Learn2Pool: Efficient and Effective Ride Assignment in Ride Sharing Systems

Ryan Cheng
University of Colorado Denver
Denver, Colorado, USA
ryan.cheng@ucdenver.edu

Selvakumar Jayaraman
University of Colorado Denver
Denver, Colorado, USA
selvakumar.jayaraman@ucdenver.edu

Robert Fitzgerald
University of Colorado Denver
Denver, Colorado, USA
robert.fitzgerald@ucdenver.edu

Farnoush Banaei-Kashani
University of Colorado Denver
Denver, Colorado, USA
farnoush.banaei-kashani@ucdenver.edu

ABSTRACT

Ride sharing systems enable people with similar itineraries and pick-up times to share their rides on the same vehicle. This has significant societal benefits as it reduces the number of vehicles used and hence reduces energy consumption and emissions to the environment. However, current systems do not fully address the potential of ride sharing as they do not utilize the transportation network effectively. A change in the ride assignment/scheduling algorithm to batch multiple requests has proven to be effective when extended time utilization is considered. Current batching based assignment methods generate all possible combinations of the trips for a given set of requests and vehicles, and then solve the assignment problem using integer linear programming (ILP). This becomes more computationally expensive as the combinatorial search space grows. In this paper, we propose overcoming this scalability problem by learning the reverse nearest neighbors order for a given location using pointer networks to effectively reduce the number of candidate trips considered in ILP. With extensive experimentation using real data, we show that our proposed solution, termed *Learn2Pool*, offers the most practical solution for ride assignment by allowing striking a balance between efficacy and efficiency, while demonstrating superior efficiency and efficacy across all existing methods.

CCS CONCEPTS

• **Information systems → Location based services**.

KEYWORDS

Ride sharing; Ride Assignment; Learning Priority

ACM Reference Format:
Ryan Cheng, Selvakumar Jayaraman, Robert Fitzgerald, and Farnoush Banaei-Kashani. 2022. Learn2Pool: Efficient and Effective Ride Assignment in Ride Sharing Systems. In *The 15th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS '22), November 1, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3557991.3567780

IWCTS '22, November 1, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9539-7/22/11.
https://doi.org/10.1145/3557991.3567780

1 INTRODUCTION

Ride-sharing systems enable people with similar itineraries and pick-up times to share their rides on the same vehicle. This has significant societal benefits as it reduces the number of vehicles used and hence reduces energy consumption and emissions to the environment. On platforms like Uber, Lyft, etc., where on-demand ride pooling service is provided, pooled rides maximize profit for the drivers while minimizing cost for the customers. However, optimally assigning vehicles to the requests in a ride sharing environment is a NP hard problem; there is no exact algorithm to solve the problem in polynomial time. Current systems either perform extensive computation using methods such as integer linear programming (ILP) to assign the vehicles to batched requests, or use other heuristics-based approaches that improve on time complexity but lose on performance measures such as total distance traveled, total waiting time, etc. This has motivated the development of learning-based systems that utilize neural networks to improve on computation time while maintaining system performance.

In particular, in [8, 11, 12, 14, 15] authors have proposed various insertion based approaches to solve the ride sharing assignment problem. With insertion based approaches, requests are handled one at a time sequentially as they arrive, and the vehicle assignment is filtered based on detour cost and assigned to a request. While insertion based methods reduce computation time for ride sharing assignment/scheduling, they are not effective since requests are assigned one at a time without considering other requests. On the other hand, batching based approaches [1–7, 9, 14] overcome this issue by waiting a certain amount of time and grouping requests and vehicles into batches, and then assigning vehicles to requests as a batch. This is formulated as an integer linear programming (ILP) problem with travel delay as the cost, and to be minimized with constraints such as maximum allowed waiting time. This approach improves performance but is computationally expensive as it generates a large number of trip combinations to be solved.

