# Anticipatory Vehicle Routing for Same-Day Pick-up and Delivery using Historical Data Clustering

Jelmer van Lochem[1,2], Maximilian Kronmueller[1], Pim van 't Hof[2] and Javier Alonso-Mora[1]

***Abstract*— In this paper we address the problem of same-day pick-up and delivery where a set of tasks are known a priori and a set of tasks are revealed during operation. The vehicle routes are precomputed based on the known and predicted requests and adjusted online as new requests are revealed. We propose a novel anticipatory insertion method which incorporates a set of predicted requests to beneficially adjust the routes of a fleet of vehicles in real-time. Requests are predicted based on historical data, which is clustered in advance. We exploit inherent patterns of the demand, which are captured by historical data and include them in a dynamic vehicle routing solver based on heuristics and adaptive large neighborhood search. The proposed method is evaluated using numerical simulations on a variety of real-world problems with up to 1655 requests per day. Their degree of dynamism ranges from 0.70 to 0.93. These instances represent dynamic multi-depot pickup and delivery problems with time windows. The method has shown to require less driven kilometers than comparable methods.**

## I. INTRODUCTION

The transportation of people and goods in a reliable, efficient and timely manner has grown to be more important than ever. Roads and cities are becoming increasingly congested and the impact of greenhouse gasses can already be observed. The need for controlling transportation systems and specifically fleets of vehicles more efficiently is therefore steadily increasing. The family of problems tackling such situations is called vehicle routing problems. A subclass are stochastic and dynamic vehicle routing problems (SDVRP), which are characterized by the reveal of information during operation and the stochastic nature of it. A broad overview summarizing the last three decades can be found in Psaraftis et al. [1]. If orders need to be fulfilled before the end of the current day we often call this situation same-day delivery (SDD). Previous work has shown that anticipating the future can improve the efficiency of dynamic vehicle routing problems. In this work we show the potential of using historical data for predictive same-day deliveries.

### A. Related Work

The field of anticipatory routing attempts to solve DVRPs by anticipating additional requests and subsequently adjusting routes according to these expectations.

This research was supported in part by ORTEC, Amazon Research Award and Ahold Delhaize. All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers and/or sponsors.

[1] Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: me@jelmervanlochem.nl; m.kronmuller@tudelft.nl; j.alonsomora@tudelft.nl).

[2] ORTEC B.V., 2719 EA Zoetermeer, The Netherlands (e-mail: pim.vanthof@ortec.com)

Van Hemert and La Poutre [2] define regions where requests are likely to occur as fruitful regions and selectively route vehicles trough those regions. Ghiani et al. [3] propose sampling requests from an assumed future arrival distribution and add these sampled requests to the current problem definition. Both insert newly revealed requests in their current solution. Changing routes prior to solving by incorporating a representation of the expected request is called *anticipatory insertion*.

Bent and Van Hentenryck [4] introduce the *multiple scenario approach* (MSA), which creates different scenarios by sampling requests from an assumed future arrival distribution multiple times. Upon arrival of an additional request, the best insertion location within all scenarios is determined. The insertion location which is best most frequently, among 50 scenarios using a consensus function, is then chosen. Azi et al. [5] and Voccia et al. [6] applied MSA to solve variations of SDD. Ghiani et al. [7] compared the performance of anticipatory insertion to the multiple scenario approach in solving the dynamic and stochastic TSP. They concluded that anticipatory insertion offers comparable performance to MSA while requiring less computational resources.

Additional strategies are so called *waiting strategies* which keep vehicles waiting at locations for which nearby future requests are expected. Thomas [8] shows that in a single vehicle case, on instances with up to 50 customers, waiting strategies which make use of stochastic information can improve upon strategies which do not. Ichoua et al. [9] show comparable results on larger problem instances with up to 6 vehicles and 200 customers.

A common shortcoming of all mentioned methods is that they use identical assumed arrival distributions for their predictions and their expectation of additional future requests. The work in this paper differs by deriving the predictions from historical data and testing them on untouched test sets. Further, the problem instances solved here are considerably larger than the ones solved in the mentioned papers.

