Taxi re-positioning considering driver compliance

Cebrina Lindstroem[1][0000−0002−2306−0140], Stefan Ropke[1][0000−0002−6799−9934], and Reza Pourmoayed[2][0000−0002−4636−3400]

[1] DTU Management, Technical University of Denmark, Akademivej Building 358, 2800 Kgs. Lyngby, Denmark
[2] BNR A/S, Rolighedsvej 32, 8240 Risskov, Denmark

Abstract. Managing taxi fleets in large cities is challenging, especially when drivers operate independently. This study improves taxi repositioning by predicting demand and providing smart recommendations. Using real-world data from a Scandinavian taxi service, we employ neural networks with LSTM layers to forecast demand and test different strategies, like a simple greedy algorithm and a more structured matching-based approach, to guide taxis to high-demand areas. Since drivers ultimately decide whether to follow these suggestions, we also model their behavior using a probabilistic acceptance strategy. Through a simulation of a real day, we analyze how different approaches impact passenger wait times and overall efficiency. The results show that proactive re-positioning significantly reduces wait times but can increase total driving distance. The greedy algorithm tends to perform better in quickly getting taxis to passengers, while the matching model is more efficient in minimizing unnecessary travel. However, increased mobility comes at a cost, as rerouting leads to longer driving distances. Additionally, driver behavior plays a crucial role, with lower acceptance rates reducing the effectiveness of predictive recommendations. Overestimating demand in such cases helps mitigate inefficiencies.

Keywords: Taxi re-positioning · LP based heuristics · Multivariate time series forecasting.

1 Introduction

Efficiently allocating taxis in large Scandinavian cities is a difficult challenge. The aim is to minimize passenger wait times while optimizing the transport system, driver earnings, the number of trips serviced by the same cars, and inefficient wait times. Much of the existing literature assumes this problem to have centralized authority that can dispatch the taxis and direct taxis to specific zones or passengers. Yet, in multiple real-life settings, taxi dispatching is decentralized with each taxi working as an individual agent. This shifts the problem from optimizing a centralized system to guiding and incentivizing a collection of autonomous agents.

In both the centralized and de-centralized versions of this vehicle routing problem, the dispatching occurs based on some assumptions or predictions for future demand. To predict future demand, recurrent neural networks with LSTM

cells are a strong tool, as used in this study. However, with these predictions, there is still a need to understand how they can be used to guide taxis working as individual agents efficiently.

We tackle this challenge by integrating demand prediction with a simulation-based framework to explore different aspects of decentralised taxi routing. Our simulation models a scenario where individual drivers receive zone recommendations based on predicted demand. The drivers have full control over their routing decisions. To generate recommendations, we test different approaches, such as greedy algorithms and solving an assignment problem.

In this paper, we examine this challenge using a real-life case study from a Scandinavian taxi service. Currently, the system does not provide any guidance to the taxis, so the drivers make completely independent decisions regarding their movement. However, there is interest in implementing a guidance system to help drivers make more efficient decisions, which motivates this investigation of integrating demand prediction with decentralized routing decisions.

Furthermore, we model driver behavior through a probabilistic acceptance strategy to see how these behaviors affect different outcomes for both drivers and passengers.

Our key contributions are 1) providing a simulation-based framework for evaluating decentralized taxi reouting in systems where drivers have full control over their rerouting decisions and 2) examining the dual aspects of recommendation generation and driver response, providing insights into how system performance is influenced by drivers who choose not to follow the guidance.

2 Related Work

The problem of repositioning taxis and similar services is well-studied. Research in this area follows two main approaches: classical operations research (OR) techniques and deep reinforcement learning.

In the OR stream, Hao et al. [8] address taxi relocation in Singapore after the morning rush hour, when taxis tend to cluster in low-demand downtown areas. Their model, solved once daily, uses demand prediction based on weather data, applying a regression tree with precipitation-based splits. A distributionally robust model plans repositioning, outperforming simpler approaches.

Tavor and Raviv [11] assume fully automated taxis and solve repositioning continuously. They propose three LP-based models: one maintaining a fixed number of cars per zone and two using demand forecasts. These models are solved every 15 minutes and tested on New York City taxi data, showing that demand-aware approaches balance empty mileage and customer wait times effectively.

Guo et al. [7] use the smart-predict-then-optimize approach proposed by Elmachtoub and Grigas [4] for taxi re-positioning. They consider a taxi-like service provided by Didi Chuxing in Chengdu, China, and provide several mathematical models for relocating vehicles. They conclude that their smart-predict-then-optimize approach outperforms simpler benchmarks as well as other mathematical models defined in the paper.

The problem of re-positioning taxis has been addressed through reinforcement learning in, for example, [6], [9], and [10]. The approach by Gammelli

et al. [6] is of special interest as it uses graph reinforcement learning combined with linear programming to reposition taxis. The graph neural network suggests how taxis should be distributed among zones, and the linear program (LP) finds a cost-efficient way to obtain this distribution from the current one. This simplifies the reinforcement learning algorithm as it does not have to consider the feasibility of its proposed distribution and does not have to specify the precise movements of taxis, the LP takes care of these parts.

