Learning-based Dwell Time Prediction for Vehicular Micro Clouds

Max Schettler*, Gurjashan Singh Pannu*, Seyhan Ucar†, Takamasa Higuchi†, Onur Altintas†, Falko Dressler*
*School of Electrical Engineering and Computer Science, TU Berlin, Germany
†InfoTech Labs, Toyota Motor North America R&D, CA, U.S.A.
{schettler, pannu, dressler}@ccs-labs.org
{seyhan.ucar, takamasa.higuchi, onur.altintas}@toyota.com

***Abstract*—Vehicular Micro Clouds (VMCs) are an emerging development in the domain of vehicular networks posed to provide local services to users without the need for external infrastructure. This can significantly improve the user experience, in particular due to the low latencies that such systems can achieve. Due to the distributed nature of such a VMC, effective local coordination is important while using minimal communication resources. To this end, it is important to know, how long vehicles will be participating in, and contributing to a VMC. In this work, we investigate, how previous, heuristic-based approaches can be improved by incorporating local, learning-based techniques. Our analysis indicates a potential improvement of the accuracy of the prediction, and resulted in an improved simulation environment within which the learning-based approach can be deployed.**

I. Introduction

Vehicular Micro Clouds (VMCs) are a tool for providing services to users by leveraging local vehicles' resources. An effective implementation of a VMC uses local communication methods, like Vehicular-to-Vehicle (V2V) communication for exchanging both payload and control information [1], [2]. Due to this localized coordination and vehicles' inherent mobility, managing a VMC includes incorporating joining vehicles, as well as addressing vehicles' departures. Otherwise, data stored at vehicles departing the VMC can be lost. Only by continuously shifting data from these vehicles to new members, a consistent state can be maintained within the system [3].

To this end, a VMC protocol needs to accomplish this goal, while using as little resources as possible, as the shared channel is limiting the communication capacity. As such, the protocol aims to determine a schedule of data transmissions by individual vehicles that will result in maximized data lifetime with as few transmissions as possible. An optimal schedule would take into account various factors:

- current data availability within the system, i.e., each data item's amount of replication, including on which vehicles they are stored;
- positions and planned trajectories/routes of the VMC's members; and
- the state of the communication channel.

Based on this global view, important data can be derived, such as the likelihood of successful transmissions for individual vehicles and the expected (remaining) time VMC-members will remain part of the system. However, determining such an optimal schedule is infeasible: The amount of factors that influence the decision are large, the optimization goal is multi-dimensional, and the VMC is a distributed system.

As an alternative, our current protocol attempts to circumvent these complexities by avoiding costly explicit coordination among vehicles, and computing choices for data transmissions locally. To this end, each vehicle determines which data item to disseminate for the VMC based on the dwell time heuristic, i.e., each vehicle's remaining time within the VMC [3]. This design decision is based on the observation that this dwell time is a governing factor for deciding on optimal resource use: Selecting transmitted data in a way that the vehicle within the VMC that will be its member for the longest time will maximize the contribution of this transmission to overall data life time, as it reduces the amount of future required retransmissions for this data item. While this approach is not optimal due to its focus on a single metric, e.g., it does not take into consideration whether other data items could be more urgent to transmit due lower replication count, it can be computed locally, avoiding expensive coordination.

This metric cannot be known exactly in advance since it depends on when exactly a vehicle leaves the VMC. For this reason, Pannu et al. [3] derived an heuristic based on the distribution of the dwell times observed in a traffic simulation, which was then sampled to retrieve realistic values for use in the protocol. While already showing good performance, we think it can be further improved using a machine learning based approach. In this paper, we present a solution to replace the stochastic approach in [3] with a more accurate machine learning (ML)-based solution. Such a learning-based approach can be much more flexible: Instead of relying on offline-data, the route of the vehicle, current traffic patterns, and the time-of-day can be taken into account. In addition, this approach can be used as a proof-of-concept, how a VMC can supply aggregated resources for ML algorithms that optimize the VMC itself, or even other applications and services.

Our main contributions can be summarized as follows:

- we conducted a learnability analysis based on data sampled from a VMC simulation to confirm the potential benefits of the new approach, and
- we implement a VMC simulation that includes the adapted protocol, incorporating online learning to show the overall benefits on the protocol.

