to each other and also leverages the time efficiency of zonal costs for other feasible combinations.

Figure 2 illustrates the procedures described above to efficiently find the *k* nearest vehicles to a single request and construct cost objects for the feasible request-vehicle combinations to be passed as an input to the optimization problem.

III. NUMERICAL EXAMPLE

A. Case Study

The proposed algorithm is applied in the city of Bloomington, Illinois, USA to match on-demand requests to vehicles. It is a medium-scale network that consists of 2,540 nodes, 7,023 links and 185 Traffic Analysis Zones (TAZs). The network also includes 2,833 activity locations that generate or attract trips. We assume that the fleet size is fixed at 1000 vehicles as current AV-based ridesourcing companies such as Waymo plan to operate the their own AV fleet [11]. Figure 3 shows the Bloomington network on which the assignment methods are run.

B. Agent-based simulation framework

The proposed ride-matching algorithm is implemented as a Dynamically Linked Library (DLL) to the Polaris agent-based modeling framework [10]. Polaris synthesizes the population and trips for the region for a 24-hour simulation period starting from 12 am. TNC trip requests are generated by running the mode choice model. The TNC vehicle fleet

size is set as a constant throughout the day and the vehicle locations and movements are tracked by Polaris in each time step. The proposed ride-matching DLL is invoked by Polaris in a 60 second interval, passing unassigned requests and all vehicle agents as arguments. The optimal ride matching DLL was developed in C++ making use of the CPLEX optimization package. The proposed algorithm is iteratively run in each time step until there are no feasible request-vehicle combinations remaining and the results are updated in Polaris.

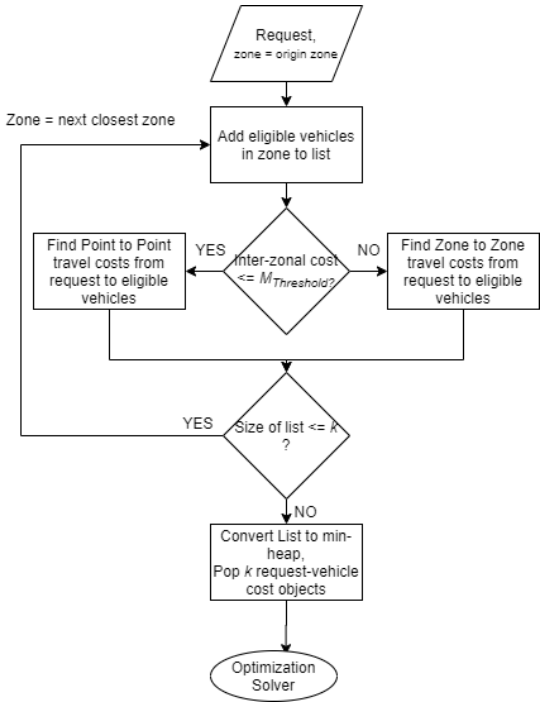

Figure 2: Finding k nearest feasible vehicles for a single request

C. Matching Strategies and Performance Measures

The study compares three assignment strategies:

i. A1 - Non-optimal assignment: Unassigned requests are matched to unassigned vehicles on a First Come First Serve (FCFS) basis in each time step using zonal skims. This is the algorithm currently in use in Polaris.
ii. A2 - Optimal K Nearest Hybrid with Zonal Skims: The algorithm proposed in Figure 2 is run by building feasible combination objects using zonal cost skims only.
iii. A3 – Optimal K Nearest Hybrid Assignment with Point to Point and Zonal Skims: The algorithm proposed in Figure 2 is run by using point-to-point costs for feasible vehicle-request pairs that are within a zonal cost threshold of 15% of maximum pickup cost *M* in equation *(1)*, and zonal skims for costs beyond the threshold.
iv. A4 – Global Optimal Assignment. This algorithm assigns requests to vehicles using Point to Point travel skims, without reducing the decision space [17].

