The Effect of Alternative Battery Management Strategies for the Dynamic Dial-A-Ride Problem using an Electric, Autonomous Vehicle Fleet

Master's Thesis Operations Research and Quantitative Logistics
Erasmus School of Economics

Sean Gilbert
413885

ERASMUS UNIVERSITEIT ROTTERDAM

Supervisor: Dr. P.C. Bouman
Second assessor: Y.N. Hoogendoorn

1st May 2021

Abstract

In recent years, increased traffic in cities combined with easily accessible state-of-the-art online mobile technology has led to a rise in demand for on-demand ride-sharing services. Simultaneously, developments in vehicle technology have made electric and autonomous vehicles an increasingly realistic option for shared-ride networks, while academic research continues to incorporate many of the constraints and possibilities that new transportation modes present. This research investigates the effect of various vehicle recharging strategies in the context of a dynamic Dial-A-Ride Problem (DARP). It is found that a charging strategy which strikes a balance between maintaining reserves of battery when charging, while keeping flexibility in the vehicle schedules, is well suited when trying to maximise the quality of service. Moreover, it is found that the fraction of customer transportation requests known before the start of the day, as well as the amount of advance warning for requests that appear throughout the day, both have a clear impact on service quality and operational cost.

The submission of this thesis marks the closing stages of my education at Erasmus University Rotterdam, a period which I have thoroughly enjoyed and learned a great deal from. I would like to say a big thank you to my family and friends, whose tremendous support and encouragement along the way has been invaluable. I would further like to express my gratitude to my supervisor Dr. Paul Bouman, who has been continually supportive and encouraging throughout the thesis writing process, as well as being constructive in his feedback and open to discuss different ideas. Finally, I would like to thank my second reader Ymro Hoogendoorn, whose useful input at such short notice I believe has helped improve the structure and readability of this thesis.

Contents

1 Introduction

The Dial-A-Ride Problem (DARP) is characterised by the dispatching of one or more vehicles in order to serve a series of customer transportation requests between pre-specified origin and destination locations. These customer requests, which are often accompanied by a time window in which they must occur, constitute a form of shared-ride service as customers are typically allowed to be transported simultaneously in the same vehicle. Although the DARP is best known as a branch of Vehicle Routing Problems (VRPs) in academic literature, its existence is highly motivated by real-life applications. The most common real-world application of the problem has been in studying pick-up and delivery transit services for the disabled or elderly population segments, who typically cannot access traditional public transportation networks. Such problems often entail door-to-door service, as well as a later return trip from the destination to the pick-up point. Moreover, in recent years, increased congestion in urban areas coupled with improved mobile applications have led to a rise in demand for traditional ride-sharing as well as on-demand services (Shaheen and Cohen, 2019). At the same time, advancements in vehicle technology have made automated vehicles (AVs) an increasingly tangible possibility for shared-ride networks. According to Greenblatt and Shaheen (2015), AVs will become an accepted mode of transport by the year 2030 and dominate shared-ride technology in the years to follow. Since automated vehicles have the ability to improve the mobility of both the elderly and those with travel-restricted medical conditions, their integration in a shared-ride network offers improved service for population segments that are currently underserved, as well as reducing reliance on personal vehicles in urban areas (Harper et al., 2016).

Not only are advancements in vehicle and mobile technology going to manifest themselves in the form of increased demand for shared-ride services and AVs, the current surge in electric vehicle demand will alter the landscape of transportation services too. In the Netherlands, for example, the use of EVs has increased tenfold in the last six years, with a projected one million EVs to be found on Dutch roads within the coming five years (Swagerman, 2019). This trend is not specific to one country either: according to the International Energy Agency (IEA), global electric vehicle sales have expanded to such a large extent that over 1% of new registered vehicles are electric in more than 20 countries worldwide. Furthermore, whilst the majority of this increase is attributed to privately owned cars and charging points, public transit systems using electric buses have been trialled in countries such as India, Finland, Chile and China (IEA, 2020). Due to large-scale international targets concerning fuel economy and CO_2 emissions, use of EVs is increasingly subsidised and incentivised at national levels to work towards a long-term vision of cleaner energy use.

In order to model the changing requirements and objectives of ride-sharing services, scientific literature in vehicle routing, scheduling and management has evolved to increasingly incorporate many of the restrictions imposed, and possibilities enabled by, new technologies. This has, in many fields, led to the emergence of new problems and, in turn, new solution approaches for tackling them. This research seeks to contribute to this endeavour. To this end, the research proposes a solution methodology for handling many of the real-life constraints presented by the necessity for environmentally friendly, alternative fuel sources in a shared-ride system. Specifically, the aim of this research is to construct a framework for handling both offline and online customer transportation requests using electric autonomous vehicles, and to investigate the effect of various vehicle recharging strategies on the performance of the fleet of vehicles under this framework. The report proceeds as follows. Section 2 introduces the problem at hand, while Section 3 brings to light the existing body of relevant literature on the subject. Section 4 then outlines the methodology used for solving the problem and Section 5 presents the data used to generate the necessary problem instances. Finally, Section 6 presents the results of the investigation before conclusions are drawn in Section 7 and suggestions for further research are highlighted in Section 8.

2 Problem Description

In the following chapter, we expound the framework in which the problem is investigated. The relevant notation and constraints are introduced, and the way in which dynamic customer requests are treated is explained. Within the problem framework, the direction of investigation and relevant research questions are outlined.

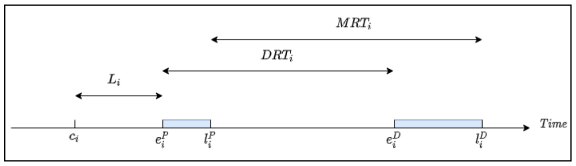
2.1 Static e-ADARP

The Electric Autonomous Dial-A-Ride Problem (e-ADARP) is a variant of the classical many-to-many dial-a-ride problem, in which n customers must be transported between their pre-specified pickup and drop-off location. The variant, which was introduced in Bongiovanni et al. (2019), incorporates a fleet of electric, autonomous vehicles. This gives rise to additional constraints related to battery consumption, but simultaneously offers greater flexibility than other variants. Specifically, since autonomous vehicles do not contain a driver, vehicle service is not restricted by the maximal duration of a driver's shift. The absence of a driver further allows the possibility for vehicles to freely select between depots at which to end their service. However, due to the limited battery of the vehicles, intermediate stops at charging stations may be necessary.

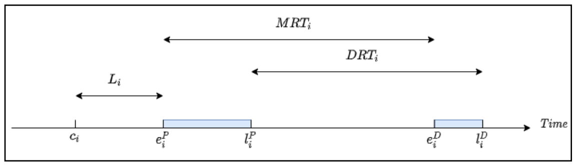
The objective of the e-ADARP is to construct a list of vehicle routes that each begin at some predetermined origin depot, service a subset of customer requests and end at a particular destination depot, while optionally making use of recharging stations to maintain sufficient battery levels. This is done with a view to minimising a particular objective function, whilst respecting the various constraints imposed on the vehicle routes.

The network used to represent the e-ADARP is a directed graph $G = (V, A)$, wherein the nodes of the graph represent all possible stops in the network at which a set of heterogeneous vehicles ($k \in K$) can stop. As well the customer pickup (P) and delivery (D) locations, the set of vehicle stops (V) comprises, for each vehicle, an origin depot station (O^k), a set of charging facilities (S) and a set of optional destination depots (F). The edges A^k along each vehicle's graph capture the distances c_{ij} between stops, from which travel times t_{ij}^k and battery consumption β_{ij}^k along the same edge can be inferred. Furthermore, each customer has a maximum ride time u_i and both pickup and drop-off requests have a service duration s_i to represent the time needed for loading and unloading passengers. In addition, customer requests may entail multiple passengers being carried together. Therefore, a change in load q_i (positive for pickups, negative for drop-offs) represents the number of customers requiring service at a given vehicle stop. The set of m vehicles used in the network are assumed to be heterogeneous. They may differ in their (battery) capacity, speed and initial battery levels. Charging stations (S) may differ in their rate of battery replenishment, i.e. the rate at which vehicles' battery levels are restored upon visiting the station. Finally, it is required that all vehicles must preserve a minimum battery level (r) when arriving at any of the destination depot stations.

Finally, each pickup and drop-off is accompanied by a time window $[e_i, l_i]$ in which service must commence. Each request is characterised either as an inbound or outbound request. For inbound requests, only the pickup time window is pre-specified, while only the drop-off time window is given for outbound requests. The construction of the time window for the corresponding pickup (drop-off) of outbound (inbound) requests is done based on the direct travel time (DRT) between the pickup and drop-off and maximum ride time (MRT) u_i of the customer. This is depicted visually in Figure 1 below. As a result, inbound requests typically have a narrower time window for pickup than for drop-off, while the opposite holds for outbound requests.



(a) Inbound customer request



(b) Outbound customer request

Figure 1: Time window construction based on the maximum ride time (MRT), direct ride time (DRT) for inbound and outbound customer requests

2.2 Dynamic Considerations

In contrast to the static e-ADARP in which all service requests are known in advance, some customer requests arrive during the service horizon in the dynamic version of the problem. The ratio of dynamic service requests requests to total requests is referred to as the *degree of dynamism* of the system, and its value may affect the performance of different recharging procedures. A degree of dynamism of zero boils down to a static problem, while a value of one represents a fully dynamic system with no advance requests.

Dynamic requests are additionally characterised by a *lead time*, which measures how far in advance a service request is booked. This is also illustrated in Figure 1 above. The lead time L_i of a request is calculated as the time between the earliest pickup time e_i^P and the call time c_i of the request (Luo, 2006). Short leads times may hinder the ability of vehicles to accommodate new requests as their current schedules may not facilitate the insertion of a new request at such short notice. On the other hand, longer lead times are likely to offer greater flexibility in determining the optimal vehicle schedules for accommodating incoming requests.

2.3 Research Questions

As modern ride-sharing and on-demand services need to be capable of handling an increasing number of dynamic customer requests, finding optimal solutions to real-life problems is becoming increasingly difficult. Accordingly, greater emphasis is being placed on quick and effective heuristics for handling a large number of customer requests. Furthermore, improving online technology is increasingly allowing for the real-time modification of existing vehicle routes as new requests emerge. To embed these modern requirements into the solution approach, this research aims to answer the following questions:

Q1: *Does a hybrid VNS/Tabu Search (meta)heuristic approach to e-ADARP perform competitively against exact solution methods?*

Q2: *Does an online insertion procedure tailored to the specific constraints of the e-ADARP effectively facilitate the insertion of dynamic customer requests?*

- ***Q2a:*** *How do different battery management strategies improve the system's ability to accept incoming customers?*
- ***Q2b:*** *Under which conditions do different battery management strategies allow new customer requests to be accepted?*

3 Literature Review

The DARP differs from a traditional Pick-up and Delivery Problem (PDP) in that it strictly concerns the transportation of passengers, and thus entails service-related constraints or objectives. It is a generalisation of the PDP and the traditional Vehicle Routing Problem (VRP), both of which have been shown to be $\mathcal{NP}$-hard problems. As a consequence, the majority of studies on Dial-A-Ride problems have focused on heuristic solution approaches for handling a large number of transportation requests. Nevertheless, a brief overview of exact solution approaches will be provided, followed by a more extensive review of (meta)heuristic solution methods, focusing primarily on the multi-vehicle DARP. Furthermore, the integration of insertion algorithms in a real-time, or online setting, will be discussed, followed by brief discussion of the techniques used in electric vehicle routing problems to incorporate battery management into routing decisions.

3.1 Exact Approaches

The primary benefit of exact solution approaches is that when solved, they obtain the optimal solution to the problem instance at hand. As compared to heuristic approaches, the drawback of exact approaches is that they are computationally expensive, and often intractable for large problem sizes. Despite this, a range of exact approaches have been utilized for different variants of the DARP.

Firstly, Psaraftis (1980) provided an exact solution approach for the single-vehicle, many-to-many DARP, by means of dynamic programming (DP). The algorithm builds on the classic solution approach for the travelling salesman problem (TSP) by exploiting the particular structure of the problem, and includes an objective function which incorporates both the total travel time and a user satisfaction metric. This algorithm was later adapted (Psaraftis, 1983) to deal with customer time windows for pickup and delivery, by employing a forward, rather than backward, recursion scheme, in order to minimise the total time needed to serve all customers. However, due to the exponential computational complexity of $\mathcal{O}(N^2 3^N)$, this approach was feasible for instances containing at most $N = 9$ customers. Later, Desrosiers et al. (1986) also formulated a forward DP labelling algorithm for the single-vehicle, many-to-many DARP, but minimized the distance travelled instead of total service time. By eliminating infeasible labels on account of infeasible time windows, insufficient vehicle capacity and precedence constraints, the authors were able to optimally solve instances up to size $N = 40$.

Cordeau (2006) proposed a branch-and-cut algorithm for the static version of the many-to-many, multi-vehicle DARP, with the objective of minimizing total vehicle distance travelled. A mixed-integer program (MIP) is formulated for the problem, and a series of novel valid inequalities inspired by other classes of VRPs were incorporated. These inequalities strengthen the LP relaxation of the problem. Within a time limit of two minutes, the branch-and-cut algorithm was able to find the optimal solution for problem instances up to $N = 32$ customers. Ropke et al. (2007) presented a tighter, two-index formulation for the DARP which incorporated previously formulated constraints as well as new families of valid inequalities. As compared to a three-index formulation, a two-index formulation provides the benefit of demanding fewer variables and constraints, but does not allow for the inclusion of vehicle-specific constraints. The research described above formed a foundation for a vast range of subsequent research into the DARP and its variants. Liu et al. (2015) integrated many of the valid inequalities proposed in Cordeau (2006) and in Ropke et al. (2007) in a DARP setting that accounts for several additional realistic constraints. Namely constraints relating to multiple trips, heterogeneous vehicles, multiple request types, configurable vehicle capacity and manpower planning restrictions were added to the problem. Moreover, Bongiovanni et al. (2019) presented both a two- and three-index MIP formulation for the electric, autonomous DARP (e-ADARP). The authors further formulated new problem-specific valid inequalities relating to battery capacity and recharging stations before applying a branch-and-cut algorithm.

