Rethinking Operational Research: Learning policies for on-demand delivery services with supervised learning

Natalie Bolón Brun

January 2020

We tackle an on-demand delivery service problem from a supervised deep learning perspective in order to explore the capabilities of learning a target policy. We consider the special case of dial-a-ride problem with dynamic demand using electric vehicles and allowing shared rides. We will explore different representations of the environment state as well as different architectures and the effect of the different parameters in the final performance. To evaluate the quality of the results, we will compare them in terms of total service time from the customer's perspective and ride time from the vehicle's perspective against the times obtained using the assignments provided by the target policy.

1 Introduction

Every year on demand delivery services become more and more common among customers thanks to the comfort they provide and an easy access through an increase variety of web and apps. Different companies provide services dispatching goods that range from food (Glovo) to people (Uber) but that share a common point: a dynamic demand.

In a static environment the demand in known before hand, transforming the problem into a vehicle routing problem [6]. In such a case, an optimal route must be designed using the available resources to minimize an objective while meeting the different constraints. This will be the case of classic delivery services such as the post service where the postman has to deliver a set of goods with a known destination and therefore can plan an optimal route with this information. Nevertheless, in a dynamic setting, the demand is not known beforehand but reveled only under a new customer request. This makes the planning a harder task since future requests cannot be taken into account at the current moment of handling a new task.

In this case, the problem we want to tackle consist on the transportation of customers within a time window and facing a dynamic demand. These kind of problems (DARP) are usually solved using branch-and-cut algorithms [7] or variants of it. Although they allow to find optimal solutions, these methods usually increase their complexity with the size of the problem, making them non scalable. Moreover, we will be taking into account the resulting times from both the clients and vehicles perspective, using a multi-objective formulation.

Our aim is to explore alternative methods using supervised learning to obtain a more scalable system. Results in [8] show the possibility to learn structured representation of policies using supervised and reinforcement learning in a stochastic environment. Following these results we will explore how these methods adapt to learning more complex policies.

We will work on a problem with an autonomous electric fleet of vehicles, handling a dynamic demand in a stochastic environment and allowing sharing rides. However, we will not consider constraints related to the battery of the vehicle which were initially considered in the policy we aim to learn.

2 Related Word

In dial-a-ride problems, we aim to design vehicle routes and schedules that allow to satisfy the demand for as many customers as possible while minimizing the operational cost. This cost can be defined either in terms of time or from an economic perspective. Actually, the consideration of the human component introduces the clients' requirements in terms of time and satisfaction as part of the objective [5] creating a difference between DARP and other vehicle routing problems. To tackle it, different formulations around the classical optimization set up have been proposed [2].

From a deep learning perspective, recent results have shown a promising path to solve offline routing problems. Vinyals et al. (2015) [9] showed the potential to solve discrete combinatorial problems with the pointer network architecture. Their approach allows to generalize to variable output size by using attention as a pointer to select a member of the input sequence as the output.

In [1], Alabbasi et al. (2019) tackle the ride-sharing problem and propose a distributed model-free algorithm using deep reinforcement learning. They learn optimal assignment policies using deep Q-network techniques to learn the environment. By incorporating travel statistics and demand predictions using deep learning models they achieve to learn strategies that can adapt rapidly to changing environments and perform better that other strategies that do not allow sharing rides or exploit future demand predictions.

In his work, T. Stocco [8] explores different representations of a pickup and delivery problem with online demand and a stochastic environment in order to learn stochastic policies using supervised and reinforcement learning. By using image representation they achieve more than 30% accuracy on learning hand-crafted assignment policies and more than 75% accuracy on learning a nearest neighbor policy in a supervised way. Later, the learnt policy is improved by introducing reinforce with the best policy as feedback. The reinforce allows to explore new strategies and over-perform in some cases the deterministic case.

Finally, the policy we aim to learn is proposed by Claudia Bongiovanni on her work "A Learning Large Neighborhood Search for The Dynamic Electric Autonomous Dial-A-Ride" [4]. The proposed method works in two-stages: insertion and improvement. The first stage handles dynamic requests by first checking feasibility of the request for the different vehicles. Then, among the ones capable of handle the request, they propose a quadratic worst-case complexity procedure in order to optimize the schedules taking into account excess time from the customer perspective and battery of the vehicle. Finally, the chosen insertion is the one that minimizes the operational cost given an objective function. The second stage consists on an improvement of the schedules once a myopic number of insertions has been done. For this, they present an improvement metaheuristic based on intra-route and inter-route customer exchange to improve vehicle plans. This optimization is made using Large Neighborhood Search in which the neighborhoods are generated based on classification and regression trees.