### B. Contribution

We propose a method for anticipatory routing, which derives predicted requests from historical data. The method does not rely on assumed arrival probabilities but builds on patterns of the demand captured by historical data. Further, we experimentally evaluate the performance of the proposed approach.

The remaining paper is organized as follows. Section II introduces a formulation of the underlying problem. In Section III the proposed method is described. Section IV presents the performed experiments and the corresponding results. The paper is concluded in Section V.

## II. PROBLEM FORMULATION

The same-day delivery problem we face in this work can be described as a dynamic multi-depot pickup and delivery problem with time windows. This section first formally introduces the static problem definition in Section II-A. Secondly, the dynamic problem is described in Section II-B based on the definition of the static problem.

### A. Static Problem

Formally, let $G = (N, A)$ be a directed graph where $N$ defines a set of nodes and $A$ defines a set of arcs. $N$ consists of $n$ pickup nodes ($P$), $n$ corresponding delivery nodes ($D$), $m$ vehicle starting nodes and $m$ end nodes. $R$ defines a set of requests of size $n$ where each request $r_i$ is defined by the pickup node $N_i$ and delivery node $N_{n+i}$. Each pickup node $N_i$ has an associated positive load $q_i$ which should be picked up at that node. This load is required to be delivered to its corresponding delivery node $N_{n+i}$ which means its associated load $q_{n+i}$ has the same value, only negative. All pickup and delivery nodes also have a non-negative service duration $d_i$ which defines for how long a vehicle should stay at node $N_i$ before it is considered to be serviced. The service of each node $N_i$ should start within a uniquely configurable time window starting at $e_i$ and ending at $l_i$. Let $K$ define a set of vehicles of length $m$ where each vehicle $v_k$ should start its route at starting node $N_{2n+k}$ and end its route at end node $N_{2n+m+k}$. Furthermore each vehicle $v_k$ has a limited capacity equal to $Q_k$. Indirectly, each vehicle $v_k$ also has a time window during which it is able to service pickup and deliver nodes. This time window is defined by the time windows of the start and end node of the route. More specifically, vehicles may only depart from their starting node $N_{2n+k}$ after a time $e_{2n+k}$ and should be at their end node before a time $l_{2n+m+k}$. Lastly, all arcs in $A$ from node $N_i$ to node $N_j$ are defined by an associated travel cost $c_{i,j}$ and travel time $t_{i,j}$. The objective of the problem is to find a set of routes, of minimal traveling cost, servicing all pickup and deliver nodes while meeting all constraints. The problem, using a three-index vehicle flow formulation, can be defined as described by Equations 1 to 17.

$$\min_{x_{i,j}} \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{i,j}^k x_{i,j}^k \tag{1}$$

subject to :

$$x_{i,j}^k \in \{0,1\} \qquad i \in N, j \in N, k \in K \tag{2}$$

$$x_{i,i}^k = 0 \qquad i \in N, k \in K \tag{3}$$

$$\sum_{k \in K} \sum_{j \in N} x_{i,j}^k = 1 \qquad i \in P \tag{4}$$

$$\sum_{j \in N} x_{j,i}^k - \sum_{j \in N} x_{i,j}^k = 0 \qquad i \in P \cup D, k \in K \tag{5}$$

$$\sum_{j \in N} x_{i,j}^k - \sum_{j \in N} x_{n+i,j}^k = 0 \qquad i \in P, k \in K \tag{6}$$

$$x_{2n+l,j}^k = 0 \qquad k \in K, l \in K \backslash \{k\}, j \in N \tag{7}$$

$$x_{i,2n+m+l}^k = 0 \qquad k \in K, l \in K \backslash \{k\}, i \in N \tag{8}$$

$$\sum_{j \in N} x_{2n+k,j}^k = 1 \qquad k \in K \tag{9}$$

$$\sum_{j \in N} x_{j,2n+k}^k = 0 \qquad k \in K \tag{10}$$

$$\sum_{i \in N} x_{i,2n+m+k}^k = 1 \qquad k \in K \tag{11}$$

$$\sum_{j \in N} x_{2n+m+k,j}^k = 0 \qquad k \in K \tag{12}$$

$$B_j^k \geq \left(B_i^k + d_i + t_{i,j}\right) x_{i,j}^k \qquad i \in N, j \in N, k \in K \tag{13}$$