The issue of drivers not adhering to repositioning suggestions has not received much attention in the literature. Wang and Wang [12] studies relocating vehicles using data from Didi Chuxing in China. It is acknowledged that vehicle relocation tasks will only be a suggestion to the drivers, and drivers are not required to follow the suggestion, however, this fact is, to the best of our understanding, not embedded in the solution method or in the simulation results. The solution method is based on mathematical models that aim at matching up as many vehicles with transport requests. Very recently, Brar et al. [2] and Chen et al. [3] studied the question further. Brar et al. [2] model driver behavior based on repositioning preferences and confidence level in the recommendations provided and incorporate the driver model in a repositioning algorithm. Chen et al. [3] approach the issue of driver behavior from two angles, one is to analyze past data to learn how a driver *cruise* while being idle, and the other is to use a survey to understand what makes it likely for a driver to accept a repositioning request. The model for driver behavior is then integrated into a repositioning algorithm based on reinforcement learning. Our work differs from both Brar et al. and [3] by focusing on a fully decentralised fleet, thus evaluating the effects of driver uncertainty on system-wide outcomes rather than mitigating the effects of incomplete driver compliance.

3 Problem description

The inspiration for this problem stems from a collaboration with BNR A/S and the challenges faced by Scandinavian taxi centrals. This section provides context on the issue.

Taxis are vital to urban transport, and in many Scandinavian cities, they operate independently while the central dispatches rides. Although trips are assigned via a queue system, drivers retain full autonomy over repositioning between trips and accepting assigned rides. This study assumes full trip acceptance, focusing on the effects of repositioning while maintaining driver control. Taxi demand fluctuates based on temporal, spatial, and contextual factors like weather and events. Currently, repositioning is left to the driver's intuition, making the system reactive. The goal is to develop a proactive guidance system. Trips are allocated based on queue order within zones. If a zone queue is empty, the trip is assigned from the nearest available queue. Each trip has a fixed pickup and drop-off, without ridesharing. While some trips originate from street hails, the simulation assumes all trips are centrally booked.

In the simulation, taxis act as independent agents, making movement decisions freely. They enter new queues when switching zones and receive rerouting recommendations but decide whether to follow them. Priority is given to taxis

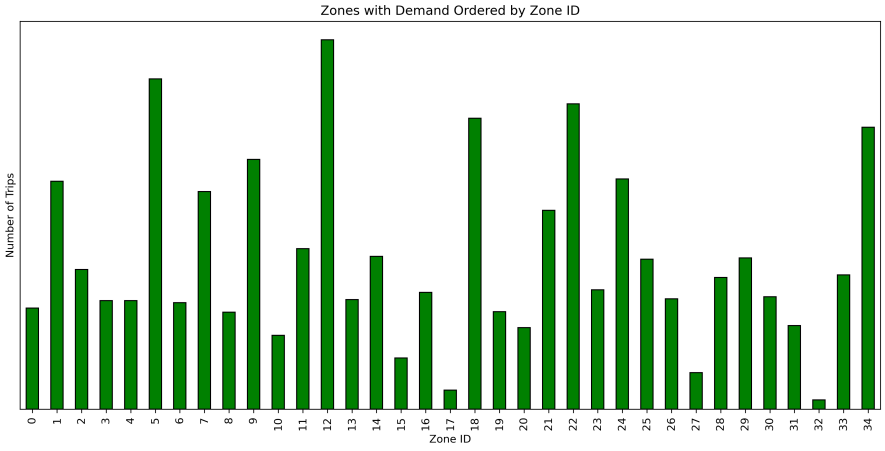

Fig. 1. A bar chart respresenting the number of trips per zone, ordered by zone ID.

already waiting in a zone over those en route. A probabilistic acceptance strategy is applied, with drivers accepting rerouting suggestions at rates of 100%, 70%, 50%, or 30%. This approach was chosen due to no available data regarding historical acceptance behaviour as no such recommendation system is currently implemented. Additionally, the study aims to focus on understanding the impact of decentralized drivers within a system, making this approach well-suited to the research objective. The goal is to minimize passenger wait times while balancing the additional kilometers driven due to rerouting.

4 Datasets and demand prediction

A central part of the simulation is the ability to predict demand arising in different zones and, based on this, reroute the taxis to zones where there is a supply deficit. The prediction is trained and used on a dataset collected over multiple years.

4.1 Data

The data for this study comes from a real Scandinavian taxi central and includes all trips from August 2016 to December 2024. This dataset serves as the foundation for training the demand prediction model, containing details on trip type, price, pickup/delivery times, and locations.

Notably, the dataset only includes completed trips, omitting declined or unrecorded ones, which limits insight into total demand.