The performance of the above algorithms is evaluated in terms of: *Total number of assigned travelers across* 24 hours*, Average Pickup Distance (feet), Polaris Run Time (seconds),* and *Fleet Productivity (%).* Polaris Run Time is the time it takes for Polaris to execute the entire simulation for the Bloomington network invoking the vehicle-request matching algorithm. Fleet Productivity is the percentage of TNC vehicle miles with a passenger on board. Table 1 shows an overview of the three assignment strategies compared in this study.

Figure 3: Bloomington, IL – USA Network

D. Model Scenarios

The request-vehicle matching algorithms listed in Table 1 are run for the following scenarios:

i. Base Scenario: TNC trips are generated based on the taxi mode choice parameters for Home Based Work, Home Based Other and Non-Home Based trips. The fleet size for the base model is 1000 vehicles. The entire fleet is in service during the 24-hour simulation period.
ii. Other scenarios: The assignment strategies A1, A2 and A3 are run for other scenarios with varying fleet size from 500 to 3000. Mode choice parameters are also changed for each fleet size scenario to capture system performance at different demand-supply imbalances.

Table 1: Overview of Matching algorithms

#	Algorithm	Optimization-based?	K-Nearest?	Cost Objects
A1	Non-Optimal [9]	No	NA	Zonal skims
A2	Optimal K Nearest Hybrid with Zonal Skims	Yes	Yes	Zonal Skims
A3	Optimal K Nearest Hybrid with Point to Point and Zonal Skims	Yes	Yes	Point-to-Point and Zonal Skims
A4	Unrestricted Assgn. Problem with Point-to-Point Skims [7]	Yes	No	Point-to-Point Skims

For all scenarios, the maximum pick up cost *(M)* in the objective function *(Eqn. 1)* is set as 20 minutes. A *k* value of 10 is used in the optimal K nearest hybrid strategies. An inter-zonal cost of 15% of *M* is used as threshold to switch from using point-to-point costs to zonal costs in the third assignment strategy (A3).

IV. Results and Inferences

A. Base Scenario

Table 2 shows the results of running the four assignment strategies for the base scenario.

Table 2: Evaluation of Matching methods for Base Scenario

Algorithm	A1	A2	A3	A4
Fleet Size	1000	1000	1000	1000
Assigned Travelers	92,162	101,504	103,464	105,934
Polaris Run Time (sec)	1104	1138	1291	92,960
Avg Pickup Distance (feet)	3,021	1,994	1,855	1,271
Fleet Productivity (%)	68.4	76.7	78.5	83.3

The assignment results show that the Optimal K Nearest hybrid algorithm with Point to Point and Zonal skims (A3) provides a significant improvement in terms of computational efficiency compared to the global optimal assignment algorithm A4. A3 reduces the total Polaris run time by around 98% compared to A4, and performs the best in terms of the average pickup distance, fleet productivity, and the total number of assignments when compared to A1 and A2. A3 reduces the average pick up distance by almost 40% compared to the non-optimal solution (A1), with a 16% increase in total computation time. Using point to point travel times for feasible combinations of nearby requests and vehicles reduces the average pickup distance by 7% compared to using zonal skims for all feasible combinations (A2). Algorithm A1 serves the fewest travelers over a 24-hour period in the base scenario. Aside from A4, A3 makes the most assignments and has the highest fleet productivity.

Even though Algorithm A4 perfoms the best in terms of average pickup distance, its very high Polaris Run Time of 92,690 seconds (25.8 hours), makes it unviable for use in dynamic assignment for large fleets. The proposed algorithm A3 provides a significant improvement in terms of computational efficiency compared to A4. Other computational tests illustrate that the fleet performance associated with A3 does not notably decrease relative to A4.

B. Other Scenarios

Figure 4 shows the Polaris run time associated with algorithms A1 (▲), A2 (♦), and A3(X) over different fleet sizes (500 to 3000 vehicles) and demand levels (40,000 to 175,000 riders). The plot shows that the matching algorithm A3, which yielded the best objective value in the base scenario, takes the most time to compute over different scenarios. It takes about 20% more time to run Polaris invoking the algorithm A3 compared to using strategy A1. The total computation time for A2 is only slightly more than A1 over different fleet sizes and demand.