$$B_i^k + d_i + t_{i,n+i} \leq B_{n+i}^k \qquad i \in P, k \in K \tag{14}$$

$$e_i \leq B_i^k \leq l_i \qquad i \in N, k \in K \tag{15}$$

$$Q_j^k \geq \left(Q_i^k + q_j\right) x_{i,j}^k \qquad i \in N, j \in N, k \in K \tag{16}$$

$$\max(0, q_i) \leq Q_i^k \leq \min(Q_k, Q_k + q_i) \quad i \in N, k \in K \tag{17}$$

The objective function (Equation 1) calculates the total travel cost. The total travel cost is the sum over the individual costs of all traversed arcs. The set of binary variables $x_{i,j}^k (i \in N, j \in N, k \in K)$ indicate which arcs from node $N_i$ to node $N_j$ are traversed by each vehicle $v_k$. The objective function is subject to multiple constraints. The first set of constraints (2) ensures that the set of variables $x_{i,j}^k$ can only be binary. Constraints (3) ensure that all arcs have different starting and end nodes. Constraints (4) ensure that each pickup is only served once by a single vehicle. Constraints (5) ensure that each individual pickup and delivery node has as many arcs towards it as there are arcs originating from it. Constraints (6) ensure that for each vehicle the number of arcs originating from a delivery node is equal to the number of arcs originating from its corresponding pickup node. Together constraints (4), (5) and (6) ensure that each pickup is assigned to a single vehicle, that the corresponding delivery is assigned to the same vehicle and that continuous routes are formed. Constraints (7) ensure that the starting node for each vehicle can only be serviced by its appropriate vehicle. Similarly, constraints (8) ensure that the end node for each vehicle can only be serviced by its appropriate vehicle. Constraints (9), (10) ensure that a single arc originates from each start node and no arcs go towards each of them, forcing them to be the starting nodes. Similarly, constraints (11), (12) ensure that a single arc goes towards each end node and no arcs originate from each of them, forcing them to be the end nodes. Constraints (13) ensure that the time between the start of service of two nodes, when an arc occurs, is at least equal to the traveling time between the two nodes and

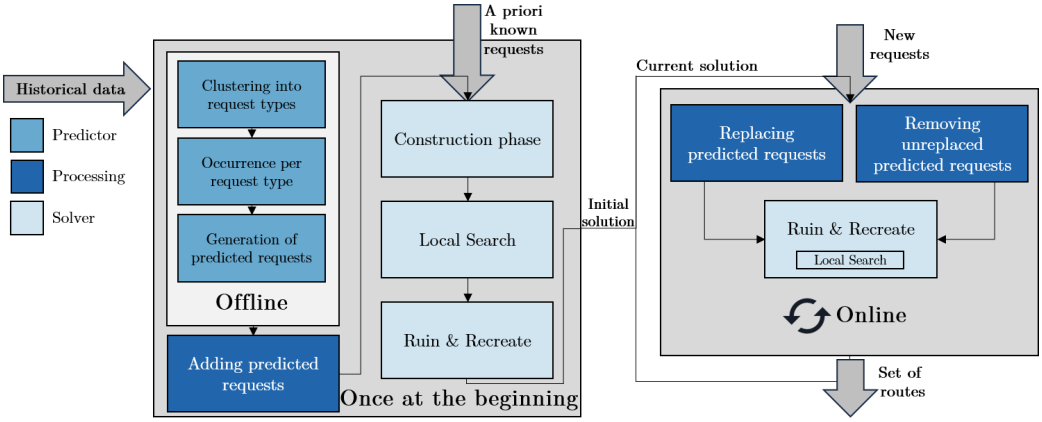


Fig. 1: Schematic overview of our solution approach.

the service duration of the first node. These constraints also ensure that sub-tours are eliminated. Constraints (14) ensure that each pickup node is serviced before its corresponding deliver node. Constraints (15) ensure that the start of service for each node is within its defined time window. Constraints (16) ensure that the load on board of each vehicle after servicing a node is at least equal to the load it had on board at the previous node plus the load of the node it just serviced. Together with constraints (17) this ensures that the upper bound on the capacity of each vehicle is not exceeded. This three-index vehicle flow formulation is similar to Cordeau and Laporte [10], but it additionally allows vehicles to have unique start and end locations, different from a single depot case.