3 Method

In this section we will detail different representations of the problem, the network architecture used in the different cases and the framework to evaluate final results.

3.1 Planar Representation

In terms of representation, we are given a problem defined by a set of vehicles that are the agents to carry out the different tasks. The tasks are each of the clients we are requested to transport from the pickup to their dropoff locations. For each new request arrival we define the state of the system as the state of the vehicles at the given instant and the pickup and dropoff locations requested by the new client. For each vehicle we consider its current location, load, queue and plan -clients to pickup and drop-. The different locations are described in the usual coordinate system of the earth, described by latitude and longitude.

We will explore the planar representation of the problem in order to exploit its spatial properties. To do so, first we start by transforming the locations from the coordinate systems to a planar representation using the Mercator projection. This projection allows to preserve angles but may distort distances and areas as we move further from the equator. Nevertheless, we work with a small area in where the maximum distance between two points of interest is less than 3km.

Given the planar representation, we can encode the state of our system as a 3d tensor. The width and height of the tensor (dimensions 1 and 2) determine the size of the image we use to describe the space while dimension 0 represents the number of different channels we use in our representation. The final result is a very sparse tensor since the number of non-zero entries is of the order of the number of agents which is usually considerably smaller than the size of our tensor.

We will also consider a vectorial representation of the state in which each vehicle is represented by 6 features and then they are stacked in a final vector with size proportional to the number of agents.

3.1.1 Image size

The size of the image represents a compromise for the level of detail of our representation. As we reduce the size, the number of nodes being projected in the same pixel increases, losing bijectivity of our projection and making those nodes indistinguishable from the image perspective. The representation of each node is given by a uniform discretization of the space to obtain its corresponding entry in the matrix.

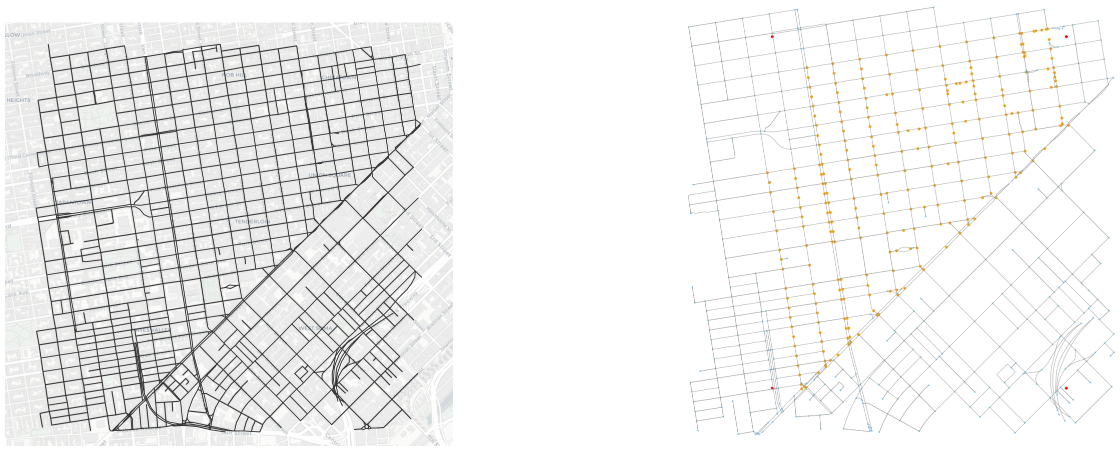

Figure 1: Area being represented. Left: Area with marked vehicle roads. Right: Area with marked vehicle roads in blue and nodes of interest in orange. Red nodes represent the area of study.