For a given fleet size, the computation time of all algorithms increases with higher demand. Specifically, algorithm A3 requires more computation time as demand increases due to the need for higher resolution of the spatial distribution of both vehicles and requests. In addition, Polaris calculates point-to-point travel times from a vehicle to a location of the request by building a network path. In other words, Polaris only stores a zone-to-zone travel time skim matrix for efficient memory management and calculates point-to-point travel times on request. Since the point-to-point shortest path calls take significantly more computational time compared to indexing from an inter-zonal cost table, algorithm A3 consistently has a higher computation time compared to A2.

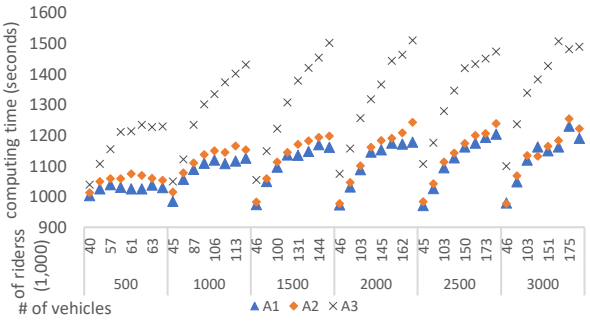

Figure 4: Comparison of Polaris Run Time

Figure 5 compares the average pickup distances (in feet) across the 3 assignment strategies for various fleet sizes and demand levels. The algorithm A3 performs the best of all algorithms across different fleet sizes and demands with consistently lower pickup distances.

Figure 5 also indicates that for a fixed fleet size, the average pickup distance decreases for the algorithms that use optimization (A2 and A3). Since assignments are made myopically in A1, average pick up distance does not tend to decrease with increasing demand for a given fleet size.

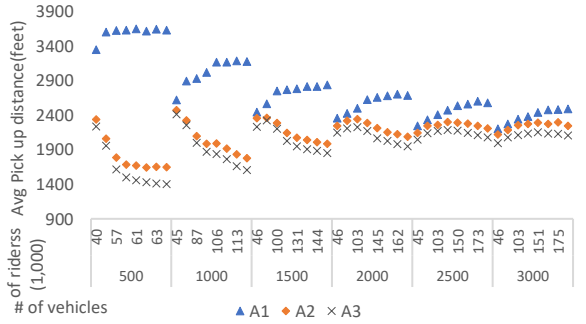

Figure 5: Comparison of Avg Pickup Distances

Figure 6 compares the fleet mile productivity across the assignment strategies over different scenarios of fleet size and demand. A3 consistently outperforms A1 and A2 in terms of productive fleet miles. A1 has the worst fleet productivity percentages across all scenarios. A3 has higher productivity than A2 since it makes better matching decisions for vehicles and requests that are close to each other by considering point-to-point travel costs instead of zonal skim approximations.

While the fleet productivity increases with increasing demand under fleet sizes for A2 and A3, this trend is not visible in very high fleet sizes (2500 and 3000). This trend could be because A2 and A3 do not yield a globally optimal solution (where all vehicle-request combinations are passed to the optimization solver). Especially in cases where the fleet size is very large, better matches may be missed out by just considering the *k* nearest vehicles or requests. In other

words, k value is a critical factor in addressing the spatiotemporal density of passengers and vehicles. Our experiments, however, set the constant k value over the scenarios.