### B. Dynamic Problem

The static problem described in Section II-A becomes dynamic with the introduction of a release time $k_i$ for each request at which it becomes known. The operation is carried out along a time dimension, on which these events occur. As a result the static problem definition will change at discrete time events and many updated static problem variations are required to be solved consecutively.

## III. METHOD

We propose an anticipatory insertion method which incorporates a set of predicted requests prior of solving. Requests are predicted based on historical data, which is clustered in advance. We exploit inherent patterns in the demand captured by the historical data. Additionally, we developed a dynamic vehicle routing solver which makes use of a range of heuristics and an adaptive large neighborhood search. We divide our solution approach into three different components: Predicting requests (Predictor), Section III-A, incorporating, replacing and removing requests (Processing), Section III-B, and solving the resulting problem (Solver), Section III-C. Our solution approach is graphically summarized in Figure 1.

### A. Request Prediction

The basic idea for request prediction is to group similar historical requests such that potential patterns in their occurrence can be exploited. The process consists of three parts: First, historical requests are clustered on relevant dimensions to create groups named request types. Second, the number of occurrences of each request type is predicted using a prediction model. Lastly, the predicted number of requests for each request type are generated.

Request types are generated by clustering historical requests using hierarchical agglomerative complete linkage clustering (HACLC) [11]. HACLC starts with each object being its own cluster. Next, the two clusters which have the minimum distance to each other, using a predefined similarity measure, are merged into a new single cluster. The distance between two clusters is defined by the maximum distance between two objects belonging to either one of the clusters. This is done iteratively until a predefined maximal distance is reached. By choosing this maximal distance the final number and composition of the clusters can be tuned. A set of requests is first clustered on a portion of the relevant dimensions to separate the complete set into multiple subsets. These smaller subsets are then further separated. We use 10 subsequent levels of clustering. The similarity measures are individually set on each cluster level as a percentage of the evaluated dimension, for example, 10% of the median trip distance. A visualization of HACLC in one dimension is shown in Figure 2. After the generation of the request types, the number of future occurrences during a time frame for each request type are predicted. As our prediction model we use the Relative Frequency of Occurrence ($r_{fo}$), also called empirical probability. It describes the ratio of the number of outcomes in which a specified event occurs with respect to the total number of trials. The number of trials is determined by the number of comparable time frames available in historical data. Lastly, the predicted number of requests for each request type are generated. The nearest node of the road network to the mean of all requests belonging to a request type is used as the representation of the corresponding request.

### B. Incorporation of Predicted Requests

Predicted requests are used to alter the solution of a VRP such that later appearing requests can more likely be served more efficiently when they become known. The

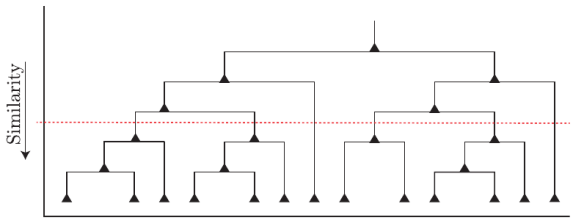



Fig. 2: Visualization of hierarchical clustering. The red dashed line visualizes the chosen measure of similarity at which the clusters are cut. This determines the number and composition of the clusters.

incorporation procedure of predicted requests is three-fold: Adding, replacing and removing requests. These methods ensure that any structure within the solution of a dynamic vehicle routing problem, imposed by the presence of predicted requests, is preserved.