In this paper, we introduce *Learn2Pool* which deploys a pointer network [13] to prune the candidate trips before using ILP to find the optimized assignment. With effective pruning of the search space, our proposed solution enables efficient assignment while also

Table 1: Categorization of the related work based on various features

Paper	Batching	Insertion	Main Optimization Criteria						Optimization Strategy					Prediction Based	Indexing	Rebalancing
			Waiting Time	Distance	Computation Time	Service Rate	Shared Trips	Profit	ILP	DP	MDP	DRL	Other			
[2]	✓		✓	✓	✓				✓							
[8]		✓	✓	✓	✓	✓		✓					✓		✓	
[5]	✓		✓	✓	✓	✓							✓		✓	✓
[11]		✓	✓	✓		✓	✓		✓							✓
[1]	✓		✓	✓		✓	✓		✓							✓
[7]	✓		✓			✓		✓	✓							✓
[12]		✓	✓	✓	✓	✓		✓		✓						
[9]	✓		✓		✓				✓		✓	✓		✓	✓	
[4]	✓		✓			✓						✓				✓
[15]		✓	✓		✓	✓		✓	✓	✓						
[14]		✓	✓	✓	✓	✓		✓		✓				✓		
[6]	✓							✓			✓	✓		✓		
[3]	✓		✓		✓							✓			✓	
Our Paper	✓		✓	✓	✓	✓	✓		✓			✓			✓	

maintaining performance iterms of metrics such as total passenger waiting time, total vehicle travel time, etc.

In the rest of this paper, first we examine related work in Section 2. We then formally define the problem in Section 3 before we explain our methods in Section 4. Finally, we present our experimental evaluation of the proposed solution in Section 5, and conclude and discuss future directions in Section 6.

2 RELATED WORK

Several approaches exist in current literature to solve the ride sharing scheduling/assignment problem. These can be broadly categorized by insertion-based approaches that handle requests individually, and batching-based approaches that group requests and vehicles together to minimize the defined cost function. Related solutions can be further examined by their optimization strategy and problem formulation, most of which employ a form of integer linear programming (ILP), dynamic programming (DP), Markov decision process (MDP), deep reinforcement learning (DRL), or a combination of these to find the optimal assignment.

Table 1, summarizes the existing work in terms of the aforementioned capabilities and approaches. Below we further elaborate on the categorization offered in this table as well as related work in each category.

2.1 Insertion versus Batching

Insertion based approaches handle requests one at a time in a First Come First Serve manner. Vehicles are filtered based on detour cost and assigned to a ride request [8, 11, 12, 14, 15]. These approaches reduce waiting time for the passengers as well as computation time for assignment; however, they lose performance at the system-level with metrics such as total travel time for the vehicles.

Batching-based approaches overcome this issue by grouping requests and vehicles to improve on system-level performances requirements. Many of these approaches formulate the ride assignment problem as an integer linear programming (ILP) problem using the cost as the travel delay, and minimizing the cost by considering constraints such as maximum allowed waiting time [1–7, 9, 14]. While these approaches achieve better system-level performance, they are computationally expensive as large numbers of trip combinations are generated and assessed to identified the optimized assignment solutions.

2.2 Optimization Criteria

Different ride assignment solutions consider a variety of optimization criteria such as the total waiting time (as well as total travel time) for the passengers, total distance traveled by the vehicles, computation time to solve the assignment problem, service rate (i.e., ratio of the requests assigned within time-constrains for ride assignment), total number of shared trips (versus trips that are not shared), and the total profit for ride service. For the majority of solutions, waiting time is a primary optimization metric since the objective of an effective ride sharing system is to minimize the cost of assignment, which is the total delay caused for accommodating the shared trips. Distance traveled is another common metric, comparing the total or average distance increase of vehicles caused by shared trips. Computation time measures the efficiency of the optimization algorithm, which is paramount due to the application of ride sharing solutions in real-world settings. Service rate, which is the number of requests served, and the number of shared trips are also significant metrics for system optimality. Lastly, some methods define a pricing scheme to maximize driver profit while minimizing passenger expense.