II. Related Work

The concept of vehicular clouds has been proposed to combine cloud computing with Vehicular Ad-Hoc Networks (VANETs) [4]–[6]. Different architectures have been proposed in the various aspects that are relevant to this new approach, such as the provided services, or networking challenges [7]. The common denominator of the designs is the utilization of vehicles' resources for providing applications [8].

Mobile Edge Computing (MEC) has been proposed in the scope of 5G to provide such services to the end users. Various MEC applications haven been put forwarded, which can be grouped into consumer-facing, third-party, and network services [9]. Even a standardized architecture for MECs has been put forward already by ETSI [10]. Similarly, originating from fog computing [11], virtualized MEC aims to use these local resources, at the networks edge, for providing services [12]. Combined, both concepts can provide VMC of different scales [13]. Comparably small, localized vehicular clouds can provide localized services such as data collection [1], [4], while larger, e.g., city-wide, clouds can be utilized for connectivity to longer-distance services.

Often data is specific to certain locations, which makes data replication and allocation an important problem to solve in this domain. Similar problems have been investigated already, e.g., to improve fault-tolerance in mobile ad hoc networks [14]. In VANETs, Lee et al. [15] exploited the vehicles' mobility to distribute data, aiding in its replication. Complementing this, pre-fetching of data has been used to improve data availability [16], [17].

Recently, advanced approaches have been proposed that are based on a more detailed understanding of vehicle mobility. In [18], the vehicle mobility is analyzed to determine a set of vehicles for data storage, where the vehicles' mobility has a low cross-correlation, reducing the likelihood of a data-item being lost. A more localized strategy has been chosen in [3], aiming to determine the locally optimal candidate for data storage based on the remaining membership time in a VMC.

Another recent trend is the use of learning-based to more efficiently solve challenges in VANETs. Using machine learning explicitly for vehicular clouds as been investigated by Liang et al. [19]. A general framework for using ML in VANETs was provided in [20]. The presented CarML approach describes a distributed ML platform built on top of an vehicular clouds. Generalizing to cooperative driving, Uhlemann [21] studied opportunities and challenges of using ML in this field.

From a tooling perspective, Schettler et al. [22] developed Veins-Gym, a framework based on the popular OpenAI Gyms for enabling the use of ML solutions within the VANET Veins.[1] To showcase its potential, an agent was trained using this tool to select the optimal communication method in a heterogeneous communication application. A similar approach was used previously for integrating data-driven approaches with the Sumo [23] and ns-3 [24] simulators. These tools enable the creation of standardized environments, which can serve as a benchmarks of common problems in their respective domains, i.e., traffic management, and (vehicular) networking. Beyond the perspective of simplifying development of flexible solutions, this approach also aids in the reproducibility of reasearch as the environments are well-defined and can be defined in a self-contained manner.

Inspired by these works, we aim to improve dwell time prediction using learning techniques to more accurately, flexibly, and robustly predict these membership times to improve the VMC's performance.

[1]https://veins.car2x.org/

III. Vehicular Micro Cloud: A Primer

The concept of MEC proposes utilizing resources at the network's edge, i.e., at, or close to, the base stations to allow for reduced latency and improved flexibility reacting to users' demands [25]. Utilizing these resources can reduce load on the backbone network, reduce centralization and, thus, improve resilience, and improve user experience. Similarly, the VMC architecture explores exploiting resources present in vehicles, as these, are and will be increasingly equipped with resources for applications such as autonomous driving [1], [13]. Such resources may be compute power, storage capabilities, and, critically, connectivity. In this architecture, the network's edge is extended to the vehicles which provide service among themselves, and other users. While parking vehicles can be used, commonly they will be moving, which adds significant complexity to managing and maintaining the VMC.

VMCs can be utilized in different situations where both a demand exists, and a sufficient amount of resources is available. Both these criteria correlate with the presence of vehicles, which in a VMC act as both servers and clients. Conversely, such situations may be busy intersections in city centers, or vehicles moving at similar velocities on a highway. In any case, as vehicles are moving individually, they need to be integrated into, or removed from a VMC as they approach or leave the vicinity without compromising its consistency.

This has important impacts on VMCs as storing data is a relevant application not only for users, but also to for internal coordination of the VMC itself. Therefore, a sufficient replication of data items needs to be maintained. To this end, protocols have been proposed which exploit knowledge about vehicles' movement, as this allows for inferring likelihoods of vehicles leaving the VMC [3], [26]. Vehicles transmit data items based on cooperative perception, which allows transmitters to consider not only data relevant for vehicles' current VMC, but also future ones they will likely join considering their projected route. To optimize transmissions utility for the local micro cloud vehicles are preferred if they are expected to remain in the current VMC for long, additionally taking into account channel utilization to avoid react avoid degrading performance of the shared channel.