Before the start of the operation the predicted requests are added to the already known ones, together they build the starting point of solving the routing problem. Each time a new request is placed during operation, it is determined if one of the incorporated predicted requests is similar enough to be replaced in the current solution and if so, which one. To enable this operation, a request classifier is introduced. This classifier uses the generated request types, cluster levels and similarity measures as the prediction process to match new requests to predicted requests. In case an additional request is dissimilar to all request types, the additional request is considered to not match and thus no predicted request is removed. If a single request type remains as a matching candidate after classification, a predicted request belonging to that request type is replaced if one resides within the current solution. If multiple request types remain as candidates, the one for which the cluster center is closest to the additional request is chosen as the final candidate. If a predicted request is indeed replaced, the prediction is removed and the corresponding new request is directly inserted using sequential cheapest insertion.

Unmatched predicted requests are removed according to two criteria. First, predicted requests are removed when the current time exceeds the time at which they were supposed to be made known. Predicted requests are also removed when a vehicle is about to start traveling towards their pickup location.

### C. Solver Design

The solver passes through three phases, Construction [12], [13], Local Search (LS) [13], [14] and Ruin & Recreate (R&R) [12], [15], [16], [17]. Construction and LS are only performed once at the beginning, while R&R is performed over and over again as new requests arrive. This process is depicted in Figure 3.

The construction method is required to create an initial solution. We developed a heuristic which makes use of the relatedness of requests among themselves. It tries to insert requests which are clustered together into a single route. This method is therefore named cluster insertion. It shows

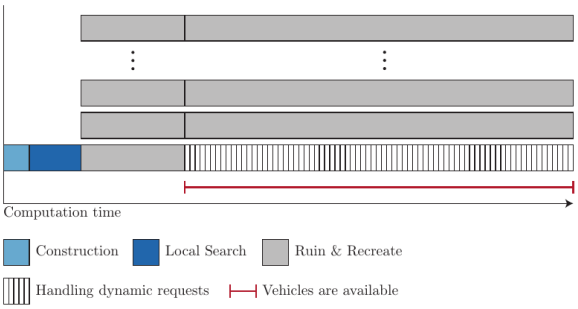



Fig. 3: Phases of the developed vehicle routing solver. Each block represents a process. Vertically stacked blocks are processes that are executed in parallel.

performance comparable to parallel cheapest insertion while requiring approximately ten times less computation time.

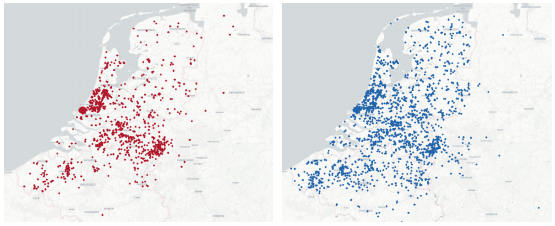
The LS phase uses the rearrange, shift and 2-opt operators. The R&R phase is inspired by the procedure described by Ropke and Pisinger [12]. We deviated from their approach by implementing several additional removal methods as proposed by Laporte et al. [17]. We additionally use all removal methods equally often. For insertion heuristics we chose the frequency to use them based on their computation time, fast ones are used in the beginning, if the solution doesn't improve for several R&R phases more expensive heuristics are used. Finally, the R&R process is parallelised by letting multiple processes run simultaneously. When any of these processes finds a solution improvement during an iteration, the improved solution is directly copied to a global best solution shared by all procedures. For this solution a short LS phase is applied in advance of continuing the R&R phase.

## IV. EXPERIMENTAL ANALYSIS

### A. Experimental Setup

To perform a qualitative analysis of the performance of our proposed method (anticipatory method (A)), we compare the implementation to three other optimization strategies for solving DVRPs. The implementations of these strategies use parts of the same solver as described in Section III-C. This allows for a relatively fair comparison which is not subject to the quality of the implementation. The three comparison strategies are:

- **Reactive + Cheapest Insertion (CI)** This strategy is added to serve as an upper bound. The strategy consists of creating an initial solution based on the requests which are known before the operation starts. The solver goes through the construction and following LS phase to create the initial solution. To incorporate additional requests only a sequential cheapest insertion heuristic is used to insert requests into the solution as they arrive. No further re-optimization is performed. This strategy is often also referred to as the greedy approach.
- **Reactive + R&R (RR)** This strategy is added to serve as a close competitor to the anticipatory method. The strategy consists of constructing an initial solution



(a) Pickup locations (b) Drop off locations

Fig. 4: Pick up and drop off locations as they are distributed of the combined instance A and F.