For prediction, we used a zone-based approach, clustering historical trips with k-means clustering based on the locations of the trips. As city-supported zones evolved over time, this method ensured a stable, flexible framework. We selected 35 zones based on the elbow method [5]. We found 35 zones provided the best granularity, balancing spatial detail with model simplicity.

Figure 1 provides insight into the dataset used in this paper and the clustering. It shows a bar chart of the zones (ordered numerically) based on the total demand within each zone. There is a clear variance in the number of trips across

the zones, reflecting the differences in demand between high- and low-density areas.

4.2 Demand prediction using Neural Networks with LSTM-cells

To optimize taxi routing toward high-demand areas, our solution relies on a demand prediction model. We use a neural network with Long Short-Term Memory (LSTM) cells, which effectively handle time-series data by capturing long-term patterns and mitigating vanishing gradients. This capability is crucial for demand forecasting, where historical trends strongly shape future outcomes.

The model predicts demand across all zones simultaneously as a multivariate time-series problem, rather than 35 separate univariate time-series as the zone demands are highly correlated. This also ensures more consistent and coherent predictions across the city. Input data consists of summed historical demand, with predictions based on lag variables representing past demand. We used 10-minute time steps and found the best balance between accuracy and efficiency with 144 lag variables (covering two days). We experimented with incorporating external variables such as weather conditions, COVID-19 case counts, pre-bookings, and holidays. However, these features offered minimal gains in prediction accuracy while substantially increasing computational complexity and runtime. The only added inputs were seasonality/cyclic inputs, specifically the month, weekday as well as the period within the hour. These inputs were cyclically encoded to preserve the cyclic nature of the data, while also reducing the number of input features compared to alternatives like one-hot encoding [1].

The model provides both point predictions and confidence intervals by estimating quantiles, where the 90th percentile serves as the upper bound and the 10th percentile as the lower bound. The network is trained through quantile regression, which minimizes a quantile loss function tailored to each specified quantile. We predicted three time periods into the future, thus as any point we predict the demand for the next 30 minutes divided into 10-minute intervals.

Our architecture includes two shared LSTM layers (256 and 128 units) and an individual LSTM layers (128 units each) for median, upper, and lower quantile predictions. A dropout rate of 0.3 is applied between layers. Figure 2 shows an overview of the architecture.

The demand prediction is programmed in Python using the Python package Keras. It is trained using a training-validation-testing split of 70%-10%-20%, and the day of the simulation is part of the testing data (and occurs at the end of the time series). Thus the model is not trained on the data for the simulation, but it has been evaluated based on this data before use. The model achieved an R-squared value of approximately 0.74 on validation and testing data.

We compared the prediction performance of our multivariate model to traditional models such as Auto-ETS, which were run separately on each zone. Auto-ETS achieved an average R-squared of 0.48 across all zones on the test set, with values ranging from 0.17 to 0.66. The multivariate LSTM model achieved a significantly higher average R-squared of 0.74 across all zones on the test set, ranging from 0.30 to 0.94. It outperformed the Auto-ETS models in all but three zones.

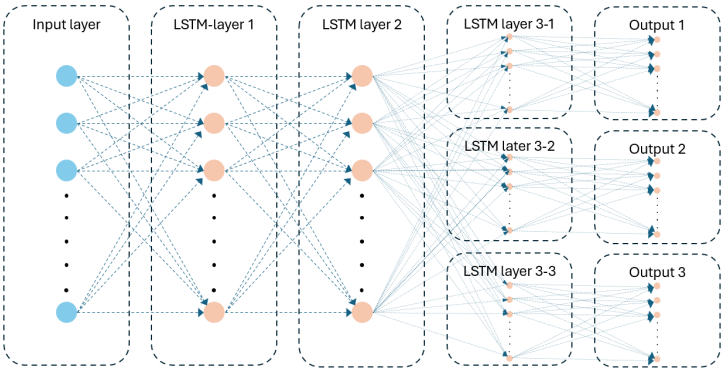

Fig. 2. The architecture of the neural network used for predicting demand.

In Figure 3, the predicted and actual values for each zone are displayed, along with the confidence intervals for the predictions, for a single day between the times 8 and 18. The model demonstrates strong performance in many zones, accurately predicting demand patterns. However, in certain cases, such as Zone 10, the model struggles due to significant fluctuations in the actual values. This uncertainty is reflected in larger confidence intervals, whereas zones like Zone 7 has smaller confidence intervals and still manage to capture key demand peaks effectively. A similar plot for a low-demand day is shown in the appendix to illustrate the models ability to handle diverse demand situations.

5 Simulation and suggestion methods

Based on the prediction model obtained as described in Section 4.2, we modeled a simulation framework to "replay" an 8-hour interval of a real day's demands.