Finally, branch-and-price has also been used as an exact solution approach in DARP literature. This technique typically involves solving a shortest path problem in order to price vehicle routes to be added to a restricted master problem (RMP). Examples of this include Garaix et al. (2010) who utilized dynamic programming in order to generate columns, and Feng et al. (2014), where an exact constraint programming approach was proposed for the sub-problem. However, the drawback to a branch-and-price methodology is that the pricing problem is often NP-hard as well. Hence, realistically sized instances often necessitate the use of inexact approaches to generate columns for the RMP. For example, Hu and Chang (2015) employed a large neighbourhood search (LNS) and Parragh et al. (2015) used a variable neighbourhood search (VNS) in a column generation procedure for the DARP.

3.2 Heuristic Approaches

For both the static and dynamic DARP, a wide range of heuristic solution methods have been proposed, with the majority of research being focused on the static variant. Since several requests in a dynamic DARP are typically known in advance, many of the procedures applied to the static DARP can be extended to the dynamic counterpart. In practice, this is either done by resolving a static DARP whenever a new customer request appears, or by applying efficient insertion-based methods to insert new requests into existing vehicle routes. The following section will review several insertion heuristics, as well as metaheuristics, which have been proposed for the DARP.

3.2.1 Constructive Heuristics

Early research on heuristics for the DARP focused primarily on constructive procedures in order to obtain feasible vehicle routes, rather than local search procedures for improving existing routes. One of the first procedures for the dynamic DARP was proposed by Daganzo (1978), in which three simple insertion algorithms were introduced. The first iteratively visits the nearest customer next, the second does same but alternates between pickups and deliveries and the third only permits delivery once a given number of customers have been picked up. However, it was the algorithm proposed by Jaw et al. (1986) for the static variant of the multi-vehicle, many-to-many DARP that formed a basis for many subsequent approaches in DARP literature. Jaw et al. (1986) proposed a sequential insertion heuristic, in which user time windows are constructed based on their desired pickup or drop-off times, as well as the direct ride time between origin and destination. Their algorithm processes customers one by one, according to their earliest pickup time. For each customer, the algorithm identifies both the vehicle and position within their route to which the customer can feasibly be inserted at the lowest additional cost. If no such feasible insertion is identified, the customer's request is rejected. As in many other multi-objective DARP formulations, their objective function accounts for both operational cost and customer service levels.

Utilising the same problem framework as in Jaw et al. (1986), Toth and Vigo (1997) proposed a constructive heuristic to minimise the number of required vehicles for the static, multi-vehicle DARP. A parallel insertion algorithm minimizes total routing and inconvenience costs by first attempting to assign unserved customer trips into existing routes, and using simple inter- and intra-route moves to improve upon their solution quality. Their solution approach is applied to a real-life instance for transporting handicapped users in Bologna, Italy.

For the dynamic version of the DARP with time windows, Coslovich et al. (2006) developed a two-phase insertion heuristic for dealing with unexpected customers. In their problem setting, unexpected customers with a desired drop-off location and time window may request transportation during a vehicle stop. Drivers must then determine, in real-time, whether the request can be accommodated. Their heuristic distinguishes between two stages; namely, offline and online. During the offline stage between vehicle stops, a simple 2-opt solution neighbourhood, which swaps the position of two customers within a route and reverses the visiting order of all customers between them, is explored in order to improve upon the current solution. When an unexpected request appears during a vehicle stop, the online algorithm is initiated to quickly find the feasible insertion of the new request into an existing vehicle route that minimizes user inconvenience. Furthermore, Madsen et al. (1995) proposed an insertion heuristic similar to that of Jaw et al. (1986), called REBUS, for the dynamic DARP with time windows. They applied their algorithm to the scheduling of 300 disabled and elderly customers in Copenhagen. In contrast to Jaw et al. (1986), whose approach sorts unserved customers according to their earliest pickup time, the approach of Madsen et al. (1995) sorts unassigned customers on the basis of how easily the customer can be inserted into existing routes. To assign priority of one job over another, the authors consider the narrowness of the customer time window, the maximal excess ride time permitted, as well as the ability of vehicles to handle the customer capacity demands.

One fundamental drawback to the algorithm proposed by Jaw et al. (1986) as well as many other solution approaches proposed since, is the propensity to result in myopic behaviour. Both in a static and dynamic setting, the optimal insertion of a customer at one point in time may lead to significantly worse insertions, or infeasible solutions, once subsequent insertions are considered. This phenomenon becomes increasingly likely as time windows tighten. To circumvent this problem, Diana and Dessouky (2004) developed a regret-insertion solution approach. Their algorithm inserts customers according to their regret cost, defined as "the potential price that could be paid if a given request were not inserted" (Diana and Dessouky, 2004). For each unassigned request, the algorithm computes the incremental cost (increase in objective value) from inserting the request in a given route. For a given request, regret is then calculated by summing the differences between a route's incremental cost and the minimum incremental cost across all routes. The request with the largest regret is immediately carried out by inserting it in the route which minimises incremental cost.

3.2.2 Metaheuristics

In recent years, the use of metaheuristic approaches to solve the Dial-A-Ride Problem have risen in popularity, culminating in a range of solutions methods being devised and employed for the DARP and its variants. In particular, tabu search, (adaptive) large neighbourhood search and variable neighbourhood search have received considerable attention due to their effectiveness as solution methods. An overview of these metaheuristic approaches in DARP literature will be outlined below.

3.2.2.1 Tabu Search

Tabu Search (TS) is a popular metaheuristic that has been applied to a wide range of VRPs. The premise of the method is to maintain a list of solutions, or attributes of solutions, that are considered tabu; in other words, are prohibited from being selected. Typically, solutions in the tabu list remain there for a certain number of iterations, in order to diversify the regions of the solution space explored and thereby escape local optima.

Cordeau and Laporte (2003) was the first research to propose a tabu search heuristic for static version of the multi-vehicle DARP. In their approach, they utilize a simple random construction heuristic to initialize a set of vehicle routes, which may be infeasible in terms of vehicle load and duration, as well as customer ride time and time windows. Solutions are evaluated in terms of a cost function, which accounts for total vehicle routing costs, as well as penalties for total violated load, duration, time window and ride-time constraints. The parameters relating to each type of constraint are dynamically adjusted, increasing if a given solution violates the constraint and decreasing otherwise. To stimulate diversification, the algorithm allows intermediate solutions found to be infeasible and also penalizes solution attributes based on the frequency of their appearance. In their approach, the neighbourhood of a solution is characterized by all simple insertions of the request of some customer i from its current route k into the route of some other vehicle $k' \neq k$. The success of this solution approach has contributed to its continued application in Dial-A-Ride Problems. For instance, Guerriero et al. (2013) embedded a tabu search procedure based on that of

Cordeau and Laporte (2003) in a greedy randomized adaptive search (GRASP) heuristic, and Paquette et al. (2013) integrated the same approach in their reference point method for multi-criteria objective functions.

3.2.2.2 (Adaptive) Large Neighbourhood Search

Large Neighbourhood Search (LNS) represents another popular metaheuristic in many classes of vehicle routing problem literature, including the DARP. While some local search methods either consider only a small fraction of the solution space or investigate neighbourhoods with many infeasible solutions, LNS differs in its approach by repeatedly destroying and repairing a current solution. Adaptive LNS, or ALNS, is characterized by the presence of multiple destroy and repair methods, whose frequency of use are dependent on weights which are dynamically altered throughout the search. Typically, the weights are adjusted based on factors such as the success of the method in previous iterations, run time of the method or the state of the search itself. In all papers discussed below that use an ALNS framework, acceptance of new solutions is done by means of a simulated annealing criterion. Consequently, worse solutions are sometimes accepted (to escape local minima), but with decreasing probability as the search evolves.

In their article on the Pickup and Delivery Problem with Time Windows (PDPTW), Ropke and Pisinger (2006) proposed an ALNS approach, and observed significant performance improvement through the use of multiple destroy and repair methods, as opposed to a single sub-heuristic for this purpose. The destroy methods employed are the Shaw removal, random removal and regret removal methods, with greedy and regret repair methods being utilized to repair the solutions again. Both the repair and destroy methods are selected using a classic roulette wheel principle, while weight adjustment of the methods is determined by past performance and prior frequency of use. The ALNS procedure introduced by Ropke and Pisinger (2006) has since been tailored to the DARP. For example, Braekers and Kovacs (2016) solved the driver consistent DARP (in which customers receiving repeated service are served by a maximum number of different drivers) by means of an ALNS procedure inspired by Ropke and Pisinger (2006). Destroy operators in this research are taken directly from previous literature, while a tailored repair operator based on regret is introduced. Similarly, for the standard DARP, Gschwind and Drexl (2019) further incorporated a destroy method in their solution approach for badly positioned sequences of requests, and another based on sequences where the vehicles are empty. They additionally achieve intensification by using a dynamic programming (DP) approach to find the best solution in a specific neighbourhood of promising solutions, and solve a set covering problem to determine the optimal selection of generated vehicle routes.

3.2.2.3 Variable Neighbourhood Search

Variable Neighbourhood Search (VNS), as a local search procedure, has been applied to the Dial-A-Ride Problem by several different authors in the past decade. The search procedure uses different neighbourhoods in the perturbation of solutions, and typically moves to larger neighbourhoods if random moves in smaller neighbourhoods do not yield improved solutions.

The implementation of VNS in a DARP framework gathered momentum through Parragh et al. (2009), who applied a variable search neighbourhood to the multi-objective DARP. The authors used four classes of neighbourhoods, namely a move neighbourhood (moving a request to another route), a swap neighbourhood (swapping two requests between routes), an ejection chain neighbourhood (sequentially moving randomly sized sequences of customers to other routes) as well as a zero-split neighbourhood (moving customer from sequences between which vehicle load is empty into other routes). VNS was later applied by the same authors to the single objective DARP (Parragh et al., 2010). Since then the approach has been modified to solve other problem variants, such as in Detti et al. (2017), who embedded a VNS within their solution framework to tackle a multi-depot, heterogeneous fleet DARP variant. Moreover, in addition to the classic swap and chain neighbourhoods, Muelas et al. (2013) introduced four novel greedy neighbourhoods, as well as an improved adaption of the zero-split neighbourhood, to solve a DARP in a real-life instance from San Francisco.

3.3 Hybrid Approaches

An increasingly popular trend in DARP literature is the partnership of multiple standalone solution methods. This may occur by integrating a computationally efficient heuristic with the use of mathematical programming techniques. For instance, Hu and Chang (2015) and Parragh et al. (2015) employed heuristics in a branch-and-price framework, while Gschwind and Drexl (2019) utilized a DP algorithm in the neighbourhood of potentially fruitful solutions. Alternatively, Berbeglia et al. (2012) applied a constraint programming approach to check for the feasibility of new insertions in a dynamic DARP environment whilst using a tabu search procedure to determine vehicle routes of accepted customers.

Similarly, the different strengths of different metaheuristics may be utilized by applying them in conjunction with one another. Metaheuristics may be applied in a sequential manner (such as Toth and Vigo (1997) who applied a tabu thresholding approach to improve on their fast parallel insertion procedure) or may involve embedding one within the other (for example, Guerriero et al. (2013) integrated their tabu search algorithm in a greedy randomised adaptive search procedure).

3.4 Online Requests

The integration of dynamic requests in Dial-A-Ride Problems poses additional challenges in terms of constructing feasible vehicle routes. In certain settings, new requests must be incorporated in existing schedules in real time, whereas other settings allow a longer acceptance time for the insertion of new requests to be facilitated without the possibility of the request being cancelled. Varying problem characteristics of the instances presented in academic literature have given rise to alternative solution approaches for dealing with incoming requests during the service horizon. Broadly speaking, however, the different solution methods for accommodating these requests in a DARP setting can be categorised as one of the following techniques: insertion algorithms, reinsertion algorithms or (meta)heuristic approaches. The underlying idea behind each of these approaches is outlined below and illustrated by means of one or more relevant examples from academic literature.

3.4.1 Insertion Algorithms

The underlying idea behind insertion algorithms is to insert new customer requests into existing vehicle routes in a quick and efficient manner. Madsen et al. (1995) adapted the simple sequential insertion procedure presented by Jaw et al. (1986) for advance requests to the dynamic variant, and opted to rank new requests based on the difficulty of insertion, such that more difficult requests (in terms of time window tightness and vehicle capacity) are scheduled before relatively easier jobs. Furthermore, Mitrovic-Minic et al. (2004) considered both a short- and long-term decision horizon for the dynamic variant of the DARP's sister problem, the PDPTW. In their research, the authors identified the need for dynamic vehicle scheduling procedures to preserve as much available time slack for potential future requests while minimizing travel distance in the short term. For the short term, the authors employed a simple cheapest insertion procedure for the new requests, followed by potentially reinserting requests whose pickup had not yet occurred using a cheapest reinsertion approach. For the long-term horizon, however, a tabu search heuristic was used to improve on solutions generated in the short-term. Since then, more sophisticated insertion procedures have been developed for dealing with online requests. For example, Maalouf et al. (2014) use a fuzzy logic algorithm for inserting dynamic requests in vehicle routes, focusing on minimising customer travel time as well as vehicle travel distance when selecting the vehicle to incorporate an online request. Furthermore, Wong et al. (2014) consider the effect of different vehicle scheduling policies on travel distance and customer acceptance rates in a dynamic setting when applying a cheapest insertion heuristic.