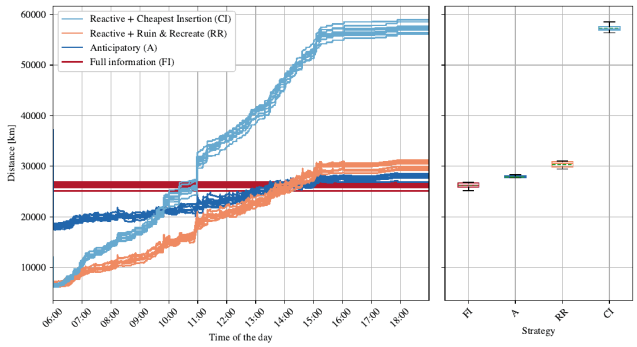
followed by the LS phase. Dynamic requests are incorporated using the sequential cheapest insertion heuristic as in the previous strategy. Additionally, R&R processes are run which constantly try to re-optimize the part of the solution which has not yet been realized or is being realized. This strategy is also referred to as the reactive approach.

- **Full information (FI)** This strategy is added to serve as a lower bound. During this strategy it is assumed that all additional requests are already known right from the start. The initial and final solution is therefore created based on all requests. In creating this solution the solver goes through the phases as visualized in Figure 3.

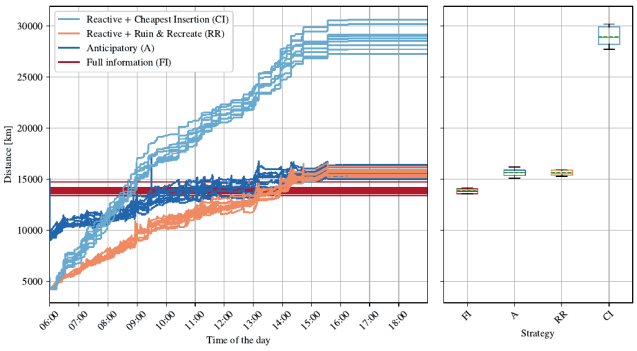
The problems to be solved are a range of real-world problem instances with up to 1655 requests per day. These instances represent dynamic multi-depot pickup and delivery problems with time windows. All four strategies are simulated 10 times on the 10 different instances representing one day (A - J), in total representing the workdays of two weeks. The data was provided by a transportation company who deals with urgent deliveries of flowers to and from the growers, distributors and auctions in The Netherlands and Belgium. A visualization of the pick-up and drop-off locations is shown in Figure 6. 500 vehicles are assumed to be available from 06:00:00 AM to 23:59:59 PM located at five depot locations. Each vehicle is required to return to the same depot location as the one it originated from. Furthermore, vehicles are assumed to have a capacity of 43.0, which equals the maximal size of requests. The pickup and deliver time windows end later than 06:00:00 AM and start before 18:00:00 PM. The time window length varies from 1h to nearly 24h. The degree of dynamism [18] of the described instances ranges from 0.70 to 0.93. Within the simulation it is checked every 30 seconds whether additional requests have arrived and if so, these are added to the problem definition; in between the problem is considered to be static. Van Lochem [19] contains more details on some aspects of this paper, including decisions on methods, actual implementations, details of the used data set, details on carrying out the experiments and parameter choices.

### *B. Results*

The results on the instances where the anticipatory method performs best (instance F) and worst (instance I) as compared to the reactive approach are shown in Figures 5a and 5b.




(a) Best instance, F.