IV. Dwell Time Analysis as a Learning Problem

A. What is dwell time?

Dwell time in the context of VMC is the time vehicles, i.e., the VMC's constituents, spend in its area. As the vehicles'

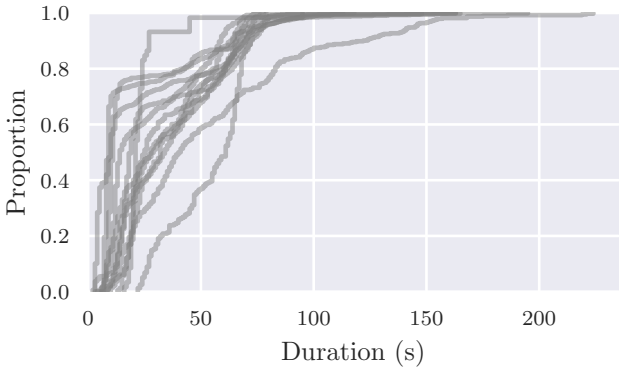


Figure 1. eCDFs of the dwell times for different intersections in Luxembourg.

mobility is a crucial reason for complexity in VANETs which protocols have to account for. Assessing the degree of mobility ahead of time can allow for performance improvements [3].

Predicting and communicating the exact trajectories of vehicles, however, is a prohibitively complex and expensive task. Alternatively, in the context of VMCs, the *dwell time*, i.e., the time a vehicle spends in a VMC, can be used, as it is a strong indicator of its mobility. Still, like the exact trajectory, the dwell time is influenced by many factors, like driver behavior (both of the vehicle itself, as well as other vehicles'), traffic lights, the desired route, or even the weather. For example, in most countries, vehicles will spend more time on an intersection when doing a left turn, compared to those doing a right turn, as they have to cross oncoming traffic. Consequently, dwell times can vary drastically between vehicles for intersections intersection. In Figure 1, we plot the dwell times for different intersections in Luxembourg using the LuST simulation scenario [27]. In particular, most intersections exhibit a non-trivial distribution for the first 60–80 %, with a long tail for the remaining vehicles.

Prediction of dwell times allows vehicles to utilize this knowledge for optimizing VMC or other VANET protocols. This is enabled by the fact that most of the aforementioned factors can be exchanged between vehicles using Vehicle-to-X (V2X) communication.

B. A Supervised Learning Model

To predict the vehicles' dwell times we formulate a supervised learning problem. The predictors will utilize only information that is locally available to a vehicle participating in a VMC. The following features are considered:

- Entry and exit location for the VMC, as a categorical label for the corresponding road.[2]
- The vehicle's current speed.
- The current time, in seconds of the day.
- The vehicle type as a categorical label.

[2]We assume the vehicle's intent to be known, such that not only the location of entry, but also that of exit from the VMC is known.

Table I
KEY PARAMETERS OF THE SETUP.

Artificial Neural Network (ANN)	
Network structure	64^4
Numerical preprocessing	Normalization
Categorical preprocessing	One-hot encoding
Optimizer	Adam[28]
Training epochs	100
Learning rate	0.01
Loss	Mean absolute error
Random forest	
Number of trees	300
Maximum tree depth	16
Minimum examples	5

We used two different learning models in this work, a Random Forest [29] (labelled as *Tree* in the figures) , as well as an ANN. The Random Forest was chosen for its resilience to overfitting and strong ability for generalization. The ANN was included, as its evaluation is generally efficient, and since it is more suitable for federated learning, which we consider employing for a continuation of this work. A detailed list of parameters used for both models can be found in Table I. In addition, the ANN uses a one-hot encoding for the categorical inputs (location, VMC, and vehicle type identifiers), as well as normalization for the numerical inputs.

During training, the validation dataset is used to evaluate the models' performance The test dataset, on the other hand, is utilized to ensure the training data is not overfitted. This was particularly important due to the comparatively large amount of epochs which we used since the prediction performance still improved even in later epochs of the training process. This is likely a consequence of the relatively complex ANN which we employed to avoid limiting on the model's performance, The training and model parameters were yielded by manual tuning, which we chose for simplicity over a formal hyper parameter optimization.