5.1 Simulation Framework

To evaluate the effectiveness of different rerouting techniques, we implemented a discrete event-based simulation. This approach lets up replay an actual day. The discrete event-based framework was chosen for its ability to model systems where changes occur at irregular intervals as is the case of taxi demand and routing. It is implemented as a priority queue, so new events can be added (e.g. a end-of-trip-event added whenever a new begin-trip-event has been handled. Key events for the simulation are:

1. New Trip: Trip requests generated based on a single day. These are part of the event queue from the beginning. When a new trip arises, it is assigned to a taxi, which in turn generates a new trip end event that is added to the event queue.
2. Trip ends: When a trip ends, the status of the assigned taxi is updated. The taxi becomes available and is added to the dispatching queue of the end zone of the trip, where it currently is. A routing suggestion event is also added to the event queue, enabling the system to provide a relocation recommendation.

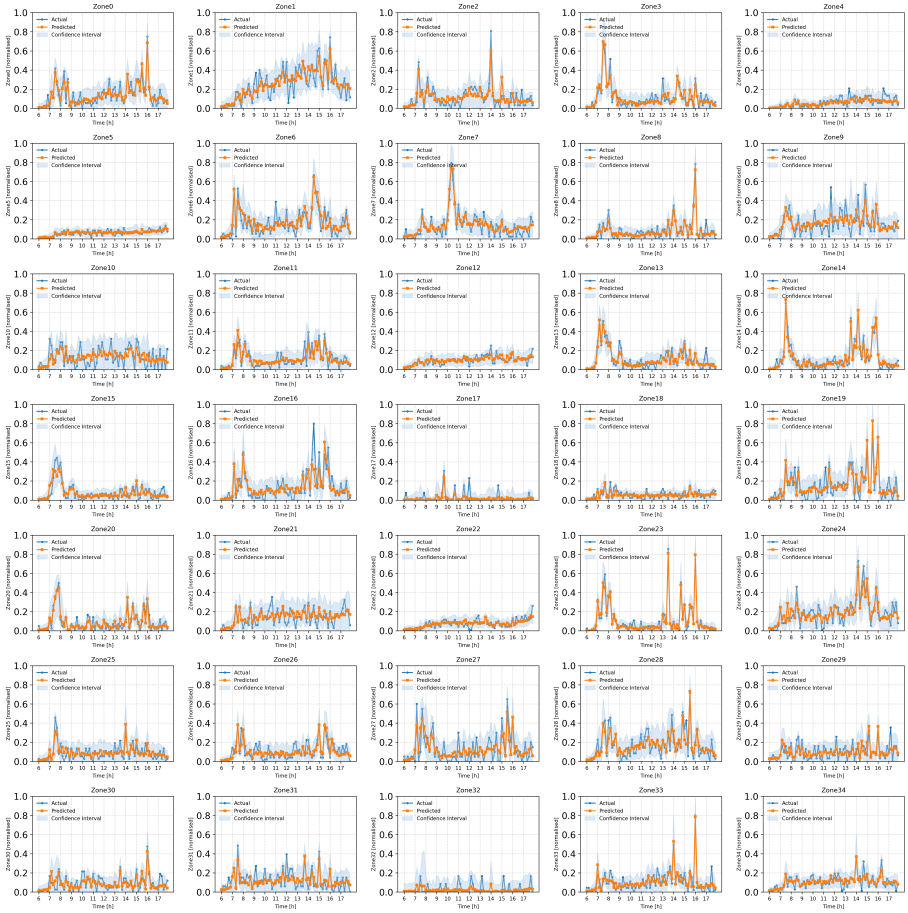

Fig. 3. Each plot shows the actual and predicted values for a single zone, with actual values as blue dots and predictions as orange x. Confidence intervals are shown with light blue shading. The x-axis shows time from 6 to 18 hour, and the y-axis represents normalized demand.

3. Prediction: Demand forecasts are periodically updated and inserted into the event queue at fixed intervals. In this setup, predictions are generated every 10 minutes, matching the model's forecast interval. These events are used during routing suggestion events but do not themselves generate additional events.

4. Routing Suggestions: Relocation recommendations are issued to drivers either when a trip ends or when a taxi has remained idle for an extended period. If accepted, the taxi begins moving to the suggested zone, an end-reroute event is added to the event queue, and the taxi is assigned a queue number in the new zone (though dispatching prioritizes taxis already present in the zone, if such taxis exist). If the suggestion is rejected, a new routing

suggestion event is scheduled after 10 minutes in case relocation becomes preferable.

5. End Reroute: The rerouting process concludes and the taxi's status is updated. A new routing suggestion event is added to the event queue after 10 minutes, in case the taxi remains unassigned and relocation becomes necessary.

A pseudocode of the simulation framework can be found in Algorithm 1.