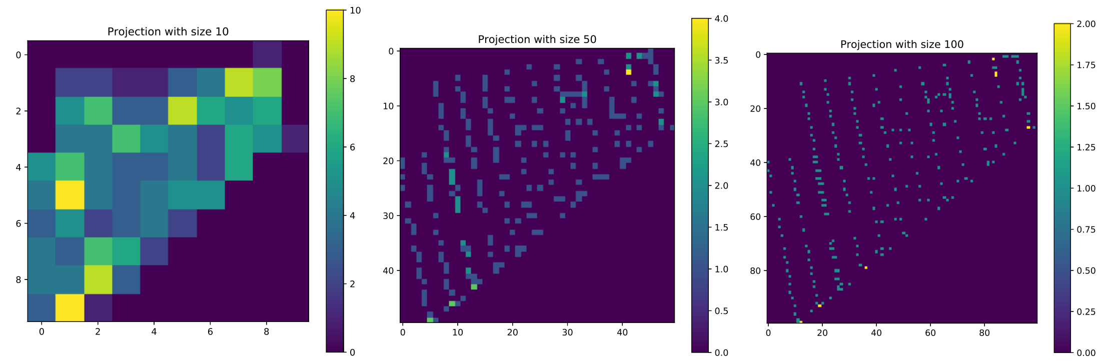

Figure 2: Image presentation. Color map shows the number of nodes embedded per pixel. Image size (width and height) from left to right: 10px; 50px; 100px. The real side length of a pixel in each case corresponds to 111m for the 10px size image, 20m for the 50px side image and 10m for the 100px side image.

Images in 2 show how the number of collisions evolves with the size of the image used for the representation. This has an effect in how we consider the client since different nodes being represented as the same entry in the matrix make them equal to the network while they can be located in distant places or even present differences in terms of accessibility.

For the smallest size proposed all the nodes are overlapping at least with one more and in the worst case up to 10 nodes will be represented in the same cell. When increasing the side size of the image up to 50 the number of colliding nodes reduces under the 10% and the worst case represents four nodes sharing the same matrix entry (top right part of the image). However, this area of the space is has a higher density of nodes and the case of the multiple collision represents nodes that are separated by a distance smaller than 15m and located within the same street. The structure of the considered traffic network is now distinguishable. Finally, with an image of side size 100px we can lower the amount of collisions to 2%.

We will consider representations of size around 50px since it presents a good compromise between image size and level of detail. It still allows to distinguish the spatial structure with which are working and provides a low number of overlaps.

3.1.2 Number of channels

In order to represent the different vehicles and customers we can use different channels. We will perform experiments with the following number of channels:

- 1 for customer; 1 for vehicles: The first channel represents the customer information and the second channel provides the vehicles information. The customer information is given by a 1 indicating the pickup location and a -1 at the dropoff location. The vehicles information encode their location at the moment the request is made represented as a 1 in the corresponding entry of the channel.

- 1 for customer; 1 per vehicle: The first channel represents the customer information and the following channels provides the vehicles information. The customer information is given by a 1 indicating the pickup location and a -1 at the dropoff location. The vehicles information encode their location at the moment the request is made represented as a 1 in the corresponding entry of the channel.

Other options such as 2 channels per vehicle to introduce the future location of its agent were also considered but discarded after very poor results.

The results concerning the number of channels used for the representation are discussed in Section 4.

3.2 Network architecture

We are interested in learning the vehicle assignment given a new client request using deep learning. For this we will use different architectures depending on the chosen representation of the problem. Nonetheless, all of them present a a first part composed of 3 convolutional layers followed by a second part composed by fully connected layers. The output is a vector of size number of vehicles which provides the selection of the next vehicle.

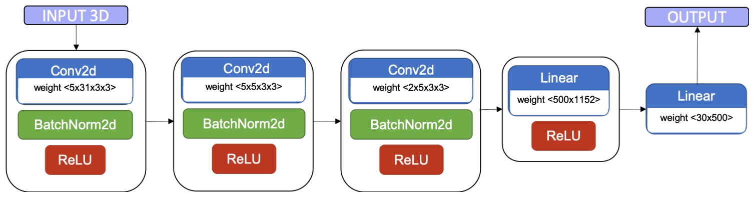

Figure 3: Architecture used in Model 1.