2.3 Optimization Strategy

Another categorization of these solutions is based on the way a solutions formulates the ride assignment optimization problem and the strategy to address the problem. Many approaches incorporate the integer linear program (ILP) formulation of the ride assignment problem to provide optimal assignment for the multi-vehicle routing problem [1, 2, 7, 9, 11, 15]. Other solutions employ dynamic-programming (DP) based insertion to find optimal routes without enumerating all possible pairs for insertion [12, 14, 15]. Alternatively, other approaches model the problem as a Markov decision process (MDP) to find optimal solutions through a sequence of decisions [6, 9]. Lastly, some approaches opt to use a greedy lazy shortest path algorithm [8] or kinetic trees [5] to minimize computation time.

Heuristics-based methods have been shown to either lose performance on system-level metrics (such as total travel time) or require large computational resources to find the optimal assignment. Deep reinforcement learning (DRL) methods have been introduced to address these issues by leveraging neural networks to learn the optimal assignment policy [3, 4, 6, 9]. Results from these methods have been shown to maintain comparable system-level performance to

the exact optimization solutions while requiring less computation time.

2.4 Other Categories

Some solutions also take into account demand and supply based on the predicted distribution of upcoming vehicles and requests when calculating the optimal assignment [6, 9, 14]. Others make use of spatial or spatio-temporal indexing [3, 5, 8, 9] to quickly retrieve candidate vehicles for requests. Lastly, some approaches also take into account assignment rebalancing based on popular origin/destination stations (hot-spots) [1, 4, 5, 7, 11].

In this paper, we propose a batching based solution using the ILP formulation to find the optimal assignment. We employ a pointer network model to learn the reverse nearest neighbors order of requests in order to prune the combinatorially large search space and decrease computation time. Our solution also makes use of spatial indexing to quickly retrieve candidate vehicles for requests. As we explain in Section 5, system optimality is measured by the waiting time, distance, computation time, service rate, and number of shared trips as well as computation time in comparison to other methods.

3 PROBLEM DEFINITION

The ride sharing problem is a formulation of the NP-Hard Vehicle Routing Problem (VRP). This is a combinatorial optimization problem that has two main components: 1) assign passengers/ride-requests to vehicles, and 2) routing the vehicles from the origin to the destination location. The objective is to generate a sequence of route assignments for each vehicle such that one can maximize the number of passengers served and minimize the total cost of the system, typically measured by waiting time or distance traveled, respectively.

Formally, we consider a set of ride requests $R = \{r_1, ...r_n\}$, a set of vehicles $V = \{v_1, ...v_n\}$, and a function to compute travel time on the road network. We compute the optimal assignment $\sum$ of requests to vehicles that minimizes the cost function C while satisfying a number of constraints such as maximum waiting time, maximum incurred delay, etc. The cost C of an assignment $\sum$ is the sum of delays δ_r (i.e., the time it takes between receipt of a ride request r from a passenger to the pickup time of the passenger) over all assigned requests, plus a large constant c_{ko} as penalty for each unassigned request (i.e., the requests that cannot be assigned within the identified constraints for the assignment problem). The cost function we aim to minimize can be formalized as follows:

$$C(\sum) = \sum_{v \in V} \sum_{r \in P_v} \delta_r + \sum_{r \in R_{ko}} c_{ko}$$

where P_v is the set of ride requests assigned to vehicle v in assignment plan $\sum$, and R_{ko} represents the set of unassigned requests.

4 LEARN2POOL

Figure 1 shows the two-step process of ride assignment with *Learn2Pool*, our proposed efficient and effective solution for ride assignment in ride sharing systems. Below, we present the two components of Learn2Pool as depicted in the figure; note that the second component (i.e., ILP) is explained first to motivate the need for the first component.

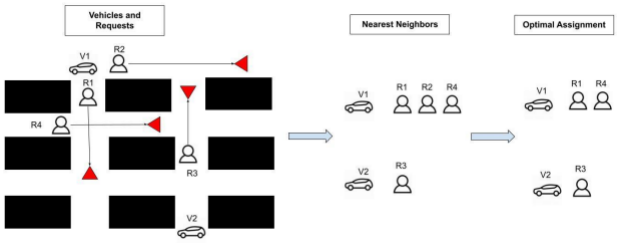

Figure 1: Learning Reverse Nearest Neighbors for Ride Sharing Assignment

4.1 Integer Linear Programming (ILP)

We use the Integer Linear Programming (ILP) formulation to find anytime optimal assignment for the multi-vehicle routing problem. This is done starting with the greedy solution and then optimized incrementally to solve the ILP. For greedy assignment, trips are assigned to vehicles iteratively in decreasing size of trip and increasing cost to maximimize the amount of requests served while minimizing cost. The optimization is then solved incrementally by applying the given cost function and constraints until a request is either assigned to a vehicle or ignored. Optimal assignment can be computed through an exhaustive search; however, this becomes computationally expensive with more vehicles and requests. In the next section, we explain how we use learning to reduce the time complexity of optimization with ILP.