Algorithm 1 Simulation framework

```
input: a set of taxis V, a set of events E, a prediction model m and a suggestion algorithm alg_sug
while E is not empty do
    get next event e from E
    if e is of type new-trip then
        Assign the demand of e to the first available taxi v from V in queue in the zone. If no queue in the zone assign to the nearest available taxi v from V. If no taxi available, wait.
        If a taxi is assigned a trip add an end-of-trip event to E
    else if e is of type end-of-trip then
        Complete trip and let taxi v be available for new trips
        Add routing suggestion event to E
    else if e is of type prediction then
        Predict from current state the demand into the next number of time periods based on prediction model m
    else if e is of type routing-suggestion then
        Suggest to vehicle from event e the next zone to travel to based on alg_sug. If the vehicle accept begin reroute, add the taxi to the dispatching queue of the suggested zone, and add end reroute event to E. Otherwise add routing suggestion event to E for the vehicle in 10 minutes.
    else if e is of type end reroute then
        The state of taxi is updated
        Add routing suggestion event to E
    end if
end while
```

5.2 Suggestion methods for rerouting

A central part of the simulation comes from the routing suggestions for the drivers. In this part of the simulation the drivers are presented to a new zone they should relocate to, and then they decide whether or not to follow the systems recommendation. We worked with three different suggestion methods: (1) A static strategy, where we did not move taxis at all, thus providing a baseline for our data, (2) A simple greedy algorithm calculating the deficit in each zone, i.e. the difference between predicted demand and available taxis, and reroutes drivers to the zone with the largest deficit, and (3) A matching problem solution, where we solve a matching problem every few minutes and reroute based on this.

We also tried different suggestion intervals, as we found it beneficial to let a taxi stay in one place right after it finished a trip. Thus, if the suggestion interval was 0 minutes, we would suggest a new zone for a taxi right after they finished a

trip, while if the suggestion interval was 10 minutes, we would wait 10 minutes before suggesting a new trip and thus let the taxi get into the queue of their finishing zone. Waiting briefly before suggesting a relocation proved helpful in some cases, as the current location might turn out to be a hotspot, allowing the taxi to get a trip without needing to relocate.

Each suggestion algorithm had some common implementations. First, if we did not have a zone recommendation for a taxi, we let the taxi stay and gave them a new recommendation after the next prediction. Secondly, if a taxi chooses to decline our suggestion, we give them a new suggestion after 10 minutes, unless they get assigned a trip before this time. Finally, in both the greedy algorithm and the matching problem, we subtracted the actual demand since the last prediction from the predicted demand. Thus, if $pred_j$ defines the predicted demand in zone j, and act_j is the actual realized demand that has occurred in zone j since the last prediction, the predicted demand used in the suggestion would be $pred_j - act_j$.

Greedy algorithm The greedy algorithm calculates the deficit in all zones based on the latest prediction and reroutes taxis according to the largest deficit. We explored both global and local approaches: the global version calculated the deficit across all zones and made decisions based on this, while the local version focused only on nearby zones for its calculations. We found that the global version often resulted in taxis being dispatched to distant zones, leading to inefficiencies. As a result, we opted for the local version of the greedy algorithm.

In this local version, the algorithm considers only the nearby zones (defined as those that can be reached within 8 minutes) and calculates the deficit for each. It then suggests the zone with the largest deficit. To prioritize more immediate demand, the algorithm places greater weight on deficits occurring within the next 10 minutes. If the predicted demand for the next 10 minutes is satisfied, the algorithm extends the horizon to 20 minutes, and later to 30 minutes, in line with our three-period prediction model.

If a deficit was predicted in a zone, we refrained from suggesting a new zone for the taxis, as it would be counterproductive to relocate vehicles to different areas when other taxis needed to return to the original location to meet demand.

Matching problem The matching problem algorithm solves a matching problem at specific time steps, where each predicted demand for each zone must be met by the same number of taxis. Specifically, we let V be the set of taxis and J be the set of zones. We then define $x_{ij} = 1$ if vehicle $i \in V$ is matched to zone $j \in J$, and 0 otherwise. Define d_{ij} as the distance from vehicle $i \in V$ to zone $j \in J$. Furthermore, we let z_j be a continuous positive slack variable, defined as a demand not met in a zone. To heavily prioritize satisfying all demand, we impose a penalty on z_j with a value of $\alpha = 1,000,000$. We let $pred_j$ describe the predicted demand in zone $j \in J$. Unlike the greedy algorithm, we do not restrict how far taxis can travel, as our goal is to find the best global solution. Travel distance is already penalized in the objective function. Thus the problem

we solve at a specific time step is:

$$\min \sum_{i \in V} \sum_{j \in J} d_{ij} x_{ij} + \sum_{j \in J} \alpha z_j$$

s. t.

$$\sum_{j \in J} x_{ij} \leq 1 \qquad \forall i \in V \tag{1}$$

$$\sum_{i \in V} x_{ij} + z_j \geq pred_j \qquad \forall j \in J \tag{2}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, j \in J \tag{3}$$

$$z_j \geq 0 \qquad \forall j \in J \tag{4}$$

We found that the optimal time frame for solving the matching problem was every 5 minutes, as this frequency strikes a balance that prevents recalculations that lead to conflicting solutions or contradictory recommendations. We used Cplex to solve the linear problem.