3.4.2 Reinsertion Algorithms

Whereas insertion algorithms focus on quickly incorporating new customer requests in existing vehicle routes, a situation may arise in which the current route configuration does not allow the new request to be inserted. While typical insertion procedures then dictate that the new request is rejected, reinsertion algorithms seek to rearrange the currently scheduled requests in order to accommodate the new request. Some procedures, such as the approach presented by Mitrovic-Minic et al. (2004), further exploit the fact that currently scheduled requests may benefit from being rescheduled following the insertion of a new customer request, even if the new request has been accepted (improvement procedure). Nevertheless, academic literature on reinsertion techniques for the online DARP remains relatively scarce. Luo and Schonfeld (2011) proposed a rejected-reinsertion approach for the problem, adapted from their procedure for the static DARP. In case a new request is rejected, the cost of removing every scheduled request from its current route, inserting the new request in the current route and re-inserting a previously scheduled request in another route, is calculated. The lowest cost reinsertion is subsequently selected. Additionally, the research periodically incorporates an improvement procedure using simple move and exchange operators, and further illustrates the benefit of deferring the scheduling of new requests in a dynamic setting. More recently, Vallée et al. (2020) proposed three novel reinsertion approaches for the online DARP. The first reinsertion approach incorporates characteristics of a large neighbourhood search (LNS), in the sense that multiple scheduled requests are removed from their respective routes and reinserted together with a new request. The second approach, known as Graph Heuristic (GH), uses the concept of ejection chains to model the reinsertion problem as a constrained shortest path problem on a directed graph before applying the well-known Bellman-Ford algorithm. Finally, the third approach comprises iteratively applying the Graph Heuristic with a different starting solution in the event that no feasible solution is found with the GH. On the whole, the authors find that the first approach performs slightly better than the graph-based approaches during their computational experiments.

3.4.3 Post-optimisation Algorithms

In an online DARP environment, the majority of proposed solution approaches seek to utilise the time between new customer arrivals, while the system is idle, in order to improve the quality of solutions obtained. To this end, a range of metaheuristics have been employed in an event-based simulation environment to run between the occurrence of events. The first example of this can be found in the work of Attanasio et al. (2004), where parallel processing is used in combination with the tabu search algorithm developed by Cordeau and Laporte (2003). This occurs after a simple insertion procedure determines whether a new request is accepted or rejected. Beaudry et al. (2010) also used tabu search in their case study of a dynamic DARP environment for a German hospital. Coslovich et al. (2006) devoted the latent system time constructing and exploring increasingly large neighbourhoods of the current solution, thereby yielding better solutions and allowing the insertion of a new request into many potential vehicle schedules.

More recently, VNS has been applied as an improvement metaheuristic procedure to the practical real-life example of the Austrian Red Cross by Schilde et al. (2011). Here, the stochastic possibility of a return trip is considered in the routing and scheduling decisions, while a cheapest insertion rule for new customers is used. Alternatively, Santos and Xavier (2015) split the service horizon into multiple periods and use a greedy randomized adaptive search procedure (GRASP) within each period to construct vehicle routes. Finally, Lois and Ziliaskopoulos (2017) defined an "optimisation cycle" as the complete exploration of the single request reassignments of a given solution. Similarly to Diana and Dessouky (2004), the authors perform an insertion procedure based on regret, but further attempt to improve on this solution by performing an optimisation cycle. By allocating a monetary value to accepted and rejected customers, they illustrate that, in an online setting, there exists a trade-off between solution quality and computation time which needs to be accounted for when considering new customer requests. The balance between the two in practice depends on many factors, such as the size of the problem and frequency of new requests.

3.5 Battery Management

Since the use of electric vehicles in VRPs is a relatively recent emergence, research to date on the role of battery management in related problems remains relatively scarce. Schneider et al. (2014) modified the VRP with time windows (VRPTW) to include electric vehicles and recharging stations (E-VRPTW) and formulated an exact formulation for small to medium sized problem instances. They also introduced a hybrid metaheuristic approach for larger instances, but in both cases assumed that vehicles must fully recharge at recharging stations. For the same problem, Keskin and Çatay (2016) proposed an exact approach for small instances, which allows for partial vehicle recharge. Recognising the fact that when all requests are known *a priori*, an optimal solution exists in which all vehicles return to the depot with the minimum prescribed battery level, the authors were able to improve on the solutions obtained by Schneider et al. (2014) in certain cases. In their ALNS procedure for large instances, they further find that freedom to select the amount of time spent at recharging stations outperforms recharging strategies in which the amount of charge acquired is predetermined. Nevertheless, very little research on battery charging strategy has been performed in dynamic vehicle routing problems. In such situations, charging strategy is a crucial yet non-trivial decision to make, as future requests may entail using more battery, and so factoring in battery reserve may prevent complete depletion. Nevertheless, such a strategy may involve longer or even additional trips to charging stations, which in turn may impact vehicles' abilities to accommodate new requests, particularly in the event of tight time windows.

3.6 Research Hypotheses

On the basis of the reviewed literature, predictions can be made regarding the outcome of the research. Firstly, it is anticipated that a metaheuristic approach for the e-ADARP will be able to provide good, yet potentially sub-optimal, solutions to the various static instances, in a short period of time. Furthermore, it is anticipated that as the problem instances impose tighter battery constraints, the difference in solution quality between exact approaches and a metaheuristic approach will increase. In addition to routing and scheduling decisions, the battery restrictions introduce a third dimension to the decision making process. As a consequence, it is expected that the difference between heuristic and exact approaches will increase.

In an online setting, the area of interest is whether vehicle charging strategies will affect the system's ability to accept new customer requests. Each of the three strategies are explained in detail in Section 4.5.1. It is hypothesised that strategies which spend more time at charging stations but plan fewer visits will prove less flexible, and thus less effective at serving new customer requests as they appear. It is expected that these differences in flexibility will translate to larger differences in service quality when request lead times are short, or the degree of dynamism of the system is high. Therefore, the research hypotheses can be stated as answers to the research questions in Section 2.3, as follows:

H1: *The difference in performance between exact and heuristic solution methods for the static e-ADARP increases with the required level of battery r at the destination depot.*

H2a: *A full recharging strategy allows fewer new requests to be served than a conservative or minimal recharging strategy.*

H2b: *The ability of a full recharging strategy to serve new requests is more sensitive than a conservative or minimal recharging strategy to an increase in the degree of dynamism or a decrease in lead times.*

4 Solution Methodology

In this section, the framework developed for serving both advance and dynamic customer requests is outlined. Before handling dynamic requests as they appear during the planning horizon, vehicle routes and schedules are determined for advance requests. The process of obtaining routes and schedules for the advance requests can be decomposed into two distinct stages. Firstly, vehicle routes and schedules obtained by applying a construction heuristic are used as a starting point for the second stage. In the second stage, a metaheuristic approach is applied in order to improve upon the solution obtained through the construction heuristic and determine routes and schedules for all advance requests. Subsequently, as each dynamic requests appears during the planning horizon, an attempt is made to insert the request into the existing solution by means of an immediate online insertion heuristic. Finally, if the customer is successfully added to the vehicle, a post-improvement procedure is applied while no new requests arrive. This post-improvement procedure assigns a pre-specified amount of time to improving upon the newly updated routes and schedules, and is only interrupted by the arrival of a new request. An overview of the various components of the solution approach can be seen in Figure 2 below.

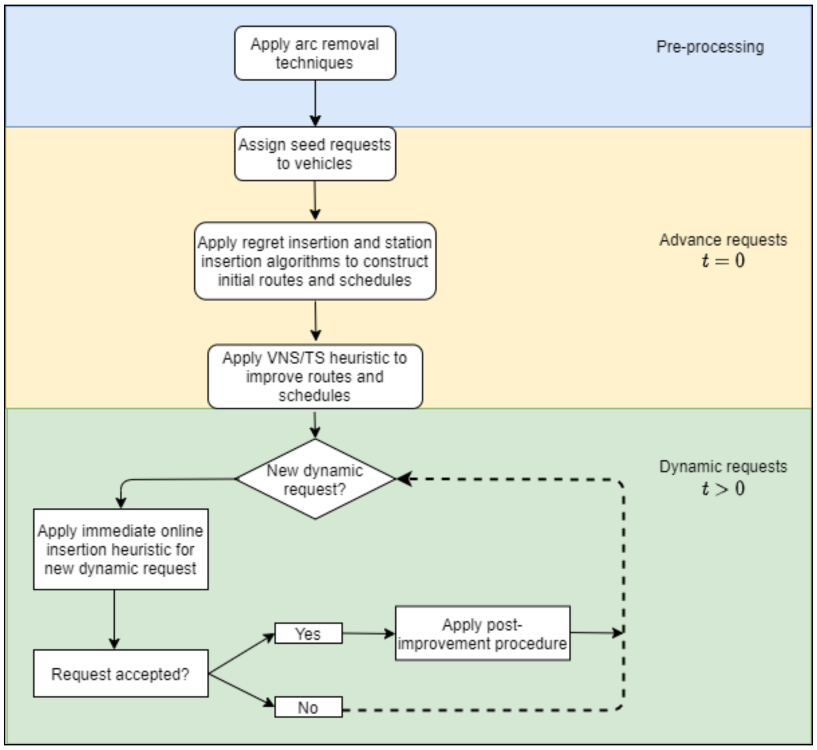

Apply arc removal techniques
Pre-processing
Assign seed requests to vehicles
Apply regret insertion and station insertion algorithms to construct initial routes and schedules
Advance requests $t = 0$
Apply VNS/TS heuristic to improve routes and schedules
New dynamic request?
Dynamic requests $t > 0$
Apply immediate online insertion heuristic for new dynamic request
Yes
Apply post-improvement procedure
Request accepted?
No

Figure 2: Algorithmic overview of the various steps in the solution framework

Each stage of the solution framework is explained in the remainder of Section 4, which is organised as follows. Firstly, the cost function used to evaluate the quality of the obtained vehicle routes and schedules is outlined in Section 4.1, and Section 4.2 describes the pre-processing techniques for improving computation times. Thereafter, Section 4.3 explains the handling of advance requests, by outlining the procedure for constructing an initial starting solution (Section 4.3.1) as well as the metaheuristic approach to improve upon this starting solution (Section 4.3.2). Finally, Section 4.4 explains how online requests are handled in the solution framework, and Section 4.5 describes the charging and scheduling decisions made by vehicles at their planned stops.

4.1 Cost Function

A solution $S = \{r_k, k = 1, ..., m\}$ is defined as a set of m vehicle routes, where each route $r_k = \langle v_0, v_1, ..., v_n, v_{n+1}\rangle$ is represented by the sequence of visited vertices. Given that vehicles start and end at a depot, v_0 and v_{n+1} necessarily correspond to depot stations in the network. A solution S is evaluated in terms of a cost function $f(S)$, which measures the total travel time $c(S)$ of all vehicle routes, and penalizes violations of the various constraints:

$$f(S) = c(S) + \alpha \times \mathrm{CAP}(S) + \rho \times \mathrm{TW}(S) + \gamma \times \mathrm{BAT}(S) + \tau \times \mathrm{RT}(S) \tag{1}$$

In equation (1) the terms $\mathrm{CAP}(S), \mathrm{TW}(S), \mathrm{BAT}(S), \mathrm{RT}(S)$ denote the capacity, time window, battery and user ride time constraint violations respectively. The parameters α, ρ, γ, τ are dynamically updated throughout the search procedure, increasing or decreasing by a factor $1+\omega$ (with $\omega > 0$) in consecutive iterations. If the current solution violates a given constraint, the penalty parameter is multiplied by the factor $1 + \omega$. If the current solution is feasible with respect to the constraint, the penalty parameter is divided by the same factor. This is done in order to steer the search towards solutions in which the constraint is violated less heavily, or not at all.

The user ride-time violation of a given route r_k is calculated as the sum of the ride time violations at each vertex, and the ride-time violation $\mathrm{RT}(S)$ of a solution is then computed by summing the ride-time violations of all vehicle routes:

$$\mathrm{RT}(r_k) = \sum_{v \in D} \max\Big\{\mathrm{RT}_v - u_v, 0\Big\}, \qquad \mathrm{RT}(S) = \sum_{k=1}^{m} \mathrm{RT}(r_k)$$

However, the calculation of battery, capacity and time-window violations is performed slightly differently. To model time window violations, an intuitive approach would be to sum the time window violations at each customer along a route. This approach may, in practice, lead to the penalisation of good sequences of vertices that occur after a violation of a customer time window, due to the resulting delay of the entire route. To circumvent this problem, Schneider et al. (2013) present an approach for modelling subsequent time window violations after a violation has occurred at customer i. For subsequent customers, time window violations are calculated under the assumption of having travelled back in time to the latest feasible time of arrival at customer i. Once again, the time window violation of a route $\mathrm{TW}(r_k)$ is computed as the sum of these modified violations at every customer, and $\mathrm{TW}(S)$ is evaluated by summing over all vehicle routes. Similarly, for battery and capacity violations, the penalty is modified to prohibit double counting constraint violations at future vertices due to an earlier violation of the constraint.