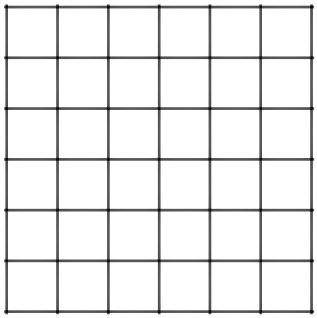
V. EVALUATION

A. Simulation Model

The simulations generating the ground truth used for this study are based on SUMO 1.10.[3] To investigate different types of intersections, both a generated Manhattan grid, as well as the Luxembourg scenario [27] was considered (see Figure 2). As the latter is a very large scenario we limited the region of interest to a set of intersections in the city center. For the Manhattan grid, each intersection constitutes its own VMC. This definition is not sensible for the Luxembourg scenario, as there are minor intersections on small roads that don't see the traffic volume required to maintain a functioning VMC. For this reason we limited the VMCs to large intersections only, which mostly correlates to the presence of traffic lights. In total, there are 13 and 49 VMCs in the Luxembourg and the Manhattan scenario, respectively.

[3]https://www.eclipse.org/sumo/

(a) Luxembourg



(b) Manhattan

Figure 2. The scenarios considered. The utilized section of the Luxembourg scenario (Figure 2a) is part of the city center and has high traffic levels. In the artificial Manhattan grid (Figure 2b) intersections are managed by traffic lights and connected by three lanes in each direction, with a distance of 200 m between each other.

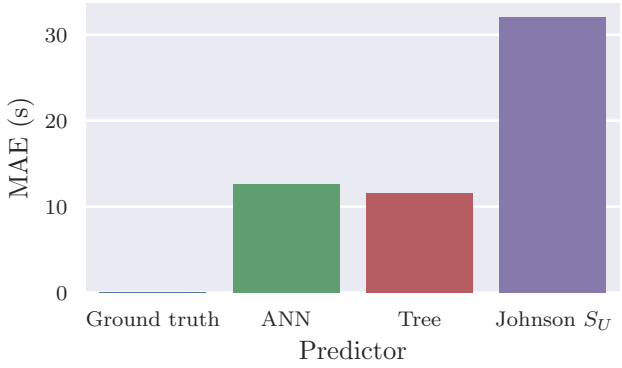


Figure 3. The accuracy of the different predictors for the Luxembourg, 8:00 scenario.

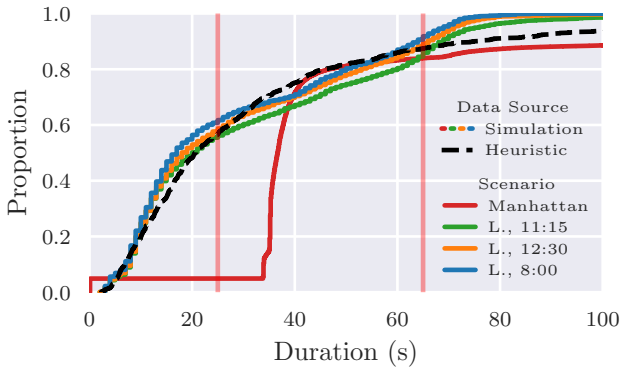


Figure 4. The eCDF of the ground-truth compared to predictions by the best fit Johnson S_U distribution.

For both scenarios 1000 s (about 16 min) of vehicle traces were recorded. For the Luxembourg scenario three different times of day (08:00, 11:15, and 12:30) are considered to account for different traffic patterns during the day, in particular including the morning rush hour. In addition, each simulation configuration was repeated eleven times, of which the first ten runs were used for training and validation, while the last run was used for testing them. In the Luxembourg simulation there are 1416, 585, and 980 unique vehicles for the 8:00, 11:15, and 12:30 time windows, respectively, while in the Manhattan grid, there are 1006 vehicles.

We compute the ground truth in a post processing step. Based on the traces of vehicles moving through the simulation we can assign the VMC for each timestep, and derive the entry- and exit points, as well as durations of each VMC visit.

B. Accuracy

As a first metric we look into the accuracy of the proposed predictors, as well as the heuristic proposed in [3] and the ground-truth. We define accuracy as the MAE between predictions and the ground-truth. In particular we chose this metric over alternatives such as the mean square root error (MSRE) to avoid exaggerating the impact of outliers. Figure 3 shows the MAE for the Luxembourg scenario. As can be seen the Johnson S_U has an average error of 25.9 s, which is on the same scale as the ground-truth measurements.