6 Computational results

For the numerical study, we ran the simulation described in Section 5.1 to replay an 8-hour interval of a real day on this Scandinavian taxi central. Specifically, we took all actual trips run between 8:00 and 16:00 on this day and let them appear dynamically throughout the simulation. This 8-hour interval had 6112 actual trips, and in the simulation, we serviced these trips with a fleet of 300 vehicles, where we assume all of them are in operation all of the time, as opposed to real life where they take natural breaks throughout the day. We used OpenStreetMap to calculate distances between any two points. We treated all taxis and trips within a zone as clustered together, considering only distances between zones.

6.1 Varying the suggestion methods and changing the acceptance rates

We ran the simulation with the different suggestion methods as well as different acceptance rates, i.e. the chance a taxi accepts our recommendation. Specifically, we ran the model with acceptance rates of 100%, 70%, 50%, and 30%. The results can be found in Table 1. The columns show the used suggestion algorithm, taxi acceptance percentage, average duration to the pickup location in seconds, percentage of trips serviced in their starting zone, percentage of taxis receiving trips in their relocation zone, and finally, the total driven kilometers.

As can be seen rerouting the taxis to zones before their suggestion greatly decreases the average duration from a taxi to their assigned pickup trip. Please note that the average duration is quite low, as we only consider distances and durations between zones, and do not consider wait times for acceptance. Compared to the static strategy, where we do not move any of the taxis preemptively, the greedy algorithm (if followed completely by all taxis) can decrease the average duration by 71%, whereas the matching solution can decrease the average

duration by 55% when followed completely. It thereby proves a significant improvement in passenger wait times and proximity to dynamically occurring trips, by using a prediction to reroute the taxis. However, in this case, it seems the greedy algorithm performs better than solving the LP of a matching problem, which can be explained by the imbalance of solving an exact problem based on predicted (i.e. not exact) numbers. The cost of using a proactive rerouting strategy can, however, be seen in the total driven kms. When using the greedy algorithm completely we see an increase in the total driven km compared to the static strategy of 16%, whereas using the matching problem only increases the driven kms by 4%. Thus, there is a definitive trade-off between an increased proximity to pickup locations and the total kms driven.

When comparing different acceptance percentages, we see an increase in the average duration to pickup when the acceptance percentage falls for both suggestion methods. However, there is still a great decrease of the average duration to pickup, as can be seen, when we only accept 30% of the suggestion methods, the average duration for the greedy algorithm is still decreased 60% compared to the static stragey, while the matching problem decreases average duration by 50%. These numbers will slowly increase until they reach the static strategy when no rerouting suggestions are accepted. The reason the number only slowly approaches the average duration in the static strategy is explained by the fact, that when the acceptance percentage falls, the simulation ends up suggesting new zones more often to the taxis, as the ones that decline, will get a new suggestion 10 minutes later, when a new prediction has been made. For example, we make 2485 rerouting suggestions in the greedy algorithm when all suggestions are accepted, whereas we make 3666 suggestions when only 30% of suggestions are accepted.

6.2 Results of the algorithm when we have complete knowledge of the future

We ran the simulation with the same algorithm strategies, except we assumed complete knowledge of the future, i.e. we knew exactly how many trips would

Algorithm	Acceptance percentage (%)	Average duration to pickup (s)	Trips serviced within a zone (%)	Successful reroutes (%)	Total kms driven (km)
Static strategy	-	177	69	-	56,783
Greedy	100	51	87	76	65,661
Greedy	70	55	85	75	63,249
Greedy	50	60	82	74	61,931
Greedy	30	71	79	74	59,857
Matching	100	80	80	68	59,036
Matching	70	82	79	67	58,461
Matching	50	83	79	69	57,983
Matching	30	89	76	69	57,389

Table 1. Results from varying degrees of acceptance rates across different suggestion methods. Successful reroute is defined as the percentage of taxis that receive a trip within the zone they were rerouted to.

occur in a zone in the next three time periods when rerouting. The results can be found in Table 2.

As can be seen, having complete knowledge of the future further decreases the average duration to the pickup locations. We see the greedy algorithm, when there is a perfect acceptance rate of 100%, decreases the average duration to pickup by 12% compared to using the predicted values, whereas in the matching problem, we got a 46% decrease, supporting the explanation that this method is more sensitive to inaccurate predictions. We actually now see that the matching problem outperforms the greedy algorithm when we have complete acceptance. However, this is quickly negated when we change the acceptance rates, as already when we have an acceptance rate of 70%, the greedy algorithm again outperforms the matching problem. This indicates that the matching problem is also more sensitive to the acceptance rate since the solution relies on all taxis following the suggestions. In contrast, the greedy algorithm only considers the current state and recommends the best option for an individual taxi. However, even with e.g. 50% acceptance rate, we find an improvement of the matching problem over the static strategy of 53% when knowing the future, but only with a cost increase of kms of 3%. Therefore, the algorithm may be advantageous if more accurate predictions can be obtained, as opposed to the greedy approach, which results in a higher cost increase.