4.2 Pre-processing

To help reduce the number of arcs considered and improve computing times, we apply arc removal techniques from VRP literature. To this end, the number of arcs considered can be reduced by eliminating time-window, capacity and battery infeasible arcs. For this purpose, we remove any arc $(v, w) \in A_k$ for vehicle k for which the following property holds:

$$\begin{aligned} v, w \in P : &\quad q_v + q_w > C_k \\ v, w \in V : &\quad e_v + s_v + t^k_{vw} > l_w \\ v, w \in V, \forall x \in F : &\quad e_v + s_v + t^k_{vw} + s_w + t^k_{wx} > l_x \\ v, w \in V, \forall x \in F \cup O, \forall y \in F \cup D : &\quad \beta^k_{xv} + \beta^k_{vw} + \beta^k_{wy} > Q \end{aligned}$$

Further arc removal can be performed by considering paths associated with pairs of customer requests, as first proposed by Dumas et al. (1991) for the Pickup and Delivery Problem with Time Windows (PDPTW). To illustrate a single example, if it holds for two customers i and j that the partial path $\mathcal{P} = \{j^+, i^+, j^-, i^-\}$ is infeasible in terms of time windows, then the arc (i^+, j^-) is removed since the pickup of customer i cannot precede the drop-off of customer j.

4.3 Advance Requests

In order to construct vehicle routes and schedules for advance requests, this research uses a simple construction heuristic to obtain an initial solution, followed by a metaheuristic approach for improving on initial solution obtained. In the following section, these two procedures are outlined in detail. Section 4.3.1 first presents the various steps in the construction heuristic, after which the metaheuristic employed, and its key components, are mentioned in Section 4.3.2.

4.3.1 Construction Heuristic

In order to construct an initial starting set of vehicle routes, a construction heuristic inspired by the approach of Diana and Dessouky (2004) for the single-depot, static DARP is applied. The approach is simply extended to account for the potential need for vehicle by inserting recharging stations to prevent battery depletion along the route. The main idea is to first allocate a single request to each vehicle (these requests are known as seed requests), before iteratively adding the remaining unassigned requests to one of the vehicle routes. This is done by applying the regret insertion rule explained in Section 4.3.1.4.

4.3.1.1 Seed Requests

When employing construction heuristics to build a set of vehicle routes, a single request is often allocated to each vehicle before allocating the remaining requests. These first requests are referred to as seed requests. In VRP literature, many methods have been suggested for selecting seed customer requests in constructive heuristics. The following approach attempts to account for both temporal and spatial considerations when selecting initial seed requests. While temporal considerations may weigh more heavily in networks with narrow time windows and many vehicle requests, incorporating spatial factors helps to reduce vehicle travel times.

We assume n advance customer requests are known and that we have $m \leq n$ vehicles at our disposal. The following approach attempts to allocate one request to each of the m vehicles as the foundation of the initial solution construction. The underlying idea behind the procedure is to sort the various requests from most to least preferable candidates for being seed requests.

4.3.1.2 Temporal sorting

Initially, the customer requests are sorted in ascending order of their earliest pickup times e_i^P, and the first m customer requests are temporarily designated as the seed requests. One potential drawback to this approach is the possibility that the time windows of these requests m requests may be relatively spaced out such that one vehicle may potentially be able to serve multiple seed requests; in this case, the non-seed requests are searched in order to find a more suitable seed request. Specifically, if it holds for two requests k and i (where $i > k$) that k is a seed request and the same vehicle could potentially serve both requests then request i loses its seed status providing another request j can be found such that a single vehicle cannot serve both requests k and j. The complete procedure for this is outlined in Algorithm 1 below.

Algorithm 1 Temporal sorting procedure for determining seed requests

```
Input: Ordered sequence of $n$ requests sorted by earliest pickup time, $m$ available vehicles
Allocate requests at positions 1 through $m$ as seed requests;
Set $k = 1, j = 2, i = 3$;
for $k = 1 : n - 1$ do
    for $j = k + 1 : n$ do
        if request $k$ is a seed request then
            if requests $j$ is a seed request and $l_k^D + t_{k^-,j^+} + s_k^D \leq e_j^P$ then
                for $i = j + 1, ..., n$ do
                    if $l_k^D + t_{k^-,i^+} + s_k^D > e_i^P$ and request $i$ is not a seed request then
                        Set request $i$ as seed and request $j$ non-seed;
                        break;
                    end
                end
            end
        end
    end
end
Output: Ordered sequence of $n$ requests, with requests 1 to $m$ assigned as seed requests
```

4.3.1.3 Spatial sorting

Having sorted the requests temporally, a similar procedure is applied to further sort requests according to their spatial position. The objective is to avoid the deferring the insertion of requests that could be difficult to insert at a later stage. To this end, each customer request is assigned a *decentralisation index* (D_k) between zero and one which measures its total travel distance to other requests. It is calculated by the following expression:

$$D_k = \frac{\sum_{j=1}^{n} (t_{k^+,j^+} + t_{k^+,j^-} + t_{k^-,j^+} + t_{k^-,j^-})}{\sum_{k=1}^{m} \sum_{j=1}^{n} (t_{k^+,j^+} + t_{k^+,j^-} + t_{k^-,j^+} + t_{k^-,j^-})}$$

A larger value of D_k represents a more decentralised request and hence a greater likelihood of being difficult to insert into vehicle routes. Once again, all pairs of requests $(k-1, k)$ are considered, and the position of requests $k-1$ and k are swapped if request $k-1$ is sufficiently decentralised. Namely, this occurs when $D_k \geq D_{k-1} + \sigma$, where $0 \leq \sigma \leq 1$ is a user-defined parameter. Having considered all pairs $(k-1, k)$ of requests, the swapping process is repeated with all pairs $(k, k-2)$, $(k, k-3)$, and so on, until all such pairs have been evaluated and a final ordering of the requests has been determined. Then, the first m requests in the list are allocated to the m vehicles.

Unlike traditional DARP problems with a single depot station, vehicles are potentially located at different origin depot stations, and so allocating each request to a vehicle is a non-trivial process. To do this, a minimum weighted perfect matching problem on a bipartite graph is solved, using the Hungarian method. Here, the weight c_{ij} between a vehicle i and request j is calculated by multiplying the travel time between the vehicle's starting depot and request's pickup location $t_{o(i),j^+}$ by a scaling factor $\delta_{ij} = \frac{1}{2} + \frac{1}{2}\frac{(n-\text{RANK}(j)+1)}{n}$, where $\text{RANK}(j)$ denotes the position of request j in the list of ordered requests. The chosen scaling factor serves to increase the likelihood that customers that need served earlier are assigned a nearby vehicle. Having served an initial seed customer, each vehicle is scheduled to end its route by travelling to the destination depot nearest to the drop-off point of its served customer.

4.3.1.4 Regret Insertion

The remaining non-seed requests are inserted sequentially into vehicle routes, according to the following regret insertion procedure, briefly outlined in Section 3.2.1. In each iteration the procedure ensures that the request whose regret (defined as the cost of not immediately inserting the request immediately) is maximised, is inserted in the route which minimises the increase in objective value. The algorithm stores an incremental cost matrix IC (with rows i representing requests and columns j representing routes), measuring the increase in objective value from inserting the request in a given route. If no such insertion is feasible, the corresponding matrix entry is set to a large positive constant. For a given request i, regret (R_i) is then calculated by summing the differences between a route's incremental cost and the minimum incremental cost over all routes:

$$R_i = \sum_j (\mathrm{IC}(i,j) - \min_k \{\mathrm{IC}(i,k)\})$$

The request with the largest regret is immediately serviced by inserting it in the route k which minimizes incremental cost, before proceeding with the remaining requests in the same manner.

4.3.1.5 Charging Station Insertion

Finally, once all requests have been allocated to vehicle routes, the addition of charging stations to vehicle routes are considered. For each position i in a given route r_k, the insertion of the nearest charger between the vehicle stop at positions $i-1$ and i is evaluated in terms of the cost function described in Section 4.1. For each route, the insertion that yields the largest decrease in cost function is performed. Charging stations are added to the route until no further decrease in cost function can be realised by doing so.

It is worth noting that the list of initial routes for each vehicle may be infeasible in terms of vehicle load, maximum ride times, battery levels and time windows. Therefore, the hybrid metaheuristic solution approach described in Section 4.3.2 is employed in order to generate the best possible feasible solution. The method temporarily allows infeasible solutions, in order to help explore parts of the feasible solution space that typically remain unexplored when being restricted to feasible moves and neighbourhoods.

4.3.2 VNS/Tabu Search Heuristic

The metaheuristic solution approach is a hybrid of VNS and tabu search. While these two approaches have been used as standalone methods in DARP literature, their combination has surprisingly not yet been explored in the context of Dial-A-Ride Problems. Firstly, an outline of the proposed algorithmic approach is presented, before elaborating on the search procedures employed to explore the solution space.

4.3.2.1 Variable Neighbourhood Search

The following pseudocode outlines the approach taken to construct vehicle routes for all known advance requests. After constructing an initial solution x, as outlined in Section 4.3.1, a random solution x' in the neighbourhood $N_k^r(x)$ is obtained during the shaking phase, as in traditional VNS procedures. In contrast to more commonly employed intensification techniques such as steepest or simple descent, a solution x'' is obtained from x' through a tabu search algorithm, as is outlined more extensively in Section 4.3.2.3. A simulated annealing criterion is used to determine whether the solution x'' is accepted. If x'' is accepted, the algorithm repeats itself using x'' as the new starting point. If x'' is rejected, however, subsequent neighbourhoods are considered. Since the search procedure allows infeasible solutions temporarily, the algorithm also keeps track of the best found feasible solution x^*, which is updated throughout the search.

Algorithm 2 Variable Neighbourhood Search for Advance Requests of the e-ADARP

```
$x \leftarrow constructionHeuristic()$
$k \leftarrow 1$
$f(x^*) \leftarrow f(x)$
while $time\ limit\ not\ exceeded$ do
    $x' \leftarrow N_k^r(x)$
    $x'' \leftarrow tabuSearch(x', x)$
    if $acceptSA(x, x'')$ then
        $x \leftarrow x''$
        $k \leftarrow 1$
        if $f(x'') \leq f(x^*)$ then
            $x^* \leftarrow x''$
        end
    end
    else
        $k \leftarrow (k+1) \bmod |\mathcal{K}|$
    end
end
Output: Best found set of feasible vehicle routes $x^*$
```

Two classes of neighbourhoods are used in the shaking phase of the VNS/TS heuristic, both of which are well known in VRP literature. A description of each class of neighbourhood is provided below, together with the neighbourhood sequence used in the VNS algorithm.

The *swap neighbourhood (S)* of size n is defined as follows. First, n (or fewer if the route contains fewer than n) customers are randomly selected and removed from the route. This process is repeated for a different route, after which the removed requests from both routes are sequentially inserted into the other. For each customer request, the vertex with a tighter time window (typically pickup in the case of inbound customers and drop-off for outbound customers) is first inserted in the best position (evaluated in terms of the cost function), and thereafter the other vertex is optimally inserted, given the fixed position of the first. Figure 3 below illustrates the procedure of removing requests from a route. In this example, a sequence of size $n = 2$ is selected, leading to requests 2 being removed from one route and inserted in another.

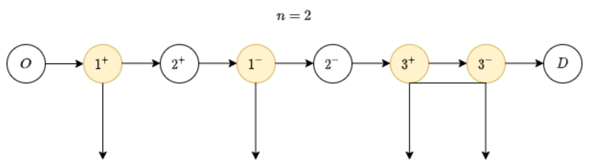


Figure 3: Swap neighbourhood sequence of size $n = 2$

The *chain neighbourhood (C)* incorporates the notion of an ejection chain, as introduced by Glover (1996) for the travelling salesman problem (TSP). To start with, two routes are selected and customers are removed from the first, before being sequentially inserted in the second, as in the swap neighbourhood. Afterwards, a sequence of vertices are ejected from the second and re-inserted in a random vehicle route. A chain neighbourhood may involve a maximum of $n + 1$ different vehicle routes (if no routes appear twice in the chain). This principle is illustrated below in Figure 4.

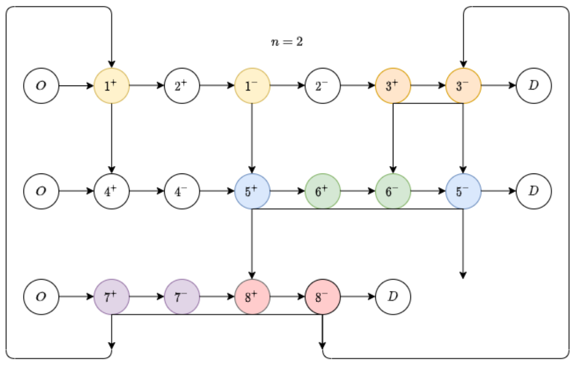


Figure 4: Chain neighbourhood sequence of length $n = 2$

The different neighbourhoods are ordered such that smaller neighbourhoods are explored before considering larger neighbourhoods. Therefore, to define the sequence of neighbourhoods, we alternate between swap and chain neighbourhoods, from sizes $n = 1$ to $n = 5$, yielding a total of $|\mathcal{K}| = 10$ neighbourhoods in the VNS.

4.3.2.2 Simulated Annealing Acceptance Criterion

During the embedded tabu search phase of the search procedure, a descent approach is employed from iteration to iteration, wherein exclusively improving solutions are accepted. In consecutive iterations of the VNS algorithm, however, a simulated annealing (SA) acceptance criterion is used. The idea behind this technique is to occasionally accept inferior solutions to avoid getting stuck in local optima. Given a current solution x and solution x'' obtained during the tabu search heuristic, an improving solution x'' is always accepted. Additionally, an inferior solution x'' will replace x as the current solution with some probability p, where p is calculated by the expression $e^{\frac{\Delta(x,x'')}{T}}$. Here $\Delta(x, x'')$ denotes the difference in objective value between the two solutions, and temperature parameter T controls the probability of inferior solutions being accepted. The temperature parameter is initially set to value T_{init} and is multiplied by a factor $0 < \epsilon < 1$ in consecutive iterations. This has the effect of accepting non-improving solutions with lower and lower probability as the search evolves. The initial value of the temperature parameter T_{init} is selected such that a solution x'' which is $\mu\%$ worse than initial solution x is accepted with probability $\frac{1}{2}$. Both μ and ϵ are user-defined parameters, in practice usually determined through parameter tuning.