4.2 Learning Priority Order of Ride Requests

To overcome the aforementioned scale-up issue with ILP, we learn the optimized order among a set of reverse nearest neighbor (RNN) ride requests for each vehicle. The optimized order is then considered as the priority list to reduce the combinatorially large search space of ILP problem explained in Section 4.1. Learn2Pool learns the optimized order of reverse nearest pick-up/request locations for a given vehicle location in a supervised manner by using simulation data for training the model (see Algorithm 1).

More specifically, to learn the order of reverse nearest neighbors/requests, we use the pointer network model [13], which is designed to learn the optimal output sequence where order and size of the output are dependent on the order and size of the input. In our case, input data is featurized as a list of a vehicle locations, each followed by the pickup locations of requests, denoted by $[v_i, r_1, r_2, \ldots r_n]$. The true output is computed by calculating the distances of all requests from the vehicle and finding the order of nearest requests from the vehicle's location. The model is trained from 5000 simulated input/output samples using a naive nearest neighbors computation with the goal of minimizing categorical cross-entropy loss between the true sequences and predicted sequences. Below we further elaborate on the training data preparation as well as the model used for learning the priority order of requests.

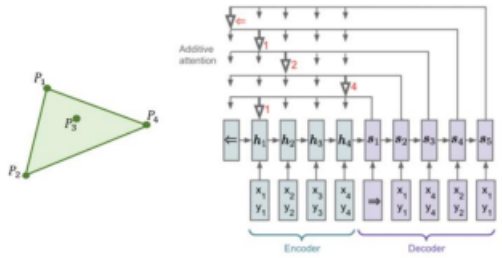

Figure 2: Pointer Network Model

Algorithm 1: NearestNeighborsILP

```
1. Input: vehicles, requests
2. nearest_neighbors = {}
3. for each v in vehicles:
4. nearest_neighbors.add(NearestNeighbors(v, requests))
5. end for
6. assignments = ILP(v, nearest_neighbors)
7. output assignments
```

4.2.1 Data Preparation. Real trips from the New York City Taxicab public dataset [10] was used for simulations. Raw latitude-longitude locations are normalized to be within a range of $[0, 1]$. As mentioned previously, the input is featurized as a list of a vehicle location followed by pickup locations of requests denoted by $[v_i, r_1, r_2, \ldots r_n]$, where n is the size of the reverse nearest neighbor set considered. Since the first element of the input is always the vehicle's location, the first index of the pointer network output is expected to be always 0.

4.2.2 Model Training. The pointer network model (see Figure 2) consists of an encoder-decoder architecture made up of RNN-based layers of LSTM. The hidden size of the pointer network is set to 128, and the model is trained for different epoch lengths ranging from 500 to 2000 with a learning rate of 0.1.

5 EXPERIMENTAL EVALUATION

We have performed extensive experimentation to compare Learn2Pool versus existing work in terms of both efficiency and efficacy. In this section, we first discuss our experimental methodology, and then review our experimental results.

5.1 Experimental Methodology

For training and testing, the New York City Taxicab public dataset [10] was used to simulate requests and vehicle location using data points from real trips in Manhattan. Vehicle capacity is set to 4 throughout the simulation. A machine with 2.7 Ghz dual core Intel i5 processor and 16 GB of RAM was used for training and testing.