3.2.1 Additional features

In order to exploit further information of the state (load of the vehicle, future clients in queue, etc.) we can introduce an additional input to our network providing a vectorial representation of the current state of the system. In this case, we will evaluate the usage of this extra information as a late input to the network. This vector will be merged with the output of the network after the convolutional layers and fed into the fully connected part.

The state representation is given by a stacked vector of the state of each vehicle. Each vehicle is represented by 6 parameters: load, queue, current location (x,y), future location (x,y).

3.2.2 Auxiliary loss

Finally, the third variant we will explore includes an intermediate loss. In this case, after the convolutional layers we will have two different outputs: a first vector that aims to learn the position of the optimal vehicle and a second one corresponding to the one-hot coding of the chosen vehicle to perform the task. These two outputs are evaluated using different losses that are combined into a weighted final version to provide the final loss.

$$loss = w1 * Loss1 + (1 - w1) * Loss2$$

where

$$Loss1 = \frac{\sum_k (o1_k - position_k)^2}{k}$$

and

$$Loss2 = -o2[v] + log\left\{\sum_j exp(v[j])\right\}$$

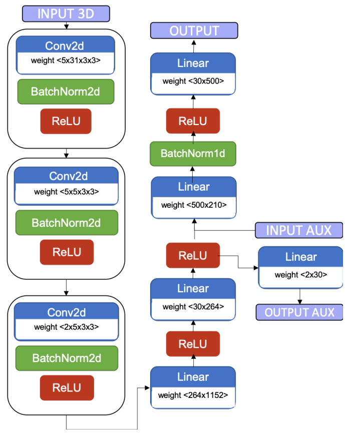

Figure 4: Architecture used in Model 4.

Among the different values, $o1$ represents the first output corresponding to the position of the desired vehicle. $o2$ corresponds to the final output which provides the vehicle assignment. v represents the index of the vehicle pointed as optimal.

3.3 Data generation and time evaluation

In order to obtain the representation of each state as well as evaluate the quality of the assignments in terms of time, we will use a simulation framework. The event simulator is implemented to work either with precomputed assignments (known as target policy) or with assignments computed by a pretrainned model given the current state.

For each new client request the simulator provides the current location of each vehicle, its current load and its current queue of clients to pick. We can also get the plan of the vehicle in terms of future location (next pickup or dropoff location to be attended). This information provides us with the state of the system. Then, once a vehicle is assigned to the client the simulator introduces this client into the vehicle plans and updates it setting as next location the closest pickup or dropoff point to be attended. Distances and travel time are computed using the library OSMNX [3] and considering a constant equal speed for all agents.

Once a new client request is received, the situation of each vehicle is updated by completing their respective plans up to the time of arrival of the new client. Every time a vehicle arrives to its destination, its new destination is set as the closest among its future pickup and dropoff locations to attend without violating the constraint of maximal load. If the vehicle has no client in queue or being transported, it remains in its current location.

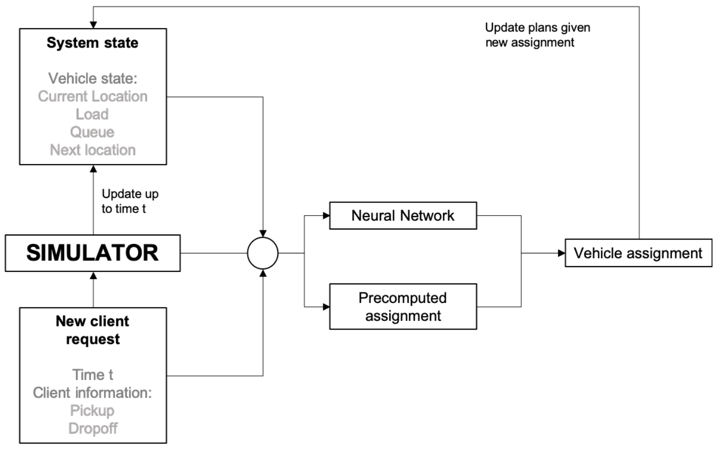

Figure 5: Schema simulator. The assignment is provided either by the neural network or by the precomputed values.

3.3.1 Assignment evaluation

Given the structure used for the implementation of the simulator, we can obtain different statistics for each simulated episode. In general we will concentrate on evaluating the ride times for each vehicle as well as the service time for each client. These two terms are the ones integrating the objective function that provides the optimal assignment we are trying to learn, so they will provide us an indicator of the quality of the assignments.