4.3.2.3 Tabu Search

A tabu search (TS) procedure is used for intensification during the VNS algorithm, and thus aims to hone in on promising solutions in the neighbourhood of the solution attained through the shaking phase of the VNS. On the whole, the structure of the TS algorithm can be summarised as follows. First, a series of route operators, which each modify an existing solution in a specific manner, are used to generate the neighbourhood of a specific solution. Each non-tabu solution in this neighbourhood is then evaluated, after which the best move is executed and the tabu list (list of prohibited moves) is appropriately updated. However, before outlining the search neighbourhoods used in the tabu search, the idea behind a *granular* neighbourhood is first explained.

Granular Neighbourhood

To avoid exploring all other solutions that can be obtained by a given modification of the existing one and thus incurring lengthy computation times, granular neighbourhoods explore only a subset of possible elements and thereby restrict the size of the neighbourhood. Possible moves are filtered by a so-called sparsification rule, with a view to only considering “promising” sequences of vertices (i.e. sequences that are likely to yield high quality solutions). For example, recognizing that shorter arcs are more likely than longer ones to appear in high-quality solutions of the capacitated VRP (CVRP), Toth and Vigo (2003) define a threshold parameter such that only arcs shorter than a given length are considered. The list of these arcs, known as generator arcs, define a move when applied in conjunction with a specific neighbourhood operator η. In this research, an alternative sparsification criterion is applied. For each customer vertex, the nearest π% of incident customer vertices to each customer are maintained and sorted to define the set of customer arcs $\mathcal{A}_c(\pi)$ in the list of generator arcs. Similar lists are constructed to maintain a list of generator arcs between customers and charging stations, and between customers and depots. These lists are then combined and sorted to yield the complete list of generator arcs, $\mathcal{A}_g(\pi)$, which constitute the arcs in a sparse graph G'.

Neighbourhood Operators

In the tabu search phase, we also define four distinct neighbourhood operators. Firstly, *2-opt** exchange, which was introduced by Potvin and Rousseau (1995) for the VRPTW, is applied. The operator connects the first part of one route with the second part of another, and vice versa. It is applied in our problem between zero-sequences (before the pickup of a customer, when the vehicle is empty) of different routes to ensure the pickup and drop-off of customers are not scheduled to be performed by different vehicles. Since the classic 2-opt intraroute operator reverses the order of arcs located between two exchanged customers, it is often infeasible in settings where time windows are present. However, the 2-opt* operator “is particularly powerful for problems with time windows, because it preserves the orientation of the routes, and introduces the last customers of a given route (i.e. those with late time windows) at the end of the first customers of another route (i.e. those with early time windows)” (Potvin and Rousseau, 1995). Furthermore, the *relocate* and *exchange* operators introduced by Savelsbergh (1992) are applied both within and between vehicle routes. As their names suggest, these operators are used to relocate vertex positions (either within a route, or into other vehicle routes) or exchange the positions of vertices. The former operator is applied to recharging stations as well as customer requests, whereas the latter is only applied to customer requests. Finally, we also use an additional neighbourhood operator named *stationInRe*, which was proposed by Schneider et al. (2014) for the VRPTW with an electric vehicle fleet (E-VRPTW). For each generator arc (v, w) in the sparse graph G' such that $v \in S$, the operator removes the arc (v, w) and connects w to the predecessor of v if the arc (v, w) appears in a given vehicle route. Otherwise, the arc is inserted between w and its predecessor. This idea is depicted in Figure 5 below.

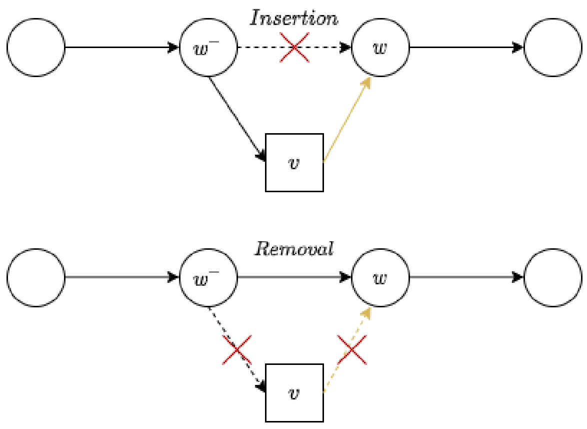


Figure 5: *stationInRe* operator for inserting and removing a charging station v

Move List
In order to effectively explore as much of the neighbourhood defined by the neighbourhood operators whilst keeping computation times as low as possible, a list of moves (whose solution value is evaluated by means of the cost function described in Section 4.1) is constructed and updated in consecutive iterations. Initially, every feasible move defined with respect to the various neighbourhood operators and initial solution, is evaluated. In each iteration, the best feasible non-tabu move is performed and the list is updated. Note that, in order to determine whether a move is feasible, both the tabu status (see the following heading) of the move as well as the presence of the involved arcs in the sparse graph is of importance. Namely, we require that at least one of the created arcs appears in the sparse graph G' in order for the move to be deemed feasible.

The updating of the move list is important in order to manage the computational complexity of the TS procedure. Firstly, certain moves (e.g. inserting an arc at a certain position in a given route) may no longer be feasible after an old solution has been replaced by a new one, therefore such solutions are removed from the move list. Furthermore, since it is only of interest to preserve promising solutions, only a fraction of promising solutions (evaluated in terms of the cost function with updated penalty parameter values) are kept while remaining moves are discarded. Then, a random neighbourhood operator is selected in each iteration and all new feasible moves are added to the move list.

Tabu Criteria
In order to avoid cycling, a tabu list is maintained through the TS procedure, which prohibits certain solution characteristics from appearing or reappearing in a solution for a given number of iterations. In this research, the approach of Goeke (2019) for the PDPTW with electric vehicles is utilised, by using arcs between vertices as the basis for the tabu criteria. When a neighbourhood operator is performed, one or more routes are modified, leading to both newly created, as well as removed arcs in certain routes. The created arcs and removed arcs associated with a given operator move are added to the tabu list when an operator is applied. Such moves are considered tabu if at least one (inter-route moves) or all (intra-route moves) of the created arcs are in the tabu list.

A few additional practical details pertaining to the implementation and maintenance of the tabu list are further required. When an item is added to a tabu list, its duration (in iterations) in the tabu list is drawn randomly from the uniform $[1, \Pi]$ distribution, where Π is user-defined. Furthermore, an aspiration criterion is exploited during the search procedure. This entails that the tabu status of each tabu item is revoked whenever a new best feasible solution is identified. An algorithmic overview of the tabu search procedure is presented in Algorithm 3 below.

Algorithm 3 Tabu Search for Advance Requests of the e-ADARP

```
Input: Current solution $x$, shaken solution $x'$ obtained from VNS procedure
$i \leftarrow 0$
$tabuList \leftarrow \emptyset$
$x_{TS}^{old} \leftarrow x'$
Construct $moveList$
while $i < \eta_{tabu}$ do
    $x_{curr} \leftarrow \emptyset$
    $x_{TS}^{new} \leftarrow \emptyset$
    Select random operator $\eta^r \in \mathcal{N}$
    for $m \in move(\eta^r, x_{TS}^{old})$ do
        Evaluate move $m$
        if $m$ is feasible $\wedge$ $m$ not tabu then
            $x_{curr} \leftarrow m(\eta^r, x_{TS}^{old})$
            Add $m$ to $moveList$
            if $f(x_{curr}) < f(x_{TS}^{old})$ then
                $x_{TS}^{new} \leftarrow x_{curr}$
                if $f(x_{curr}) < f(x^*) \wedge x_{curr}$ feasible then
                    $tabuList \leftarrow \emptyset$
                end
            end
        end
    end
    if $x_{TS}^{new} \neq \emptyset$ then
        $x_{TS}^{old} \leftarrow x_{TS}^{new}$
        break
    end
    Update penalty parameters
    Update $moveList$
    Update $tabuList$
    $i \leftarrow i + 1$
end
$x'' \leftarrow x_{TS}^{old}$
Output: Solution $x''$ from tabu search procedure
```

4.4 Dynamic Requests

When a new customer request arrives during the planning horizon, one potential solution approach would be to solve a new instance of the e-ADARP for all currently unserved requests, using the methodology outlined in Section 4.3 for advance requests. In practice, however, such an approach is often infeasible due to the sheer computation time required, especially for cases when the frequency of new requests and the degree of dynamism of the system is high. Additionally, many applications of the DARP require that new prospective customers can be quickly informed whether their request can be accommodated, further rendering this approach impractical. For this reason, a simple insertion procedure is applied in order to quickly inform a new customer whether their request can be accommodated, with a post-improvement phase dedicated to improving solutions while no new requests have arrived.

Immediate Insertion Heuristic

Quick and efficient insertion procedures are generally favoured above iteratively resolving static instances for large instances of the dynamic DARP. Typically, this is done by means of an immediate insertion rule, whereby new requests are immediately inserted into existing vehicle routes as soon as they enter the system. This offers the benefit to prospective customers that they can quickly find out whether or not their request can be accommodated by the system, without incurring long waiting times. In this research, the immediate insertion heuristic evaluates all feasible pairs of pickup and drop-off insertion positions in each route, and selects the pair and route which yields the smallest increase in the solution cost. Finally, if this resulting solution is battery-infeasible, the heuristic further attempts to insert a charging station visit in the chosen route, until a feasible solution is identified. If the addition of the request and charger do not yield a feasible schedule, the request is rejected.

An alternative to an immediate insertion procedure is a rolling horizon heuristic, which inserts requests on the basis of their earliest pickup time, instead of the call time approach as in the immediate insertion rule. The underlying idea is that by delaying the scheduling of customers until a later point in time, only the most urgent customer requests are inserted in vehicle routes, hence allowing for flexibility in making scheduling decisions of new customers. Computation time is also saved, since requests with long lead times are not included in the search process until shortly before they need to be scheduled. Although Luo (2006) found that a rolling horizon approach outperforms an immediate insertion rule both in terms of computation time and service quality for the standard DARP, the drawback of such an approach is that customers are often forced to wait long lengths of time to discover whether their request can be serviced. In practice, this type of insertion procedure may not be suitable for many practical Dial-A-Ride Services where customers require a quick response to their request. The general structure of how dynamic requests are handled in this research is outlined in Algorithm 4. Important to note is that the post-improvement procedure of the online insertion heuristic involves running a dynamic implementation of the VNS/TS heuristic outlined in Section 4.3 (with vehicle schedules fixed up until time t) One notable difference between the dynamic and static VNS/TS heuristic, however, is that the dynamic version only accepts solutions which are feasible with respect to all constraints in the acceptance phase of the VNS procedure.

Algorithm 4 Immediate Online Insertion Heuristic

```
Input: Vehicle routes and schedules for accepted requests
Initialise vehicle locations;
Set post-improvement interval $\Delta$ and service horizon $T$;
Set $k = 1$, $t_k = t = 0$;
while $t < T$ do
    Update vehicle locations at time $t_k$;
    Fix vehicle schedules up until time $t_k$;
    if new request $i$ arrives then
        Set the cost of insertion of request $i$ into the current schedule: $c(i) \leftarrow +\infty$;
        Set the chosen route for insertion $r^* \leftarrow \emptyset$;
        for each route $r$ do
            Evaluate the cost $c(i_1, i_2)^r$ of each feasible pair of insertion positions $(i_1, i_2)$ for the pickup
            and drop-off of request $i$ in route $r$;
            if $c(i_1, i_2)^r < c(i)$ then
                if insertion is feasible then
                    Set $r^* \leftarrow r$;
                    Set $(i_1^*, i_2^*) \leftarrow (i_1, i_2)$;
                    Set $c(i) \leftarrow c(i_1, i_2)^r$;
                end
            end
        end
        if $r \neq \emptyset$ then
            Insert pickup and drop-off at positions $(i_1^*, i_2^*)$ in route $r^*$;
        end
        else
            Set $(i_1^*, i_2^*, r^*) = \arg\min_{i_1,i_2,r}\{c(i_1, i_2)^r\}$;
            for each feasible position $j$ of insertion in route $r^*$ do
                Evaluate insertion of nearest charger between stops at positions $j - 1$ and $j$ in $r^*$;
                if insertion of charger is feasible then
                    Insert pickup, drop-off and charger at positions $(i_1^*, i_2^*, j)$ in route $r^*$;
                    break;
                end
            end
        end
    end
    while $t < t_k + \Delta$ do
        Apply post-improvement procedure;
        if new request $j$ arrives then
            break post-improvement procedure;
        end
    end
    Set $k \leftarrow k + 1$;
    Set $t_k \leftarrow t$;
end
Output: Updated list of vehicle routes and schedules for accepted customers
```

4.5 Vehicle Scheduling and Battery Management Decisions

The Dial-A-Ride Problem entails decision making with respect to both routing (which customers to allocate to each vehicle, and in what order to visit each customer) and scheduling (what time to arrive, begin service at, and depart from each stop). In terms of scheduling, several different strategies have been proposed in DARP literature, each with their own potential advantages and drawbacks. Particularly in a dynamic environment, where future requests are unknown while making scheduling decisions, the differences in performance between approaches can become large. In the case of the e-ADARP, however, the decision making process is further complicated by the presence of charging stations, whose service duration is a decision variable. Therefore, for the purpose of this research, three different recharging strategies are considered.