The following methods were considered as related work for comparative study agaist Learn2Pool (referenced as *Pointer+ILP* in the figures for clarity): *Greedy*, *Prune Greedy DP*, *Batch ILP*, *Grid+ILP*, *Deep Q Network (DQN)*, and *Q-Mix*. The greedy approach was used as a baseline, and works by greedily assigning the nearest vehicle to a request in an insertion-based manner as requests arrive. Prune Greedy DP is a dynamic programming-based insertion approach introduced by Wang et. al [14]. Batch ILP is a batching-based ILP approach that does not prune the search space [2][1][7]. Grid+ILP, introduced by Ma et. al [8], is a method that divides the spatial map into an equal number of grids and uses a framework to prune vehicles for a given request. Lastly, two reinforcement learning approaches were tested; the first uses Deep Q-Networks [4] and the second using the Q-Mix model introduced by Rashid et. al [13].

Methods were evaluated on the basis of total and average distance traveled by vehicles, total and average waiting time of passengers, total computation time for assignment plan generation, actual total waiting time of passengers (which includes the assignment computation time along with wait time for pickup for each passenger), service rate, and total number of shared trips. We tested three parameter configurations with varying numbers of requests and vehicles per batch, for 10 batches. Firstly, we tested a configuration with 50 requests and 10 vehicles per batch. Second, we tested 50 requests and 20 vehicles per batch. Third, we tested 30 requests and 10 vehicles per batch. This was done to evaluate the effect of different parameters on the system efficacy as well as computation time. Lastly, a sensitivity analysis was conducted to assess the trade off between efficacy and efficiency by varying numbers of nearest neighbors n considered in the pruning process with Learn2Pool.

5.2 Experimental Results

Tables 2, 3, and 4 show results of our comparative study for the 3 configurations across all methods and metrics as described in Section 5.1; respectively, Figures 3, 4, and 5 visualize most important metrics reported in the tabular data for ease of interpretation.

Across all configurations we observe the following recurring patterns. As expected, the *Greedy* method which spends minimal time for assignment computation (as it does not consider batch assignment) performs worst in terms of waiting time, making it an unacceptable option despite the low assignment time as well as high service rate and number of shared trips. The *Prune Greedy DP* method uses dynamic programming to achieve improved waiting time at the cost of huge increase in the distance traveled (performing significantly worse than all other methods); hence, equally unacceptable. The ILP based solutions (including our solution) achieve a balanced performance in terms of distance traveled and waiting time, with *Pointer+ILP* (i.e., Learn2Pool) offering the best waiting time with lower computation time due to the proposed learning mechanism for pruning the search space. Finally, considering the two methods that leverage reinforcement learning for optimization, while *DQN* achieves the lowest distance traveled, waiting time and computation time among all methods, this is achieved at the cost of poor service rate and number of shared trips, with worst performance across all methods; hence, impractical to use. Among all related work considered in this study, *Q-Mix*, alike *Pointer+ILP* (i.e., Learn2Pool) strikes a desirable balance between efficacy and efficiency metrics, offering good performance across all metrics; however, this method is outperformed by *Pointer+ILP* in terms of all efficiency and efficacy metrics, particularly when larger request and vehicle batch sizes are considered with *Pointer+ILP*. We conclude that our proposed solution, Learn2Pool, can strike the right

Table 2: Configuration 1 - 50 requests and 10 vehicles per batch, for 10 batches

Metric	Greedy	Prune Greedy DP	Batch ILP	Grid + ILP	Pointer + ILP	DQN	Q-Mix
Total distance (miles)	172	300	101	115	113	49	124
Total waiting time (mins)	168	14	17	21	20	13	37
Total computation time (mins)	0.05	0.16	11	11	6	0.16	0.13
Actual total waiting time (mins)	168	14	28	32	26	13	37
Average distance (miles)	2.02	4.4	1.18	1.32	1.25	1.22	1.25
Average waiting time (mins)	1.97	0.2	0.32	0.37	0.28	0.32	0.37
Service rate (%)	51	54.4	42	45	45.75	22	45.4
Number of shared trips	85	68	78	79	78	29	77