When comparing the distribution of ground-truth data and Johnson S_U generated dwell times (see Figure 4) in the Luxembourg scenario we observe, however, that the distribution cannot capture the complexity of the ground truth data, resulting in a skewed error. In particular, the heuristic underestimates the amount of the typically short dwell times (i.e., smaller than about 25 s, in the area to the first read line in the figure). In addition, the ground truth distributions have a significantly less pronounced tail (right of the second red line), leading to an overestimation of large dwell time. Even so, the parameters used for the Johnson S_U distribution are optimal, other parameterizations can reproduce some aspects of the ground truth data better, albeit only when accepting an even worse overall performance. Moreover, the Johnson S_U distribution generalizes badly for other scenarios, as is evident when comparing to dwell times experienced in the Manhattan scenario (see Figure 4), where typical values are larger, and the distribution's tail is larger.

In contrast, both the Tree and the ANN have a significantly lower MAE of 6 s and 9.5 s respectively. This is because of

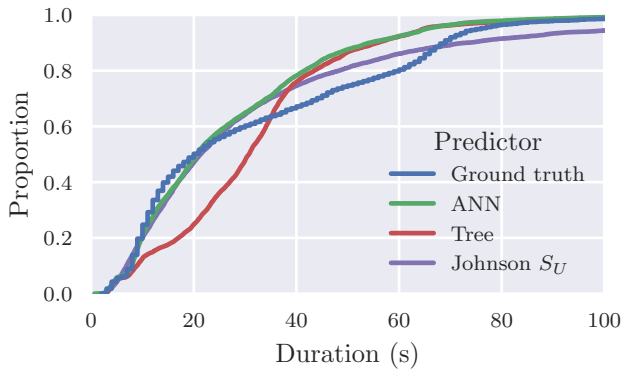


Figure 5. eCDFs of the ground-truth compared to predictions all used predictors.

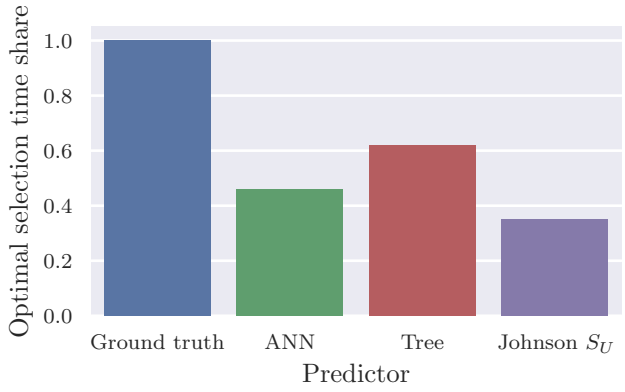


Figure 6. The fraction of time during which the use of each predictor leads to the correct candidate choice, i.e., choses the same candidate that is chosen with access to the ground truth.

their better ability to adapt to the information that is available to them, which allows them to detect the differences in dwell time distributions for individual intersections. While the overall error is quite low for these predictors, their predictions do not match the distribution of the ground truth. Figure 5 shows the distribution of each predictors' values. It can be seen that neither the heuristic, nor the learning-based approaches capture the bi-modality of the observed data.

C. Error Distributions

While the overall error allows to better understand the predictors overall performance, the error's distributions are relevant to determine how well predictions generalize to unknown situations (see Figure 7). To this end, we look at the *relative* instead of absolute errors, as this allows to better understand the magnitude of the errors: A prediction that is wrong by 10 s, has different implications when the ground truth is 30 s compared to 150 s. In addition, we train the models on only one scenario, *Luxembourg, 8:00*, to observe how well it generalizes towards other times, and scenarios.

For the scenario the models are trained on the ANN predictor appears to perform best, with the Tree-based predictor showing a larger spread of errors depending on the intersection. However, the Tree's average error is still smaller (see Figure 3), as the predictions are better for more frequented VMCs. In contrast, the Johnson S_U distribution's relative error larger in almost all cases most intersections, with most errors being of a similar magnitude to the ground truth value (i.e., the relative error being less than 1). Beyond this, a signifiant fraction of the predictions are of by at least the ground truth value, greatly overestimating the dwell time.