4.5.1 Battery Management

Careful consideration of the strategy employed by vehicles at recharging stations has been shown to have a large impact on the solution quality for various VRPs that involve the use of electric vehicles. For instance, Keskin and Çatay (2016) find that partial recharge strategies outperform full recharging. However, in a dynamic environment, full recharging offers the advantage that new requests are more likely to be inserted in existing vehicle routes without additional trips to recharging stations, since active vehicles will typically have more remaining charge. Partial recharging may require more regular, shorter periods at recharging stations, which may entail more vehicle detours to charging stations than when fully recharging. Which of the two is more effective in a dynamic environment is therefore non-trivial and not immediately clear.

The first policy considered for this research is a **full** recharging strategy, which completely recharges a vehicle upon its visit to a charging station. This strategy offers the possible advantages of building a reserve of battery to accommodate future customers with, along with relatively few trips to charging stations and thus less time spent on detours. The drawback, however, is potentially being unable to accommodate some customers due to long periods spent charging the vehicles. Secondly, a **minimal** strategy is utilised, which minimises the time spent at each charging station by supplying a vehicle with the bare minimum amount of charge required to reach the next charging or depot station scheduled in its route. The upshot to this strategy is that relatively little time is spent at charging stations, thus allowing for increased flexibility for serving customers. However, more frequent visits to charging stations may entail larger detours, while incorporating dynamic requests may be difficult due to insufficient battery levels. Finally, a **conservative** recharging strategy offers some middle ground between the previous two approaches in terms of the time spent at a given charging station. For each scheduled visit to a charging station, the policy determines the amount additional time that could be spent charging the vehicle, while not delaying the planned time of arrival at the subsequent planned customer visit. To build a certain level of battery reserve for incorporating dynamic requests, this additional time is fully allocated to charging at the station, and so customers travel to the charging station as soon as service has been completed at the previous stop. While this may allow new requests to be incorporated without violating battery constraints, the downside to such a policy relative to a minimal one is less flexibility in the vehicle schedules. This occurs as the decision to visit the charging station is not postponed once service at the previous stop is complete. Important to note is that all three recharging policies are independent from (full and minimal policies), or can be adapted towards (conservative policy), alternative scheduling procedures - not solely the procedure outlined below in Section 4.5.2.

4.5.2 Scheduling Policy

Having determined a sequence of vertices to visit for every vehicle route, a post-processing procedure is applied with a view to decreasing customer ride-time and time window violations. To this end, we denote by A_i, B_i and D_i respectively the arrival, beginning of service and departure times from a given vertex i. The waiting time W_i when arriving early at a customer is computed as the difference between the beginning of service and arrival times, while $T_{i,j}$ denotes the travel time between vertices i and j. Furthermore, we define a set $J_k^- = \{i^- \in \{k+1, ..., d\} : i^+ \in \{1, ..., k\}\}$ as the set of drop-off vertices that are yet to be serviced, but whose corresponding pickup start of service times cannot be altered.

The scheduling algorithm used in this research can be seen as a dynamic implementation of the scheduling procedure outlined for the static DARP by Cordeau and Laporte (2003), formalised by Berbeglia et al. (2012) for the regular dynamic DARP. The underlying idea behind the procedure is to first serve all customers as early as possible, in order to minimise the time window violation at each customer. Hereafter, recognising that customer ride times can potentially be reduced by delaying the start of service at the pickup node, the ride time violations of the customers are subsequently minimised by delaying the start of service. The amount of time by which this can feasibly be done without increasing time window or ride time violations is known as the *forward time slack*, a concept introduced by Savelsbergh (1992). Full details of this algorithm can be seen in Algorithm 5 below.

Algorithm 5 Vehicle Scheduling Algorithm for the e-ADARP

```
Input: Vehicle route $r = \langle 0, ..., k+1, ..., d \rangle$; index $k \in \{-1, ..., d-2\}$ of last vertex whose start of service time cannot be modified; current time $t$
1. Service all customers as early as possible to minimise time window violations
Set $B_{k+1} := \max\{e_{k+1}, A_{k+1}, t\}$
Set $D_{k+1} := B_{k+1} + s_{k+1}$
for $j = k + 2$ to $d - 1$ do
    Set $A_j := B_{j-1} + T_{j-1,j}$
    Set $B_j := \max\{e_j, A_j\}$
    Set $D_j := B_j + s_j$
    Set $W_j := B_j - A_j$
end
Set $A_d := D_{d-1} + T_{d-1,d}$
2. Minimise ride time violations by delaying the start of service at each customer vertex
for $h = k + 1$ to $d - 1$ do
    2a. Delay start of service as much as possible without increasing violations
    Set $B_h := A_h + \min\{\min_{j \in \{h,..,d\}}\{\sum_{u=h}^{j} W_u + (l_j - B_j)^+\}, \min_{i:i^- \in J^-_{h-1}}\{\sum_{u=h}^{i^-} W_u + (L_i - P_i)^+\}\}$
    Set $D_h := B_h + s_h$
    Set $W_h := B_h - A_h$
    Set $A_{h+1} := D_h + T_{h,h+1}$
    2b. Propagate updated values along the route
    for $f = h + 1$ to $d - 1$ do
        Set $B_f := \max\{e_f, A_f\}$
        Set $D_f := B_f + s_f$
        Set $W_f := B_h - A_h$
        Set $A_{f+1} := D_f + T_{f,f+1}$
    end
end
3. Update customer ride times
for request $\{i^+, i^-\} \in \{0, ...d\}$ do
    Set $P_i := B_{i^-} - B_{i^+}$
end
4.  Delay departure from each vertex
for $j = k + 1$ to $d - 1$ do
    Set $D_j := B_{j+1} - T_{j,j+1}$
    Set $A_j := B_j$
    Set $W_j := B_j - A_j$
end
Output: Vehicle route $r = \langle 0, ..., k+1, ..., d \rangle$
```

For a given battery management policy (and hence predetermined service times at each charging station), steps 1-3 of this procedure have the resulting advantage of finding a feasible schedule if one exists, given the sequence of visited vertices. However, the drawback of delaying service times in step 2 of the algorithm is that inserting new arrivals may become more difficult than alternative procedures where customers are served as early as possible. Starting later may thus result in more customer rejections. The advantage of delaying departures from each vertex as far as possible in step 4 is that vehicle schedules have greater flexibility to re-route (alter their next planned stop) in the event of new customer requests as departure from the current stop has been deferred.

5 Data

To obtain the necessary data for carrying out the metaheuristic approach proposed in Section 4, existing problem instances from the literature are used. Specifically, benchmark instances proposed for the DARP by Cordeau (2006) are modified by supplementing the instances with additional problem-specific characteristics (as done in the work of (Bongiovanni et al., 2019)).

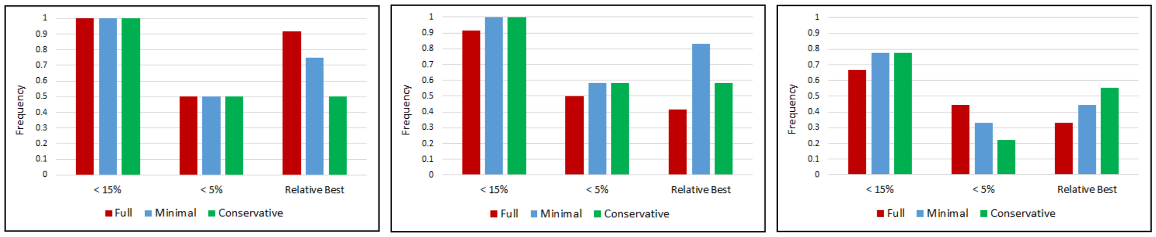
For the standard DARP, Cordeau (2006) generated problem instances by sampling pickup and drop-off locations for n customers randomly and independently from a uniform distribution in a square of size $[-10, 10]$ x $[-10, 10]$. The travel cost c_{ij} along an arc (i, j) is set equal to the Euclidean travel distance t_{ij} along the arc. For the e-ADARP, these instances were supplemented with charging stations and battery consumption data by Bongiovanni et al. (2019). Since the time window is initially undefined for either the drop-off or pickup of each request, the time window tightening procedure as illustrated in Figure 1 is applied. Moreover, each instance includes an equal split of inbound and outbound requests. Furthermore, each request has a maximum ride time of 30 minutes, irrespective of the necessary travel distance between pickup and drop-off. A total of 13 problem instances are used (12 as static instances and the other is used for the dynamic requests), each with a different number of vehicles and requests. The number of vehicles ranges between 2 and 5, while the number of requests ranges between 16 and 50.

6 Results

In this section, the quality of solutions obtained by the VNS/TS heuristic under different battery management strategies is analysed for the various benchmark problem instances and compared to the optimal known values for these static DARP instances. Subsequently, the performance of the various battery management strategies is analysed and compared in a dynamic setting.

6.1 Static Instances

Figure 6 below presents descriptive statistics concerning the performance of the VNS/TS heuristic. The solutions obtained for each instance are compared with the best known solution (BKS)[1] for the instance, and the performance of the various strategies are compared. Specifically, the fraction of instances for which the VNS/TS heuristic obtains a solution quality within 15% and 5% of the best known solution is presented. Furthermore, the fraction of instances for which each policy finds a solution at least as good as the other two policies is displayed. These statistics are reported for the final vehicle battery level requirements $r = 0.1, 0.4$ and 0.7. Full details of the solution values obtained for each instance can be found, along with the best known solution (BKS) for the instances, are presented in Appendix A. For all instances, the heuristic is run for a period of 60 seconds.



(a) $r = 0.1$ **(b)** $r = 0.4$ **(c)** $r = 0.7$

Figure 6: Performance characteristics of different recharging strategies under various final battery requirements r

Firstly, the differences in solution quality between the solutions obtained and the BKS of the various instances demonstrate the fact that the VNS/TS heuristic can find quick, feasible and good quality solutions for a set of customer requests. In the vast majority of instances for which a feasible solution exists, the metaheuristic identifies a valid solution, for all three recharging strategies. Furthermore, in the vast majority of cases, these solutions are within 15% of the optimal objective value, frequently within 5%, and in some cases even correspond exactly to the BKS. However, it can be seen that when the value of r is increased, it becomes increasingly difficult to find a feasible solution due to tighter battery constraints, the heuristic finds high quality solutions less frequently. This is not entirely surprising; the introduction of charging constraints introduces increased complexity to the problem, yet the heuristic employed for this purpose does not rely on exact optimisation techniques to deal with this dimension of the problem. This result supports the first research hypothesis **H1**.

When comparing the various recharging policies, it can be observed that when all requests are known in advance, a full recharging policy typically performs relatively well when the final battery requirement is low ($r = 0.1$). This appears in line with intuition, since longer but less frequent visits to charging stations entails relative little detouring. Differences in performance between policies are small as fewer visits to charging stations lead to similar solutions under the different policies. Furthermore, the full recharging strategy does not perform as well in situations where the battery

[1]The BKS does not necessarily correspond to the optimal solution for each instance as the exact method used to solve the instances does not always converge to optimality within a pre-specified time limit.

constraints on the vehicle are harder to satisfy ($r = 0.7$). Longer periods spent recharging vehicles may lead to reduced flexibility in scheduling customer visits and hence worse solutions. Therefore, it appears logical that the relative performances of the conservative and minimal strategies, which rely on more frequent and shorter charging visits, improve as the value of r increases.

Nevertheless, it is not the case that the best possible attainable solution using a minimal strategy outperforms the equivalent using a full recharging strategy in a static setting. For example, consider a vehicle route containing two intermediate stops at charging stations and a minimal recharging policy. If there are many scheduled requests in quick succession following the second visit to the recharging station and relatively few following the first, it may be beneficial to increase the amount of time spent charging at the first station. This allows the vehicle to decrease the time spent charging at the second, in order to satisfy both the battery and time window constraints. Therefore, if multiple charging stations are present in the route, the charging decisions are not necessarily optimal and hence could also potentially be improved by a single, longer visit to a charging station, as is custom in the full recharging policy. Alternatively, arriving as early as possible at planned charging station visits, as policy dictates in the conservative strategy, may constitute an improving strategy too in such a scenario.

In addition, it is interesting to note that the conservative recharging strategy performs increasingly well relative to the other policies in many instances where the battery constraint is tightest. This appears to be natural, as the use of idle time to charge vehicles often leads to battery feasible solutions.

When comparing the three approaches in a static setting, it is clear that these three charging strategies all contain their relative advantages and disadvantages, yet all constitute potentially viable procedures for quickly obtaining feasible schedules in a VNS/TS procedure. However, the drawbacks of a full recharging strategy become apparent as the battery constraint is tightened - conditions under which the other two strategies perform comparatively better.

6.2 Dynamic Requests

In a dynamic setting, the effectiveness of various recharging strategies may differ based on a number of factors. The three strategies (F, M, C) are analysed and compared in a dynamic environment in order to assess which of the three is most preferable for incorporating new requests into existing vehicle schedules. To do this, the effect of different lead times and degrees of dynamism on the relative effectiveness of the various recharging strategies is assessed. For this purpose, an instance `a5-50-0.7` from Appendix A is utilised. This instance uses 5 vehicles, contains 50 customer requests and imposes a final battery requirement of 70% of the vehicle battery capacity at the destination depot.