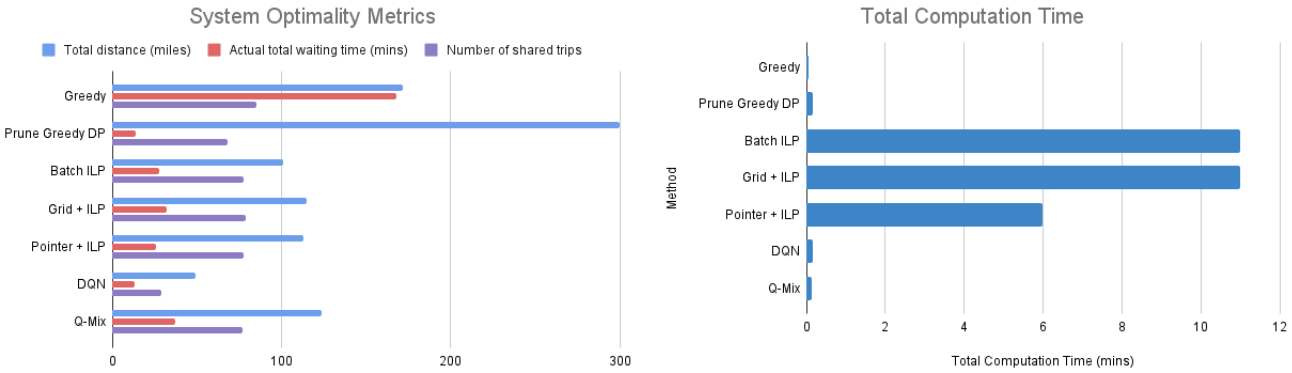

Figure 3: Visualization of selected metrics for Configuration 1

Table 3: Configuration 2 - 50 requests and 20 vehicles per batch, for 10 batches

Metric	Greedy	Prune Greedy DP	Batch ILP	Grid + ILP	Pointer + ILP	DQN	Q-Mix
Total distance (miles)	251	394	118	121	124	52	142
Total waiting time (mins)	241	26	15	14	17	11	51
Total computation time (mins)	0.05	0.16	26	22	18	0.16	0.13
Actual total waiting time (mins)	241	26	41	36	35	11	51
Average distance (miles)	1.62	2.02	0.92	0.91	0.94	1.08	1.22
Average waiting time (mins)	1.56	0.13	0.32	0.27	0.26	0.22	0.43
Service rate	71.20%	94.40%	58.50%	57.25%	59.75%	34%	71.50%
Number of shared trips	124	86	91	88	98	41	86