6.3 Performance in different scenarios

The prediction model, as described in Section 4.2, outputs three different values, one for the actual prediction and an upper and lower confidence interval. In Table 3 the results from running the Greedy algorithm on both the high and low prediction scenario is seen, where the high and low prediction compares to higher and lower predicted values in each zone corresponding to upper and lower quantiles, respectively.

As seen in the results, both the high and low scenarios perform worse than the median prediction when the acceptance rate is 100%. There is also a significant difference in total kilometers driven: the low scenario results in minimal additional driving, as it takes a conservative approach and only relocates taxis when there is a high certainty of a demand deficit. In contrast, the high scenario leads to a substantial increase in kilometers driven, as it tends to overestimate

Algorithm	Acceptance percentage (%)	Average duration to pickup (s)	Trips serviced within a zone (%)	Successful reroutes (%)	Total kms driven (km)
Greedy	100	45	91	75	70,370
Greedy	70	52	90	77	67,212
Greedy	50	59	88	78	65,327
Greedy	30	70	85	78	62,787
Matching	100	43	86	84	59,324
Matching	70	54	82	83	59,069
Matching	50	62	80	79	58,504
Matching	30	83	75	80	57,443

Table 2. Results from the simulation with complete knowledge of the future.

Algorithm	Acceptance percentage (%)	Scenario	Average duration to pickup (s)	Trips serviced within a zone (%)	Successful reroutes (%)	Total kms driven (km)
Greedy	100	low	85	77	45	58,771
Greedy	70	low	89	76	43	58,243
Greedy	50	low	93	74	42	57,657
Greedy	30	low	102	74	44	57,665
Greedy	100	high	57	86	77	77,648
Greedy	70	high	57	86	79	73,704
Greedy	50	high	58	85	80	69,218
Greedy	30	high	67	83	79	65,780

Table 3. Results from the simulation with different prediction scenarios.

the needed demand, causing more frequent vehicle movement. It is worth noting that all of the results in these scenarios are better than when not moving the taxis at all.

Interestingly, when the acceptance rate drops to 50% or 30%, the high scenario outperforms the median prediction. This is likely because, with fewer taxis accepting re-routing suggestions, more taxis need to be directed to high-demand zones. By overestimating demand, the high scenario compensates for the lower acceptance rate, ensuring better overall coverage.

7 Conclusion

This study investigates taxi re-repositioning by integrating demand prediction with uncertain driver compliance. Our simulations demonstrate that proactive rerouting significantly reduces passenger wait times, although it may increase total kilometers driven. The greedy algorithm proves particularly effective at minimizing wait times, while the matching approach balances efficiency with lower additional travel costs.

One key finding is that driver acceptance rates play a crucial role in overall system performance. When acceptance rates drop, overestimating demand (high scenario) can help maintain efficiency by ensuring enough taxis relocate to high-demand areas. This highlights the challenge of balancing system-level interventions with driver autonomy in decentralized environments.

Future research could explore alternative suggestion and prediction methods, such as Markov decision processes. Additionally, more advanced driver behavior models may provide further insights into drivers' choices. Another interesting direction is to investigate how drivers might be incentivized to accept the suggestions, as this study shows that system-wide benefits arise when all drivers follow the recommendations. This could also involve examining the outcomes for drivers who follow the suggestions compared to those who do not. Finally, examining the impact of exact distances within zones would be valuable, as it improves real-world applicability but also requires considering where taxis should stay within the zone.

Acknowledgments. This study was funded by BNR A/S. We thank Filipe Rodrigues for advice and discussions on demand prediction using recurrent neural networks.

Disclosure of Interests. C. Lindstroem is an industrial PhD student at BNR A/S, and R. Pourmoayed is employed by BNR A/S. S. Ropke has no affiliations with the company. The authors declare that they have no competing interests that are relevant to the content of this article.

A Appendix

Figure 4 shows the predicted zone plots for a low-demand day (weekend). Actual values are blue dots, predicted values are orange x. Confidence intervals marked with light blue shading. The x-axis shows time from 6 to 18 hour, and the y-axis represents normalized demand.