4 Experiments

The following experiments are performed using data provided by Claudia Bongiovanni on her work ”A Learning Large Neighborhood Search for The Dynamic Electric Autonomous Dial-A-Ride” [4]. The data consists on one week of Uber rides on the city of San Francisco generated from a simulator that provides client requests (time, pickup location and destination) as well as vehicle's information (i.e. location, load, battery level). The episodes are generated in batches of 100 clients and using 30 vehicles as size of the fleet.

It is important to remark that some simplifications from the original problem have been made when simulating the environment to evaluate the different policies. First, the battery of the vehicles is not taken into account although it was a constraint in the original setup. Besides, the time estimated for the different rides is obtained by considering the same constant speed for all agents and does not take into account traffic situations based on the considered situation. Finally, those customers that were rejected by the target policy due to infeasibility to meet their constraints will not be considered nor for the training of the models neither for their evaluation. Therefore we only treat with feasible requests and get rid of the possibility of discarding requests.

4.1 Supervised approach

As mentioned before, we will use a deep learning approach to try to learn an optimal policy in the DARP problem. To train the different models and evaluate their performance we will use two different losses depending on their output.

The first group of networks has a single output corresponding to a one-hot coded vector that provides the vehicle assignment. In this case we use Cross Entropy loss and report their accuracy in terms of matching the assignment with the optimal one provided by the data. The second group of networks considered have an intermediate output that aims to learn the position of the vehicle we are interested in. This output is evaluated using Mean Squared Error and the final loss of each training iteration is computed as the weighted sum of both losses.

We will also evaluate the influence of different variations, more specifically, the size of the input image, number of channels and information provided by channel, usage of intermediate loss and usage of intermediate input.

	Model	Parameters	Accuracy	Num. Parameters
1	**Single Input Single Output**	Im size = 30x30	0.138 ± 0.003	594911
2	**Single Input Double Output**	Im size = 30x30 Weight = 0.5	0.124 ± 0.003	597217
3	**Single Input Double Output**	Im size = 30x30 Weight = 0.99	0.137 ± 0.003	597217
4	**DI - DO 31x30x30**	Im size = 30x30 Weight = 0.99	0.243 ± 0.005	435740
5	**DI - DO 31x50x50**	Im size = 50x50 Weight = 0.99	0.248 ± 0.004	1153820
6	**DI - DO 2x30x30**	Im size = 30x30 Weight = 0.99 Single channel for vehicles	0.237 ± 0.001	434435
7	**DI - DO 2x50x50**	Im size = 50x50 Weight = 0.99 Single channel for vehicles	0.242 ± 0.004	1152515

Table 1: Results for different networks configurations. DI-DO stands for Double Input Double Output

Given the results shown in Table 1 we can see the accuracy for the different models is not very encouraging. Nevertheless different vehicles may have the same characteristics and therefore pointing out to a different value may not incur in a bad assignment since it would be equivalent to the one provided as optimal. Therefore we will evaluate the quality of the assignments in terms of travel time from clients and vehicles perspective to obtain a valuable comparison between the policy we aim to learn and the one provided by the different models we have trained.

What we can already point out is the difference in accuracy between those models incorporating extra information with the second input and those that do not make use of it. The former group achieve a higher accuracy and do not present significant differences when using 1 channel per vehicle or a single channel for all of them. Also, accuracy using different image sizes is not significantly different for this two cases (30px side image or 50px side image). Cases with more compact representation (10px) or sparser one (100px) were initially evaluated but discarded due to poor results in the case of 10px or improvements that were not relevant in front of the increase of size of the model.

4.2 Performance evaluation

In the following section we will refer to service time as the time between the customer request and the pickup. Travel time refers to the time between the pickup and dropoff for each client. Total time refers to the time between the customer request and its dropoff. Ride time refers to the total amount of time the vehicles have spend moving. Vehicles counts the vehicles that have performed at least one ride. All the time values are the sum along clients or vehicles.

Figure 6 shows the mean variation of the solution given by different models and the one obtained with the target policy. Models 1, 2 and 3 are not part of the comparison since their performance was much worse as it is shown in 4.2.1.