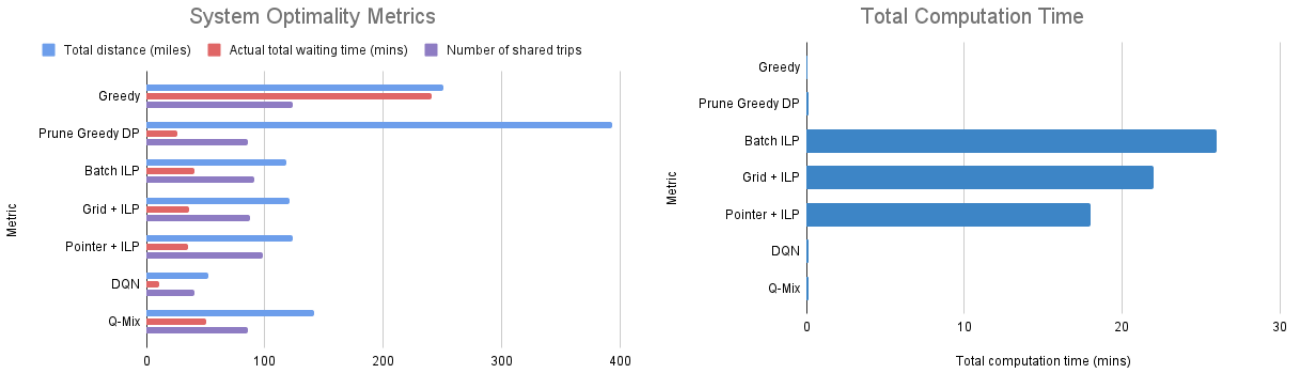

Figure 4: Visualization of selected metrics for Configuration 2

Table 4: Configuration 3 - 30 requests and 10 vehicles per batch, for 10 batches

Metric	Greedy	Prune Greedy DP	Batch ILP	Grid + ILP	Pointer + ILP	DQN	Q-Mix
Total distance (miles)	136	202	46	36	38	40	66
Total waiting time (mins)	145	14	8	7	7	12	23
Total computation time (mins)	0.05	0.16	1.58	1.45	1.11	0.13	0.13
Actual total waiting time (mins)	145	14	9	8	8	12	23
Average distance (miles)	2.34	0.93	0.8	0.8	0.8	1.37	1.2
Average waiting time (mins)	2.5	0.06	0.15	0.17	0.17	0.41	0.41
Service rate	58.00%	71.60%	43.50%	37.50%	37.50%	26%	46.30%
Number of shared trips	58	45	29	27	27	22	46

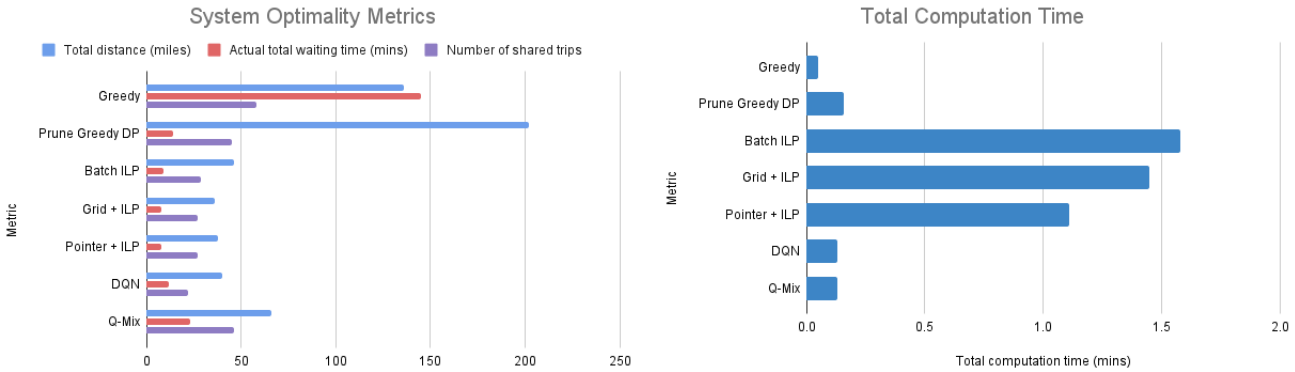

Figure 5: Visualization of selected metrics for Configuration 3

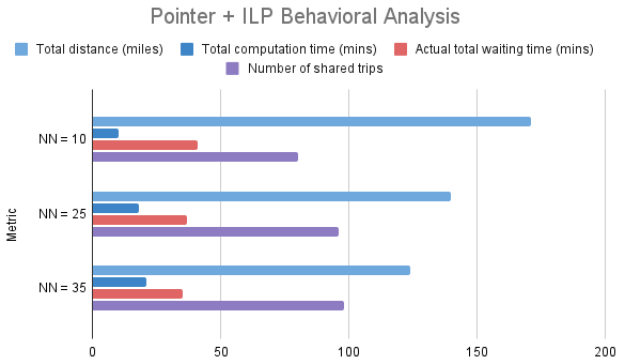

Figure 6: Pointer + ILP (i.e., Learn2Pool) sensitivity to size of Nearest Neighbor (NN) set

balance between efficacy and efficiency performance (a crucial requirement for applicability of the method), while outperforming other methods that can achieve this balance in terms of all efficacy and efficiency measures.

Figure 6 shows results of our sensitivity analysis based the size of the nearest neighbor (NN) set $|NN| = n$ used for learning the request priorities in Learn2Pool. As shown in the figure, by increasing n, as expected while the computation time for ride assignment increases, the efficacy measures all improve as Learn2Pool can more extensively explore optimized priorities. We conclude that in practice, one can use n to strike a balance between efficiency and efficacy as required in real-world use-cases.

6 CONCLUSION AND FUTURE WORK

In this paper we introduce a learning based solution for ride assignment problem in ride sharing systems that offers the most practical solution by allowing striking a balance between efficacy and efficiency, while demonstrating superior efficiency and efficacy across all existing methods.