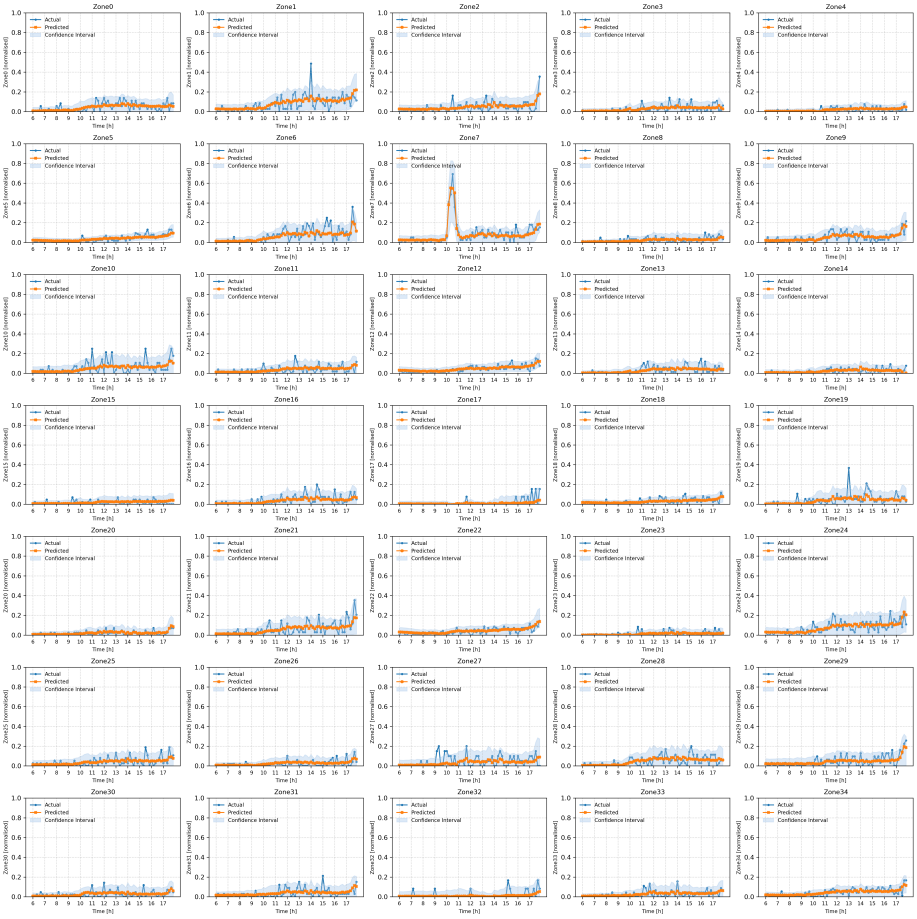

Fig. 4. Predicted and actual values with confidence intervals for all 35 zones on a low-demand day.

Bibliography

[1] Anthony Adams and Peter Vamplew. Encoding and decoding cyclic data. *The South Pacific Journal of Natural Science*, 1998.

[2] Avalpreet Singh Brar, Rong Su, and Gioele Zardini. Vehicle rebalancing under adherence uncertainty. *arXiv preprint arXiv:2412.16632*, 2024.

[3] Haoyang Chen, Peiyan Sun, Qiyuan Song, Wanyuan Wang, Weiwei Wu, Wencan Zhang, Guanyu Gao, and Yan Lyu. i-rebalance: Personalized vehicle repositioning for supply demand balance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 46–54, 2024.

[4] Adam N Elmachtoub and Paul Grigas. Smart predict then optimize. *Management Science*, 68(1):9–26, 2022.

[5] Marisa Fitri, Wardhani Arie Restu, Purnomowati Wiwin, Vitianingsih Anik Vega, Maukar Anastasia Lidya, and Puspitarini Erri Wahyu. Potential customer analysis using k-means with elbow methos. *Jiko (jurnal Informatika Dan Komputer)*, 7:307, 2023.

[6] Daniele Gammelli, James Harrison, Kaidi Yang, Marco Pavone, Filipe Rodrigues, and Francisco C Pereira. Graph reinforcement learning for network control via bi-level optimization. *arXiv preprint arXiv:2305.09129*, 2023.

[7] Zhen Guo, Bin Yu, Wenxuan Shan, and Baozhen Yao. Data-driven robust optimization for contextual vehicle rebalancing in on-demand ride services under demand uncertainty. *Transportation Research Part C: Emerging Technologies*, 154:104244, 2023.

[8] Zhaowei Hao, Long He, Zhenyu Hu, and Jun Jiang. Robust vehicle preallocation with uncertain covariates. *Production and Operations Management*, 29(4):955–972, 2020.

[9] John Holler, Risto Vuorio, Zhiwei Qin, Xiaocheng Tang, Yan Jiao, Tiancheng Jin, Satinder Singh, Chenxi Wang, and Jieping Ye. Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1090–1095. IEEE, 2019.

[10] Chao Mao, Yulin Liu, and (Max) Shen Zuo-Jun. Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. *Transportation Research Part C*, 115:102626, 2020.

[11] Shir Tavor and Tal Raviv. Anticipatory rebalancing of robotaxi systems. *Transportation Research Part C: Emerging Technologies*, 153:104196, 2023.

[12] Hai Wang and Zhengli Wang. Short-term repositioning for empty vehicles on ride-sourcing platforms. 2020.