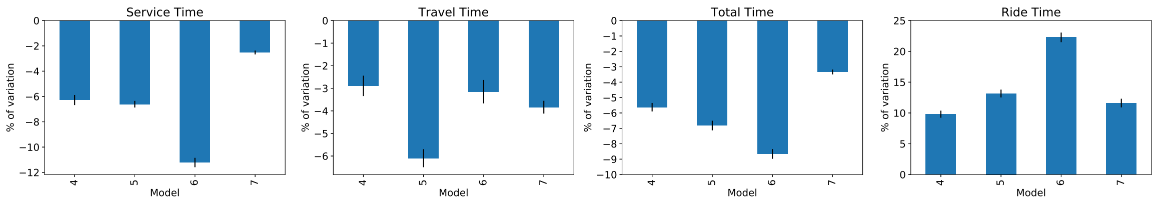

Figure 6: Mean variation in percentage of the achieved solution against the target one. Negative values indicate an increase of time while positive ones indicate a decrease and therefore improvement. Error bars estimated via bootstrapping.

The values are computed in the following way:

$$\%difference = 100 * \frac{\sum_{c=1}^{C} t_c^* - \sum_{c=1}^{C} t_c^{model}}{\sum_{c=1}^{C} t_c^*}$$

where C is the total number of customers, t_c^* is the service/travel/total time for customer c obtained with the target policy and t_c^{model} is the service/travel/total time for customer c obtained with the assignment given by the model.

For the ride time, the sum is performed over all the vehicles instead of the customers.

In general we can see that the solutions obtained incur in an increase of time from the customer's perspective while reduce the required time from the ride's perspective. Model 7 is the one closer to reduce the gap in terms of total time since if presents an average variation below 3%. On the other hand, Model 6 presents a larger gap in this field (although it remains below 10%) but presents a significant decrease in terms of ride time in front of the solution. Models 4 and 5 present similar performance in terms of service time while for travel time the performance of Model 4 is much better, what creates a significant difference in terms of the final total time.

Finally, we can point out that for all models there is a trade-off between total time and ride time and none of them presents bad results in both fields but generally a similar balance between the two objectives.

We will explore the performance in more detail to see how they can achieve solutions close to the target one in a given situation.

4.2.1 Performance in detail

	Service Time	**% difference**	**Travel Time**	**% difference**	**Total Time**	**% difference**	**Ride Time**	**% difference**	**Vehicles**
Opt Policy	**1157.37**	**-**	**1287.52**	**-**	**2444.89**	**-**	**20872.30**	**-**	**23**
Model 1	1647.35	-42.34	1412.39	-9.70	3059.74	-25.15	12023.39	73.60	14
Model 2	4734.23	-309.05	1360.72	-5.68	6094.95	-149.29	3983.95	140.46	4
Model 3	12028.73	-939.32	1227.45	4.67	13256.18	-442.20	999.41	165.29	1
Model 4	1223.34	-5.70	1274.90	0.98	2498.24	-2.18	21879.27	10.43	22
Model 5	1218.94	-5.32	1296.79	-0.72	2515.73	-2.90	19890.21	11.28	22
Model 6	1255.40	-8.47	1313.01	-1.98	2568.41	-5.05	17927.21	15.37	21
Model 7	1206.21	-4.22	1271.30	1.26	2477.51	-1.33	18918.66	9.36	21

Table 2: Evaluation of performance of different models in terms of time (minutes).

Given the results shown in 2, we can quickly point out models 2 and 3 as they present some problem. Actually, for Model 3 the network assigns the same vehicle to all customers while Model 2 produces some different outputs but still remains far from being good.

On the other hand, Model 1 (single input single output) presents an increase of the total time of 25% but incurs in a lower ride time by making use of less vehicles. Nevertheless, from a customer perspective, this policy does not provide interesting results since the delays are much higher and could actually lead to violations of the time window constraints imposed by the customers.

If we focus on models 4 to 7, the results seem more interesting.

Model 4 and 5 achieve very similar results. In both cases, there is an increase of the service while the travel time remains quite similar to the target one (Model 4 actually reduces it). As we will see in Figure 7, the increase of the service time comes from a group of outliers; a set of clients whose pickup is delayed much more than the rest. In terms of total time, the solution achieved differs less than 3% from the target one while it reduces the ride time by more than 10%. Given the multi-objective nature of our problem, we have a set of Pareto optimal solutions instead of a single one and the weighting in terms of each objective importance is left to the decision maker.