In the future, we plan to use transfer learning to learn optimal assignment policy for groups of similar road networks. Additionally, other aspects such as weight assignment based on predicted demand and supply as well as re-balancing based on popular origin/destination locations could be added to the methods to accommodate more real-world instances.

ACKNOWLEDGEMENT

The work presented in this paper conducted with support from University of Colorado Denver and the Mountain-Plains Consortium, a University Transportation Center funded by the U.S. Department of Transportation. The contents of this paper reflect the views of the authors, who are responsible for the facts and accuracy of the information presented.

REFERENCES

[1] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences* 114, 3 (2017), 462–467. https://doi.org/10.1073/pnas.1611675114 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.1611675114

[2] Jean-François Cordeau. 2006. A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. *Oper. Res.* 54 (2006), 573–586.

[3] Oscar De Lima, Hansal Shah, Ting-Sheng Chu, and Brian Fogelson. 2020. Efficient Ridesharing Dispatch Using Multi-Agent Reinforcement Learning.

[4] John Holler, Risto Vuorio, Zhiwei (Tony) Qin, Xiaocheng Tang, Yan Jiao, Tiancheng Jin, Satinder Singh, Chenxi Wang, and Jieping Ye. 2019. Deep Reinforcement Learning for Multi-driver Vehicle Dispatching and Repositioning Problem. 1090–1095. https://doi.org/10.1109/ICDM.2019.00129

[5] Yan Huang, Favyen Bastani, Ruoming Jin, and Xiaoyang Sean Wang. 2014. Large Scale Real-Time Ridesharing with Service Guarantee on Road Networks. *Proc. VLDB Endow.* 7, 14 (oct 2014), 2017–2028. https://doi.org/10.14778/2733085.2733106

[6] Yan Jiao, Xiaocheng Tang, Zhiwei (Tony) Qin, Shuaiji Li, Fan Zhang, Hongtu Zhu, and Jieping Ye. 2020. A Deep Value-based Policy Search Approach for Real-world Vehicle Repositioning on Mobility-on-Demand Platforms.

[7] Michael W. Levin. 2017. Congestion-aware system optimal route choice for shared autonomous vehicles. *Transportation Research Part C: Emerging Technologies* 82 (2017), 229–247. https://doi.org/10.1016/j.trc.2017.06.020

[8] Shuo Ma, Yu Zheng, and Ouri Wolfson. 2013. T-share: A large-scale dynamic taxi ridesharing service. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. 410–421. https://doi.org/10.1109/ICDE.2013.6544843

[9] Mohammadreza Nazari, Afshin Oroojlooy, Martin Takáč, and Lawrence V. Snyder. 2018. Reinforcement Learning for Solving the Vehicle Routing Problem. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 9861–9871.

[10] Taxi New York (N.Y.) and Limousine Commission. 2019. New York City Taxi Trip Data, 2009-2018. https://doi.org/10.3886/ICPSR37254.v1

[11] Samitha Samaranayake, K. Spieser, Harshith Guntha, and Emilio Frazzoli. 2017. Ridepooling with trip-chaining in a shared-vehicle mobility-on-demand system. 1–7. https://doi.org/10.1109/ITSC.2017.8317603

[12] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. 2018. A Unified Approach to Route Planning for Shared Mobility. *Proc. VLDB Endow.* 11, 11 (jul 2018), 1633–1646. https://doi.org/10.14778/3236187.3236211

[13] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf

[14] Jiachuan Wang, Peng Cheng, Libin Zheng, Chao Feng, Lei Chen, Xuemin Lin, and Zheng Wang. 2020. Demand-Aware Route Planning for Shared Mobility Services. *Proc. VLDB Endow.* 13, 7 (mar 2020), 979–991. https://doi.org/10.14778/3384345.3384348

[15] Xian Yu and Siqian Shen. 2020. An Integrated Decomposition and Approximate Dynamic Programming Approach for On-Demand Ride Pooling. *IEEE Transactions on Intelligent Transportation Systems* 21, 9 (2020), 3811–3820. https://doi.org/10.1109/TITS.2019.2934423