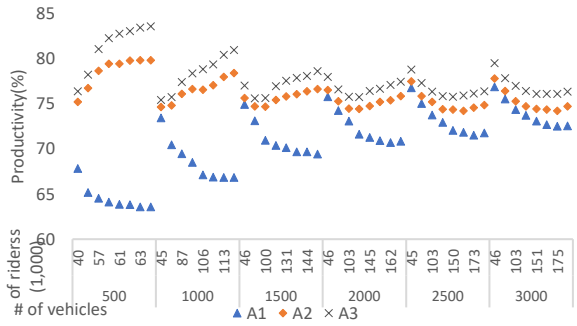

Figure 6: Comparison of Productive Fleet Miles (%)

The results in Figure 6 are particularly valuable from a transportation planning perspective. The policy implications associated with TNC/SAV fleets having a productivity rate around 75-85% are much different than TNC/SAV fleets having a productivity rate between 60-70%.

V. Conclusion

This study proposed a computationally efficient AV fleet dispatching strategy to dynamically match AVs to traveler requests within a large-scale agent-based travel demand-dynamic network simulation model. Computational efficiency was achieved by solving a restricted assignment problem considering a reduced decision space of k nearest elements. Applying the algorithm to a medium scale network of Bloomington indicated significant reduction in computation time compared to the global optimal solution, with improved fleet performance compared to a rule-based heuristic. The proposed fleet dispatching study is currently being tested on the very large-scale Chicago regional network with promising results in terms of both fleet and computational performance.

The study also found that the spatial distribution of requests and vehicles affect the number of assignments made in each iteration of the algorithm for a given value of k. Further research may explore setting of optimal value of k based on the spatio-temporal distribution of requests and vehicles, at each time interval. Another area of future research involves testing various regulations/policies related to TNCs in urban areas as well as evaluating the impact of TNCs at the network level (congestion) and system level (mode choice, activity location choice, etc.).

Acknowledgement

This research was sponsored by the U.S. Department of Energy Vehicle Technologies Office under the Systems and Modeling for Accelerated Research in Transportation Mobility Laboratory Consortium, an initiative of the Energy Efficient Mobility Systems Program. The authors remain solely responsible for this paper.

References

[1] W. Hu, "Confused About Congestion Pricing? Here's What We Know," *The New York Times*, New York City, 24-Apr-2019.

[2] T. Bellon, "Uber to limit drivers' app access to comply with NYC regulation," *Reuters*. 16-Sep-2019.

[3] S. Li, H. Tavafoghi, K. Poolla, and P. Varaiya, "Regulating TNCs: Should Uber and Lyft set their own rules?," *Transportation Research Part B: Methodological*, vol. 129, pp. 193–225, Nov. 2019.

[4] M. Maciejewski, J. Bischoff, S. Hörl, and K. Nagel, "Towards a testbed for dynamic vehicle routing algorithms," in *Communications in Computer and Information Science*, 2017, vol. 722, pp. 69–79.

[5] J. A. Auld *et al.*, "Exploring the mobility and energy implications of shared versus private autonomous vehicles," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1691–1696.

[6] R. Basu *et al.*, "Automated Mobility-on-Demand vs. Mass Transit: A Multi-Modal Activity-Driven Agent-Based Simulation Approach," *Transportation Research Record: Journal of the Transportation Research Board*, p. 036119811875863, May 2018.

[7] M. Hyland and H. S. Mahmassani, "Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 278–297, Jul. 2018.

[8] D. J. Fagnant, K. M. Kockelman, and P. Bansal, "Operations of Shared Autonomous Vehicle Fleet for Austin, Texas, Market," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2536, pp. 98–106, Sep. 2015.

[9] D. J. Fagnant and K. M. Kockelman, "The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 1–13, Mar. 2014.

[10] J. Auld, M. Hope, H. Ley, V. Sokolov, B. Xu, and K. Zhang, "POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations," *Transportation Research Part C: Emerging Technologies*, vol. 64, pp. 101–116, Mar. 2016.

[11] Waymo, "Waymo's fully self-driving vehicles are here," *Medium*, 2017. [Online]. Available: https://medium.com/waymo/with-waymo-in-the-drivers-seat-fully-self-driving-vehicles-can-transform-the-way-we-get-around-75e9622e829a. [Accessed: 07-Nov-2017].