Beyond this, we observe that the ANN predictor fails to generalize in all other scenarios: errors are not shown in the plots as all errors exceed reasonable values. This is due to the small range of time values (1000 s) in the training time window, as the predictor overfitted to the time value. Moreover the performance for both the Johnson S_U heuristic, and the Tree predictor in the Manhattan scenario is lacking. While the relative errors are relatively small, this is merely due to the typical dwell times being larger in this scenario (see Figure 4). This an expected result for the heuristic, as it doesn't operate on the available data at all. For the Tree-based predictor we attribute this to the categorical nature of the spatial data the predictors can utilize, since they cannot infer intersection shape merely based on road labels.

For different times of day in the same scenario (Luxembourg 11:15, and 12:30, respectively), however, we notice a similar performance of both the Tree-based predictor and the heuristic. In these cases, the input includes sufficient information (in contrast to the Manhattan scenario), and the prediction generalizes well enough, with the mean relative error being less than 30 % larger (0.54 compared to 0.65 and 0.69, respectively).

D. Discussion

While the results discussed previously indicate a significantly reduced error compared to the heuristic, the change in protocol efficiency based on the improved predictor may differ. For example, it is easy to think of a prediction with an arbitrarily high MAE (e.g., an multiple of the ground-truth), which still results in the exact same candidate selection. Moreover, VMC management protocol is not known to be optimal, so there may exist a set of predictions yielding a selection that outperforms the original algorithm.

To better understand the relation between the completely data-centric metrics, such as the MAE, and the implications for the protocol performance we compared the candidate selection resulting from the different predictors. To this end, we simulated the state of an arbitrary VMC for a dedicated simulation run, including dwell time predictions by all approaches described above. Based on this input we can determine which candidate would be selected by the protocol when utilizing a given predictor at all times during the simulation.

The results of this simulation indeed indicate a mismatch between prediction performance and the resulting protocol efficiency. Figure 6 shows the fraction of time during the simulation during which the utilized predictor resulted in the