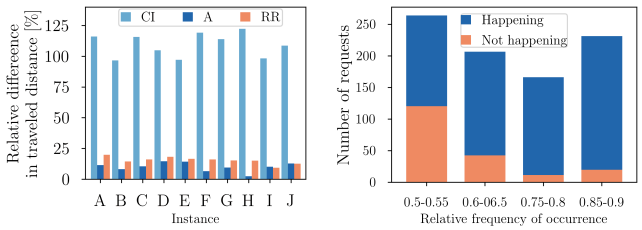



(b) Worst instance, I.

Fig. 5: Performance of different optimization strategies on the best and worst instance. On the left it is shown how the distance that is planned to be traveled during the entire day evolves during the day for each strategy. Each separate line represents a single simulation. On the right the distributions of the total distance traveled at the end of the day for each strategy are shown.

In each of these Figures, on the left, it is shown how the amount of distance that is planned to be traveled during the entire day evolves during the day, for each strategy. Each line represents a single simulation. On the right the distribution of the total distance that is required to be traveled by all vehicles is summarized for each strategy. As only a relatively small portion of the requests is known at 06:00:00 AM, solutions of both the reactive and greedy approach require relatively little distance to be traveled initially. As predicted requests are added to the problem in the anticipatory method, the initial solution requires more distance to be traveled. The averaged results over all instances compared to the full information strategy are shown in Figure 6a. Using the greedy approach, on average, $109.32 \pm 9.09\%$ more distance is required to be traveled. The reactive approach, on average, needs $15.40 \pm 2.75\%$ and the anticipatory method, on average, only requires $10.07 \pm 3.52\%$ more distance to be traveled as compared to the full information strategy.

In comparing the performance of the reactive approach and the anticipatory method it can be seen (Figure 5b), than even when the anticipatory method performs worst, as compared to the reactive approach, the confidence intervals nearly overlap completely and only a small difference in the




(a) Relative differences (b) Quality of predictions

Fig. 6: Left: Relative difference in average distance that is required to be traveled for three strategies as compared to the full information strategy, on all instances. Right: Visualizing the quality of the predictions using the anticipatory method for instance A.

average distance traveled can be observed. It can therefore be concluded that, on the evaluated instances and when making use of the described solver, the anticipatory method does not significantly perform worse in any case. When the anticipatory method performs best however, as compared to the reactive approach (Figure 5a), the confidence intervals do not overlap at all and the anticipatory method significantly outperforms the reactive approach. Using the anticipatory method, on average, $4.58 \pm 3.48\%$ less distance is required to be traveled. Additionally, the anticipatory method also requires fewer vehicles to fulfill all requests. On average, $3.34 \pm 0.74\%$ fewer vehicles are required.

### C. Prediction Quality

The improvement of the anticipatory method over the reactive approach is closely linked to the prediction quality of future requests. To evaluate this quality we first label new requests according to the classifier described in Section III-B. Subsequently, the quality of prediction is determined through a comparison of the labels of the predicted requests and the labels of the new requests. Figure 6b shows the amount of correctly predicted requests for instance A against the $r_{fo}$ as percentage of the total amount of received requests. It can be seen that a larger portion of requests is predicted correctly in case the $r_{fo}$ is relatively high. In total 78% of the predicted requests for instance A were predicted correctly. For all instances on average 61.8%, varying between 40% and 80%, of the requests were predicted such that they had a counterpart. This supports our hypothesis that similar requests which previously have occurred according to certain patterns will also occur according to the same patterns in the future. If reality deviates from the observed patterns, regardless of the cause, a relatively worse prediction is generated and therefore likely worse results are obtained.

## V. CONCLUSIONS

This work introduced an anticipatory insertion method that uses predictions of future requests based on historical data. First, a method for predicting requests based on historical data was introduced. Second, methods for adding, replacing and removing predicted requests are established. These methods ensure that any structure within the solution of a dynamic vehicle routing problem, imposed by the presence of predicted requests, is preserved while re-optimization can be performed to let the solution adapt to new information. The solver combines anticipatory routing with an adaptive variable large neighborhood search approach. The entire method was evaluated on 10 real-world problem instances with up to 1655 requests per day. The proposed anticipatory routing method is able to improve upon a competitive reactive approach by 4.58% on average in terms of the distance that is required to be traveled.