The computational experiments are structured as follows. Firstly, a random subset containing a fraction $1 - \delta$ of all requests are considered, and a static instance is solved using the VNS/TS heuristic. Subsequently, the remaining fraction δ of the requests are treated as dynamic requests with a given lead time L for each dynamic request. A dynamic instance is solved for each recharging strategy. For each value of δ and recharging strategy, this process is run a total of 10 times, such that the assignment of dynamic and static requests differs in each run, but the number of requests assigned to each category does not. The request acceptance probability of the dynamic requests reported in Section 6.2.1 therefore takes the average acceptance rate dynamic requests across all runs. To measure the effectiveness of the various strategies, we consider this average acceptance rate of dynamic requests as a metric for service quality. The effects of request lead times and degree of dynamism on the system performance are investigated.

6.2.1 Comparison of Policies

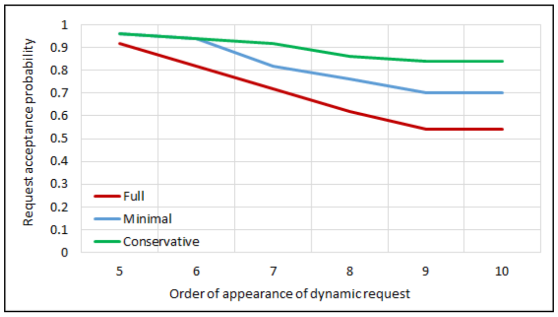
For various system degrees of dynamism, the service quality of online requests under the three recharging policies is considered. The summary statistics for each recharging strategy and degree of dynamism can be found in Table 1 below, using a lead time of $L = 60$ minutes for each dynamic request.

Table 1: Fraction of accepted requests for different recharging policies and degrees of dynamism δ

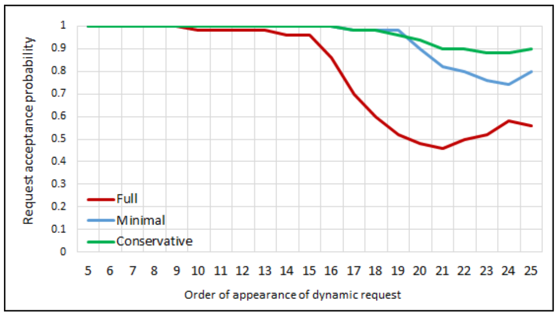
	Dynamic Requests			All Requests		
δ	Full	Minimal	Conservative	Full	Minimal	Conservative
0.2	0.73	0.83	0.90	0.95	0.97	0.98
0.5	0.80	0.94	0.97	0.90	0.97	0.99
0.8	0.73	0.93	0.96	0.78	0.94	0.97
1	0.78	0.95	0.96	0.78	0.95	0.96

Firstly, it can be seen that there exists clear differences between the various recharging strategies in their ability to accommodate incoming requests during the planning horizon. For all degrees of dynamism, the conservative strategy outperforms the minimal strategy as it can facilitate the greatest fraction of incoming requests. In a dynamic environment, this empirically supports the benefit of utilising idle slack time before planning charging station visits in order to build battery reserve for accommodating new requests. In turn, the minimal strategy outperforms the least effective full recharging policy. Hence, while the effectiveness of the recharging strategies on travel times is still somewhat ambiguous in a static setting (using heuristic methods), there can be little confusion surrounding the effect on service quality in the dynamic case. As such, these findings provide strong evidence of research hypothesis **H2a**. Additionally, we can see that the total fraction of all requests that are accepted increases when more requests are known in advance. This suggests that advance knowledge of requests benefits the performance of the system, as more timely awareness of the request leads to increased flexibility for constructing vehicle schedules. This is explored in greater depth by considering the effect of lead times and the value of information (VoI) in Sections 6.2.2 and 6.2.3, respectively. The fact that, on the whole, it can be seen that the fraction of accepted dynamic requests decreases in the degree of dynamism is logical too, as the starting vehicle schedules obtained from the static instance are increasingly tight when more requests are static. When considering (partially) dynamic vehicle routing problems, Wong et al. (2014) observe an interesting phenomenon which they refer to as the "dilemma" zone, in which the performance of a system of vehicles is more effective when the system is entirely dynamic, as compared to when some requests are known in advance. The authors attribute this to the fact that, when there are a small subset of advance requests, solving a static instance entails committing a fixed position in the vehicle schedule to certain requests. These positions may transpire to be sub-optimal later on as new requests appear throughout the planning horizon. This effect may partly explain the fact that there is little to no improvement in the performance of all strategies when moving from the $\delta = 1$ to the $\delta = 0.8$ case.

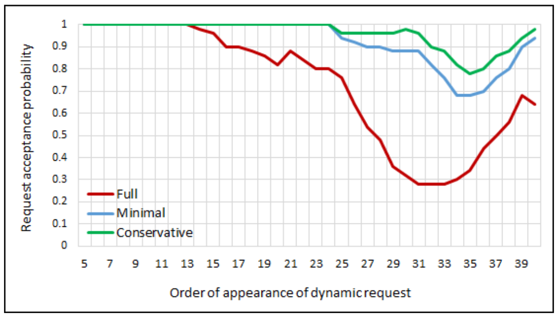
As well as considering the overall fraction of accepted requests under each policy, the timing of accepted and rejected requests also offers insight into the relative benefits and drawbacks of the various policies. Particularly, the ability to accommodate new requests during the planning horizon is highly relevant in situations where vehicle schedules become fuller and battery constraints become increasingly tight. To this end, we consider the moving average acceptance probability under the different strategies. The moving average acceptance probability of a dynamic request n is computed by averaging the acceptance probability of the previous five observations (dynamic requests $n - 4$ through n). The moving average is used instead of the "standard" average in order to observe a general trend across the dynamic requests, rather than observing sharp rises and falls in the trend in when specific dynamic requests prove to be easier or harder to insert into vehicle routes than others.



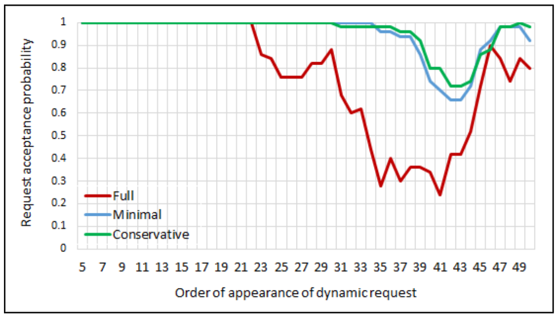
(a) $\delta = 0.2$



(b) $\delta = 0.5$



(c) $\delta = 0.8$



(d) $\delta = 1$

Figure 7: Moving average (of observations $n - 4$ through n) acceptance fraction of dynamic requests under different recharging strategies for various degrees of dynamism δ

Figure 7 illustrates that notable differences exist between the different strategies when attempting to insert new requests into existing routes. Irrespective of the recharging policy, a general trend is evident: the probability of incoming requests being accepted starts very high, before experiencing a decrease during the time horizon as time elapses. In general, a rise in acceptance rate towards the end of the planning horizon is observed too. This observation can likely be attributed to the fact that there are slightly fewer requests in the last couple of hours of the planning horizon than in the hours beforehand. The distribution of earliest pickup times for all requests is presented in Figure 11 of Appendix B. Although there are small peaks in demand in certain hours, it can be observed that the requests are relatively evenly spread over the planning horizon. Of course, if the frequency of requests were to steadily rise over time, it would not be possible to attribute the effect observed in Figure 7 to the effects of the recharging policy, as a decline in acceptance rates could also be explained by the change in customer demand. However, Figure 11 in Appendix B suggests that this is not the case for the customer requests considered.

Nevertheless, in all cases it is seen that the conservative strategy is best equipped to avoid the drop in service quality as schedules become tight and battery levels depleted, while the full recharging strategy experiences large gulfs in acceptance probabilities at different points in time. Although this finding cannot be generalised to all DARPs of this form, the results highlight the benefit of the conservative strategy relative to the other two; namely, in striking a balance between building battery reserve without compromising flexibility to schedule future requests.

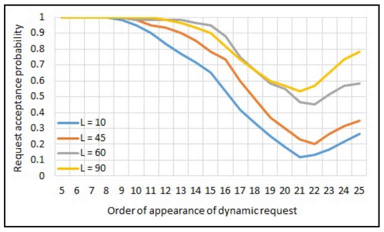
6.2.2 Lead Time Effect

The lead time L of a given request indicates the amount of time between the appearance of the request and the earliest point in time at which pickup can occur. Intuitively, the larger the call time of a given request, the greater the likelihood of being able to insert the request somewhere in the vehicle schedule, all else being equal. This can be explained by the fact that a short lead time may restrict the number of possible insertion positions for the new request and offer vehicles fewer opportunities to reschedule requests planned around the same time as the new request. In this section, the effect of different request lead times on the acceptance rate of new requests under the three recharging rules is investigated. Once again, the instance `a5-50-0.7` is used, with parameter δ set to value 0.5.

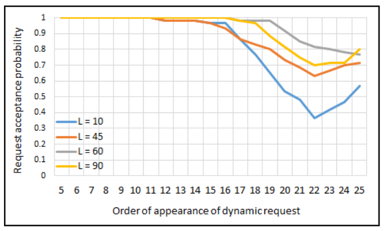
Table 2: Fraction of accepted dynamic requests for various recharging policies and lead times

Lead time (min)	Full	Minimal	Conservative
$L_i = 10$	0.59	0.81	0.88
$L_i = 45$	0.67	0.88	0.93
$L_i = 60$	0.80	0.94	0.97
$L_i = 90$	0.84	0.92	0.96

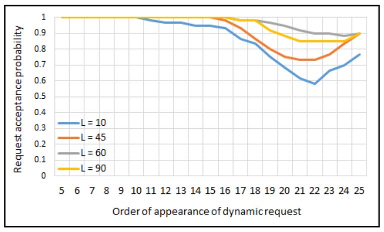
Table 2 illustrates the fact that greater advance warning of incoming customer requests directly leads to a greater proportion of incoming requests being accepted for all recharging strategies. While this effect holds across all three policies, we can see that service quality is most sensitive to request lead times for a full recharging policy. As mentioned above, using a full recharging strategy is relatively inflexible approach but leads to less time spent travelling. It therefore stands to reason that such a policies would benefit relatively more from greater advance warning of requests than more flexible policies. Note however that longer lead times do not guarantee a larger service quality, since decisions made at a given point in time can potentially give rise to schedules which are inflexible in terms of accepting future requests. Furthermore, comparable proportions of accepted requests for cases $L_i = 60$ and $L_i = 90$ suggest that the benefit of larger lead times diminishes beyond a given point.



(a) Full recharging policy



(b) Minimal recharging policy



(c) Conservative recharging policy

Figure 8: Moving average (of observations $n - 4$ through n) acceptance fraction of dynamic requests for different recharging strategies under different lead times L

As illustrated in Figure 8, the acceptance probability for the various recharging policies, across different lead times L for dynamic requests. Once more, it can be observed for all policies that the acceptance probabilities tend to start high, before decreasing towards the end of the planning horizon and finally increasing slightly at the end of the planning horizon. Once again, the relationship between service quality and lead times is most pronounced for the full recharging policy, since the schedules obtained by this strategy tend to be less flexible than schedules using the other policies, and hence benefit most from the prior warning of new requests. Consequently, the results support the validity of research hypothesis **H2b**. As well as maintaining the highest overall acceptance rate, the conservative strategy also maintains the largest minimum acceptance rate of all policies, suggesting it to be more robust to a decrease in lead times than its minimal and full counterparts.

All in all, it is clear that lead times play a crucial role in the service quality under all strategies. While all policies respond similarly to an increase in lead times, the performance, in terms of acceptance rate of new requests, appears most robust to drops in request lead times for the conservative strategy. For practical applications, where lead times may vary or be unknown, this is a useful property to observe.

6.2.3 Value of Information

In order to assess the effectiveness of a dynamic Dial-A-Ride system, the service quality - as measured by the fraction of accepted requests - is an important metric, as fleet operators seek to maximise service quality to their customers. However, amongst different branches of VRP, the Dial-A-Ride Problem is particularly challenging since operational efficiency and service quality requirements are often traded off against one another. As well as maximising the service quality, minimising operational cost is important as operators aim to perform the best possible service at the lowest possible cost.

In Section 6.2.2, the benefit of advance information (i.e. being informed of a new request's presence in due time before the request can be serviced) in a dynamic e-ADARP setting was illustrated. It was shown that longer lead times result in greater service quality, for all recharging strategies considered. Although the service quality is undoubtedly worsened by dealing with shorter lead times, whether or not the operational efficiency suffers as a consequence is an important aspect to consider. To this end, this research investigates the effect of advance information on the operational cost of vehicle routes constructed by the VNS/TS heuristic. Operational cost is measured by the total travel time of all vehicles. To do this, the solution obtained for the static version of instance `a5-50-0.7` is compared to the various solutions obtained when considering the same instance in a dynamic setting with lead time $L = 60$. Since rejected requests appear to "improve" operational efficiency - as vehicles travel further when a request is accepted than when the request is rejected - the comparison can only fairly be made for dynamic instances for which all incoming requests are accepted. Due to the limited number of cases in which this occurs for the full and minimal recharging policies in a dynamic setting, this analysis is conducted for the conservative recharging strategy only. For the same reason, the case $\delta = 1$ is omitted. The ratio of the total vehicle travel times in a dynamic setting (c_d) to vehicle travel times in a static setting (c_s), referred to as the Value of Information (VoI), provides a measure of the value of advance information on operational efficiency:

$$VoI = \frac{c_d}{c_s}$$

A higher VoI suggests that the operational efficiency of solutions found is sensitive to having customer requests available *a priori*, whereas a lower VoI implies that whether the requests appear during the planning horizon has less of an impact on operational efficiency. For fair comparison between the solution values in the static and dynamic case, the static heuristic is run for a length of time corresponding to the total run time during the dynamic instance of the problem. In the discussion below, the average VoI is computed by averaging the VoI for a given dynamic instance across all runs for which all requests were accepted. Since the number of data points used to construct this average may differ between values of δ, the standard error of the observation is included in error bars in Figure 9.