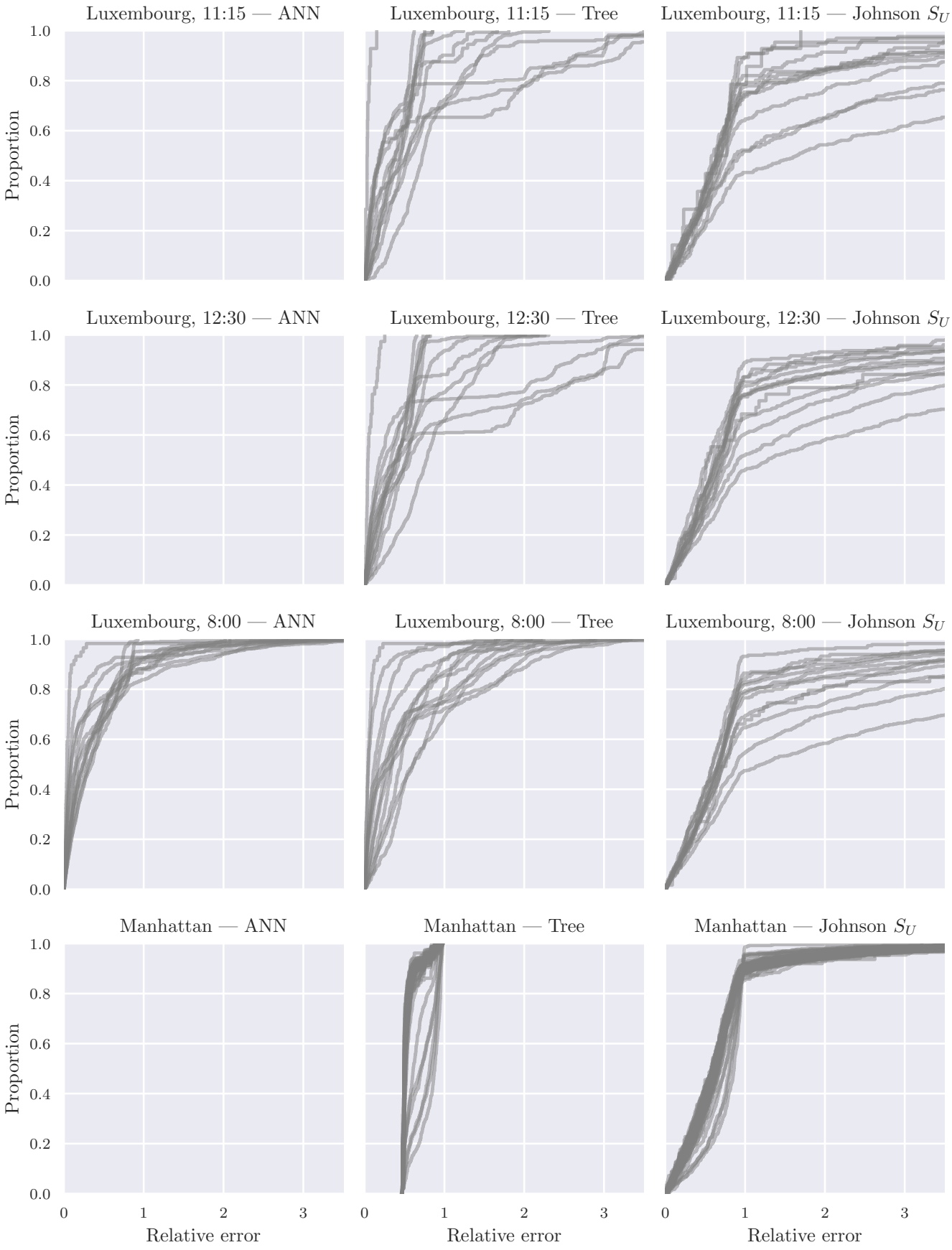


Figure 7. eCDFs of relative prediction errors for all predictors (columns) and scenarios (rows). Individual VMCs are displayed in gray lines.

same selection as having access to the ground-truth. Naturally, this ratio is 1 for the ground-truth itself. The learning-based predictors still result in the correct selection more than half of the time, while this value drops down to 31 % when using the heuristic. While the predictors' rankings in this metric match those observed previously, the relative distances have changed significantly.

While this simulation is short of a full integration into a VMC simulation and disregards certain aspects, such as the imperfect knowledge of individual nodes due to the distributed nature of the overall system, this comparison can serve as a first data point for how the true relation behaves.

VI. Conclusion

In this paper, we investigated the feasibility of a learning-based approach to dwell time prediction. We have conducted a detailed analysis of the data available to a supervised learning approach and outlined the potential improvements over current, heuristic-based approaches. The regressors that were created in the process of this study show significant improvements in accuracy for predicting dwell times. In particular, the MAE for predictions was reduced by more than 60 %.

Additionally, a detailed investigation demonstrated the potential for data-driven prediction to be more flexible when applied to unknown scenarios, while also showing pitfalls that can result in adverse performance. This underlines the importance of feature engineering when utilizing ML solutions to exhaust this potential.

Moreover, a high-level simulation of the protocol utilizing the improved prediction was conducted. The results indicate that, while the improvements also impact these metrics, as evident in the 76 % increased time of correct candidate selection, this relation to the error is very non-linear. This showcases the need for integrated analysis of not only the learning algorithm in isolation, but rather in combination with the intended system. In future work we intend to continue work in this avenue, further increasing the simulated level of detail, and with a deeper integration between VMCs and ML solutions.

References

[1] F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Vehicular Micro Clouds as Virtual Edge Servers for Efficient Data Collection," in *ACM MobiCom 2017, CarSys Workshop*, Snowbird, UT: ACM, Oct. 2017, pp. 31–35.

[2] F. Dressler, G. S. Pannu, F. Hagenauer, M. Gerla, T. Higuchi, and O. Altintas, "Virtual Edge Computing Using Vehicular Micro Clouds," in *IEEE ICNC 2019*, Honolulu, HI: IEEE, Feb. 2019.

[3] G. S. Pannu, S. Ucar, T. Higuchi, O. Altintas, and F. Dressler, "Dwell Time Estimation at Intersections for Improved Vehicular Micro Cloud Operations," *Elsevier Ad Hoc Networks*, vol. 122, p. 102 606, Nov. 2021.

[4] M. Gerla, "Vehicular Cloud Computing," in *IFIP/IEEE Med-Hoc-Net 2012*, Ayia Napa, Cyprus: IEEE, Jun. 2012, pp. 152–155.

[5] M. Eltoweissy, S. Olariu, and M. Younis, "Towards Autonomous Vehicular Clouds," in *International ADHOCNETS 2010*, J. Zheng, D. Simplot-Ryl, and V. C. M. Leung, Eds., Victoria, Canada: Springer, Aug. 2010, pp. 1–16.

[6] F. Dressler, P. Handle, and C. Sommer, "Towards a Vehicular Cloud - Using Parked Vehicles as a Temporary Network and Storage Infrastructure," in *ACM MobiHoc 2014, WiMobCity Workshop*, Philadelphia, PA: ACM, Aug. 2014, pp. 11–18.