## REFERENCES

[1] Psaraftis, "Dynamic vehicle routing problems: Three decades and counting - Networks - Wiley Online Library," 2016.
[2] J. I. van Hemert and J. A. La Poutré, "Dynamic Routing Problems with Fruitful Regions: Models and Evolutionary Computation," in *Parallel Problem Solving from Nature - PPSN VIII*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 692–701.
[3] G. Ghiani, E. Manni, A. Quaranta, and C. Triki, "Anticipatory algorithms for same-day courier dispatching," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, no. 1, pp. 96–106, Jan. 2009.
[4] R. W. Bent and P. Van Hentenryck, "Scenario-Based Planning for Partially Dynamic Vehicle Routing with Stochastic Customers," *Operations Research*, vol. 52, no. 6, pp. 977–987, Dec. 2004.
[5] N. Azi, M. Gendreau, and J.-Y. Potvin, "A dynamic vehicle routing problem with multiple delivery routes," *Annals of Operations Research*, vol. 199, no. 1, pp. 103–112, Oct. 2012.
[6] S. A. Voccia, A. M. Campbell, and B. W. Thomas, "The Same-Day Delivery Problem for Online Purchases," *Transportation Science*, vol. 53, no. 1, pp. 167–184, May 2017.
[7] G. Ghiani, E. Manni, and B. Thomas, "A Comparison of Anticipatory Algorithms for the Dynamic and Stochastic Traveling Salesman Problem," *Transportation Science*, vol. 46, no. 3, pp. 374–387, Aug. 2012.
[8] B. W. Thomas, "Waiting Strategies for Anticipating Service Requests from Known Customer Locations," *Transportation Science*, vol. 41, no. 3, pp. 319–331, Aug. 2007.
[9] S. Ichoua, M. Gendreau, and J.-Y. Potvin, "Exploiting Knowledge About Future Demands for Real-Time Vehicle Dispatching," *Transportation Science*, vol. 40, no. 2, pp. 211–225, May 2006.
[10] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem: models and algorithms," *Annals of Operations Research*, vol. 153, no. 1, pp. 29–46, Jun. 2007.
[11] D. Defays, "An efficient algorithm for a complete link method," *The Computer Journal*, vol. 20, no. 4, pp. 364–366, Jan. 1977.
[12] S. Ropke and D. Pisinger, "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows," *Transportation Science*, vol. 40, no. 4, pp. 455–472, Nov. 2006.
[13] O. Bräysy and M. Gendreau, "Vehicle Routing Problem with Time Windows, Part II: Metaheuristics," *Transportation Science*, vol. 39, no. 1, pp. 119–139, Feb. 2005.
[14] C. Groër, B. Golden, and E. Wasil, "A library of local search heuristics for the vehicle routing problem," *Mathematical Programming Computation*, vol. 2, no. 2, pp. 79–101, Jun. 2010.
[15] P. Shaw, "A new local search algorithm providing high quality solutions to vehicle routing problems," University of Strathclyde, Glasgow, Technical Report, Jul. 1997.
[16] H. Li and A. Lim, "A Metaheuristic for the Pickup and Delivery Problem with Time Windows," *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001*, vol. 12, no. 2, pp. 160–167, Dec. 2001.
[17] G. Laporte, R. Musmanno, and F. Vocaturo, "An Adaptive Large Neighbourhood Search Heuristic for the Capacitated Arc-Routing Problem with Stochastic Demands," *Transportation Science*, vol. 44, no. 1, pp. 125–135, Oct. 2009.
[18] K. Lund, O. Madsen, and J. Rygaard, "Vehicle Routing Problems with Varying Degrees of Dynamism," Technical University of Denmark, Technical Report, Jan. 1996.
[19] J. Van Lochem, "Improving Dynamic Route Optimisation by making use of Historical Data," Master's thesis, TU Delft, Apr. 2019.