Model 6 presents larger variations in terms of total time but reduces the ride time by a greater amount. Nevertheless, from the customers perspective this model may not be that interesting since as shown in Figure 6 it presents the largest deviation in terms of total time.

Model 7 is the one presenting better results from the customer perspective, differing from the optimal less than 5% in terms of service time and less than 2% in total time. As the other models, the ride time is reduced compared to the target solution by making use of less vehicles than the target solution.

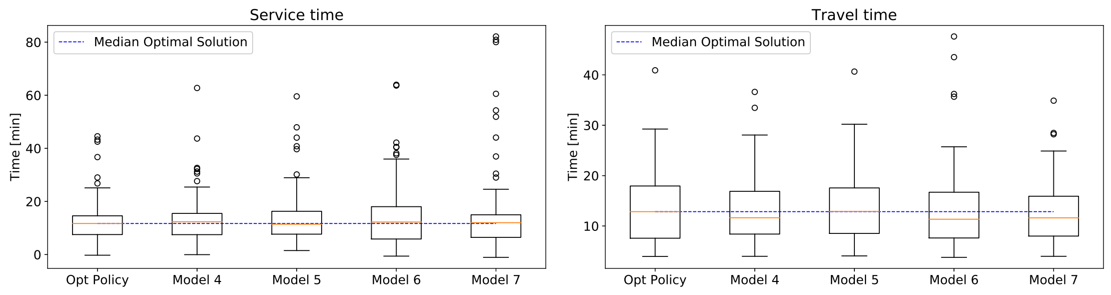

Figure 7: (a) Distribution of service time along customers. (b) Distribution of travel time along customers.

In Figure 7 we compare the performance from the clients point of view. We can see the distribution of the service time (time since client request to pickup) is wider in all cases compared to our target policy. Although achieving a lower service time for an important number of clients we obtain another group of requests that are postponed yielding in a very high service time. This may be a factor to highlight in favor of Models 4 and 5 rather than 7 since the value of the outliers for this later can actually make the requests unfeasible (service time much larger than the one a client may accept).

On the other hand, in terms of travel time, all models achieve a lower travel time for most of the customers and only some outliers obtain a travel time higher than the maximum obtained with the target policy. This effect compensates the service time, allowing to obtain a total time close to the desired one.

5 Conclusion

We have explored the learning capacities of supervised deep learning to learn a target policy exploiting different representations of the environment. We have seen that combining spatial representation and vectorial features can lead to results close to the ones obtained by the target policy in terms of total time to serve the customers and ride time of the vehicles. This representation is independent of the number of customers we face but depends linearly on the number of agents we consider. Nevertheless, solutions with a planar representation invariant to the number of vehicles can be worth exploring since they present good performance but may be more sensitive to the size of the representation we choose.

The results we obtain do not exchange customer between agents once a new customer is inserted in the plan. This is interesting since we can get rid of the second optimization step used in the target policy without incurring in a great deviation from the objective. Having said that, there are still limitations that need to be explored and that can lead to a more realistic solution since we are limiting the cases to those that we already know are feasible. Also, we are dropping some initial constraints such as the battery of the vehicles and considered a simplified environment by not taking into account traffic situations. Exploring these last fields represent key points to obtain a more accurate comparison of results.

References

[1] Abubakr O. Al-Abbasi, Arnob Ghosh, and Vaneet Aggarwal. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *CoRR*, abs/1903.03882, 2019.

[2] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *CoRR*, abs/1811.06128, 2018.

[3] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *CoRR*, abs/1611.01890, 2016.

[4] Jean-Fran cois Cordeau Claudia Bongiovanni, Mor Kaspi. A learning large neighborhood search for the dynamic electric autonomous dial-a-ride problem. 2019.

[5] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, September 2007.

[6] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. pages 80–91, 2008.

[7] Quan Lu and Maged M. Dessouky. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38:503–514, 2004.

[8] Teo Stocco. Learning online combinatorial stochastic policies with deep reinforcement. 2019.

[9] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc., 2015.