[7] E. Lee, E.-K. Lee, M. Gerla, and S. Oh, "Vehicular Cloud Networking: Architecture and Design Principles," *COMMAG*, vol. 52, no. 2, pp. 148–155, Feb. 2014.

[8] A. Boukerche and R. E. De Grande, "Vehicular Cloud Computing: Architectures, Applications, and Mobility," *COMNET*, vol. 135, pp. 171–189, Apr. 2018.

[9] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[10] ETSI, "Mobile Edge Computing (MEC), Framework and Reference Architecture," ETSI, Sophia Antipolis, France, GS MEC 003 V1.1.1, Mar. 2016.

[11] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[12] F. Dressler, C. F. Chiasserini, F. H. P. Fitzek, et al., "V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing," arXiv, cs.NI 2106.10063, Jun. 2021.

[13] T. Higuchi, J. Joy, F. Dressler, M. Gerla, and O. Altintas, "On the Feasibility of Vehicular Micro Clouds," in *IEEE VNC 2017*, Turin, Italy: IEEE, Nov. 2017, pp. 179–182.

[14] P. Bellavista, A. Corradi, and E. Magistretti, "REDMAN: an Optimistic Replication Middleware for Read-only Resources in Dense MANETs," *Pervasive and Mobile Computing*, vol. 1, no. 3, pp. 279–310, Sep. 2005.

[15] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi, "Mobeyes: Smart Mobs for Urban Monitoring with a Vehicular Sensor Network," *IEEE Wireless Communications*, vol. 13, no. 5, pp. 52–57, Oct. 2006.

[16] Y. Rao, H. Zhou, D. Gao, H. Luo, and Y. Liu, "Proactive Caching for Enhancing User-Side Mobility Support in Named Data Networking," in *International IMIS 2013*, Taichung, Taiwan: IEEE, Jul. 2013.

[17] D. Grewe, S. Schildt, M. Wagner, and H. Frey, "ADePt: Adaptive Distributed Content Prefetching for Information-Centric Connected Vehicles," in *IEEE VTC 2018-Spring*, Porto, Portugal: IEEE, Jun. 2018.

[18] T. Higuchi, G. S. Pannu, F. Dressler, and O. Altintas, "Content Replication in Vehicular Micro Cloud-based Data Storage: A Mobility-Aware Approach," in *IEEE VNC 2018*, Taipei, Taiwan: IEEE, Dec. 2018.

[19] L. Liang, H. Ye, and G. Y. Li, "Toward Intelligent Vehicular Networks: A Machine Learning Framework," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 124–135, Feb. 2019.

[20] A. Du, Y. Shen, and L. Tseng, "CarML: distributed machine learning in vehicular clouds," in *ACM MobiCom 2020*, Virtual Conference: ACM, Sep. 2020.

[21] E. Uhlemann, "Time for Autonomous Vehicles to Connect [Connected Vehicles]," *VTMag*, vol. 13, no. 3, pp. 10–13, Sep. 2018.

[22] M. Schettler, D. S. Buse, A. Zubow, and F. Dressler, "How to Train your ITS? Integrating Machine Learning with Vehicular Network Simulation," in *IEEE VNC 2020*, Virtual Conference: IEEE, Dec. 2020.

[23] E. Vinitsky, A. Kreidieh, L. L. Flem, et al., "Benchmarks for reinforcement learning in mixed-autonomy traffic," in *Conference CoRL 2018*, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., ser. CoRL, vol. 87, Zürich, Switzerland: PMLR, Oct. 2018, pp. 399–409.

[24] P. Gawłowicz and A. Zubow, "ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research," in *ACM MSWiM 2019*, Miami Beach, FL: ACM, Nov. 2019.

[25] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, Oct. 2017.

[26] G. S. Pannu, F. Hagenauer, T. Higuchi, O. Altintas, and F. Dressler, "Keeping Data Alive: Communication Across Vehicular Micro Clouds," in *IEEE WoWMoM 2019*, Washington, D.C.: IEEE, Jun. 2019.

[27] L. Codeca, R. Frank, and T. Engel, "LuST: a 24-hour Scenario of Luxembourg City for SUMO Traffic Simulations," in *SUMO SUMO 2015*, Berlin, Germany, May 2015.

[28] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International ICLR*, San Diego, CA, May 2015.

[29] L. Breiman, "Random Forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.