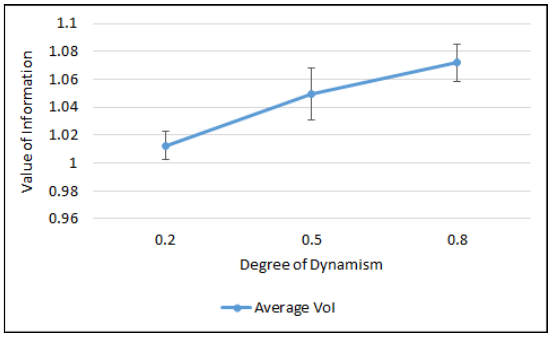


Figure 9: Average VoI under conservative recharging strategy for different degrees of dynamism δ

In Figure 9, the average VoI for the conservative recharging strategy and different values of δ is presented. As we can see, the VoI is increasing in the degree of dynamism, from 1.01 to 1.05 and 1.07 for $\delta = 0.2, 0.5$ and 0.8 respectively (or average increases in total travel time of 9, 38 and 55 minutes with respect to the static instance). The results in Figure 9 suggest that the operational efficiency is improved by having more requests known in advance. Therefore, in addition to more advance information about requests leading to higher service quality, the figures suggests that the effect of advance information on operational efficiency is positive.

Since Figure 9 above considers a special case, namely when all dynamic requests are accepted, caution must be taken in drawing conclusions on operational efficiency. Alternatively, in order to make statements regarding the value of advance information for cases in which not all requests are accepted, a proxy estimate for the rejected customer travel time can be used. Namely, by evaluating the total time that a new vehicle, dispatched from the depot, would require in order to serve a rejected request and return to its original depot, the total travel time of all customers can be computed. The resulting VoI metric obtained by this approach can be viewed as an indicator of the VoI. Furthermore, using this modified metric ensures that no particular trials are discarded from the analysis when customer requests are rejected.

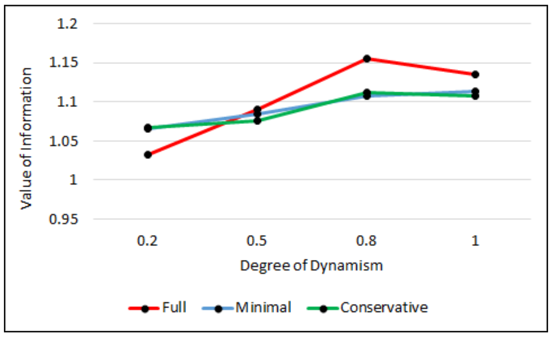


Figure 10: Average proxy VoI of different charging strategies for different degrees of dynamism δ

Figure 10 above displays the average proxy VoI metric for all three recharging strategies and multiple values of δ. Once more, the instance `a5-50-0.7` is used with lead time $L = 60$. When considering the Value of Information in Figure 10 using this modified measure of total travel time, the general trend shows that the VoI increases in the degree of dynamism of the system, and that total travel costs are smaller when more requests are known in advance (the VoI remains above one in all cases). Despite the potential drawbacks of using a proxy measure to compute the VoI, the results further support the notion that advance knowledge of requests ultimately leads to more operationally efficient vehicle routes.

7 Conclusion

In recent years, the emergence and popularization of electric vehicles has altered the landscape of vehicle routing problems, as research models are adapted to deal with the practical constraints that fleet operators experience in reality. Furthermore, the projected increased use of autonomous vehicles entails additional complexity for modellers, which requires heuristic methods for large-scale, practical applications. This research considers a many-to-many, dynamic DARP, and considers the effectiveness of three simple recharging policies, namely a full, minimal and a conservative recharging policy.

A full recharging strategy completely recharges a vehicle's battery during its trips to charging stations, leading to fewer but longer trips to charging stations. This has the advantage of fewer detours during vehicle route, yet the drawback is less flexibility in the schedule of the vehicle - as long trips to charging stations can typically only occur at a small number of points throughout the day. On the other hand, a minimal approach works by only recharging a vehicle's battery with enough charge to reach the subsequent charging station or depot along the route. While this keeps charging station visits short, the disadvantages are that accommodating new requests may become difficult in terms of vehicle battery, and that more detours to charging stations are needed than under a full strategy. The conservative strategy offers a middle ground between the two approaches, by attempting to build a reserve of battery for accommodating new requests, while maintaining flexibility in the schedules of the vehicles. To this end, solutions are constructed by modifying the solution obtained through a minimal strategy such that any trips to charging stations occur as soon as service is completed at the previous customer, as opposed to waiting until the originally planned departure time. To solve an online Dial-A-Ride Problem and investigate the effectiveness of the various policies on operational efficiency and service quality, a VNS/TS heuristic was utilised in this research, with an immediate insertion rule employed for accommodating new customer requests.

When all requests are known in advance, it becomes clear that the VNS/TS offers a viable metaheuristic solution method for obtaining quick and good quality solutions for minimising the total vehicle travel distances, under all three recharging policies. The differences in performance between the policies are fairly similar in a static setting, although the results suggest that a conservative and minimal approach perform better than a full recharging strategy in the case of tight battery level constraints.

However, the differences between the policies become more pronounced as dynamic requests are introduced, i.e. customer requests that appear throughout the planning horizon. When considering the fraction of new requests accommodated, it is clear that a conservative approach outperforms the other recharging policies as it most successfully strikes a balance between maintaining a reserve of battery level and keeping flexibility in the vehicle schedules. Furthermore, these results uphold when considering different degrees of dynamism (fraction of all requests that are dynamic) of the system, or when considering different lead times (how far in advance the requests are known). In the former case, the total fraction of accepted requests is lower when the degree of dynamism is higher, as one would expect intuitively. In the case of lead times, we see that longer lead times lead to a greater fraction of accepted requests for all policies, but that a conservative strategy appears more robust to shorter customer lead times than the other two policies.

As well as considering the service quality as a measure of system performance, the operational efficiency of vehicle routes obtained through the dynamic VNS/TS heuristic is considered, by considering the Value of Information (VoI) - the ratio of the dynamic and static total vehicle travel times for a given instance. The fact that the VoI increases in the degree of dynamism both when all dynamic requests were accepted and when using a proxy for VoI, suggests that the vehicle schedules are improved by having as many requests as possible known in advance. Also, the fact that the VoI values are above one suggest that the schedules made using advance requests are not rendered operationally inefficient in a dynamic setting when new requests appear during the planning horizon.

8 Discussion

For a dynamic DARP setting, this research adds to the body of existing literature by introducing and investigating the effectiveness of different heuristic recharging strategies on service quality and operational efficiency. It is found that a policy which balances battery reserve levels with schedule flexibility works best in an online setting. However, in all cases, each vehicle operates under the same recharging policy. In practice, if vehicles are homogeneous and customers are equally distributed across the vehicles, many vehicles will need to recharge at the same time, leading to potentially more new requests being rejected. Therefore, an interesting avenue for further research would be to consider a system where vehicles operate under different charging policies, in order to determine whether there are certain synergy effects to be obtained through the use of a range of policies amongst the fleet. Additionally, whether or not switching between different recharging rules during the planning horizon is an effective strategy (e.g. using a full recharging strategy when charge is above 50% and a conservative policy otherwise) would be an interesting area of exploration for further study too. Also, considering new charging policies which schedule visits to charging stations before such trips become necessary (to maintain sufficient battery levels throughout the planning horizon) may yield improved vehicle schedules in a dynamic environment.

Furthermore, this research considers a single scheduling policy, namely a dynamic adaptation of the scheduling rule suggested for the standard DARP by Cordeau and Laporte (2003), with the modification that departures are postponed as much as possible to allow vehicles flexibility in determining their next stop. However, it may be the case that particular scheduling rules are better suited to perform in conjunction with certain recharging rules. Therefore, further research into the use of heuristic approaches for the dynamic DARP with electric (autonomous) vehicles should focus on the joint performance of recharging and scheduling decisions.

In many dynamic VRP settings, one risk is that fruitful here-and-now decisions may ultimately lead to worse vehicle routes and schedules when new customer requests arrive. Without stochastic information about the whereabouts or timing of future requests, such problems can also be difficult to avoid. In this research, such a problem would manifest itself in lower service quality rates and less operationally efficient routes. One possible approach for circumventing these potential issues would be through the use of alternative cost functions. For example, instead of selecting customer insertions and modifications to vehicle schedules based on the total travel time alone (the current cost function), a cost function which rewards routes with greater battery reserve levels, time window, capacity and ride-time slack may help to better accommodate new requests as they emerge during the planning horizon. By steering the search procedure in the direction of solutions which do not necessarily cover the least distance, but increase the likelihood of being able to accommodate subsequent requests, service quality may be improved for all strategies. Of course, there is a risk that this would come at the expense of operational efficiency, yet this trade-off may be desirable depending on the preferences of the specific fleet operator.

Finally, in order to underline the findings in this research, the solution techniques should be applied to a wider range of problem characteristics. Particularly, problem settings with different recharge and discharge rates, vehicle battery levels, customer ride times and travel distances should be explored in order to determine whether the efficacy of different recharging strategies is sensitive to changes in the parameters of the problem.

9 References

Attanasio, A., Cordeau, J.-F., Ghiani, G., and Laporte, G. (2004). Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, *30*(3), 377–387.

Beaudry, A., Laporte, G., Melo, T., and Nickel, S. (2010). Dynamic transportation of patients in hospitals. *OR spectrum*, *32*(1), 77–107.

Berbeglia, G., Cordeau, J.-F., and Laporte, G. (2012). A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. *INFORMS Journal on Computing*, *24*(3), 343–355.

Bongiovanni, C., Kaspi, M., and Geroliminis, N. (2019). The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological*, *122*, 436–456.

Braekers, K., and Kovacs, A. A. (2016). A multi-period dial-a-ride problem with driver consistency. *Transportation Research Part B: Methodological*, *94*, 355–377.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, *54*(3), 573–586.

Cordeau, J.-F., and Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, *37*(6), 579–594.

Coslovich, L., Pesenti, R., and Ukovich, W. (2006). A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*, *175*(3), 1605–1615.

Daganzo, C. F. (1978). An approximate analytic model of many-to-many demand responsive transportation systems. *Transportation Research*, *12*(5), 325–333.

Desrosiers, J., Dumas, Y., and Soumis, F. (1986). A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, *6*(3-4), 301–325.

Detti, P., Papalini, F., and de Lara, G. Z. M. (2017). A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega*, *70*, 1–14.

Diana, M., and Dessouky, M. M. (2004). A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological*, *38*(6), 539–557.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European journal of operational research*, *54*(1), 7–22.

Feng, L., Miller-Hooks, E., Schonfeld, P., and Mohebbi, M. (2014). Optimizing ridesharing services for airport access. *Transportation Research Record*, *2467*(1), 157–167.

Garaix, T., Artigues, C., Feillet, D., and Josselin, D. (2010). Vehicle routing problems with alternative paths: An application to on-demand transportation. *European Journal of Operational Research*, *204*(1), 62–75.

Glover, F. (1996). Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, *65*(1-3), 223–253.

Goeke, D. (2019). Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. *European Journal of Operational Research*, *278*(3), 821–836.

Greenblatt, J. B., and Shaheen, S. (2015). Automated vehicles, on-demand mobility, and environmental impacts. *Current sustainable/renewable energy reports*, *2*(3), 74–81.

Gschwind, T., and Drexl, M. (2019). Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science*, *53*(2), 480–491.

Guerriero, F., Bruni, M. E., and Greco, F. (2013). A hybrid greedy randomized adaptive search heuristic to solve the dial-a-ride problem. *Asia-Pacific Journal of Operational Research*, *30*(01), 1250046.

Harper, C. D., Hendrickson, C. T., Mangones, S., and Samaras, C. (2016). Estimating potential increases in travel with autonomous vehicles for the non-driving, elderly and people with travel-restrictive medical conditions. *Transportation research part C: emerging technologies*, *72*, 1–9.

Hu, T.-Y., and Chang, C.-P. (2015). A revised branch-and-price algorithm for dial-a-ride problems with the consideration of time-dependent travel cost. *Journal of Advanced Transportation*, *49*(6), 700–723.

IEA. (2020). *Global ev outlook 2020.* Retrieved 2020-06-01, from `https://www.iea.org/reports/global-ev-outlook-2020`

Jaw, J.-J., Odoni, A. R., Psaraftis, H. N., and Wilson, N. H. (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological*, *20*(3), 243–257.

Keskin, M., and Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, *65*, 111–127.

Liu, M., Luo, Z., and Lim, A. (2015). A branch-and-cut algorithm for a realistic dial-a-ride problem. *Transportation Research Part B: Methodological*, *81*, 267–288.

Lois, A., and Ziliaskopoulos, A. (2017). Online algorithm for dynamic dial a ride problem and its metrics. *Transportation research procedia*, *24*, 377–384.

Luo, Y. (2006). *Heuristics and performance metamodels for the dynamic dial-a-ride problem* (Unpublished doctoral dissertation).

Luo, Y., and Schonfeld, P. (2011). Online rejected-reinsertion heuristics for dynamic multivehicle dial-a-ride problem. *Transportation research record*, *2218*(1), 59–67.

Maalouf, M., MacKenzie, C. A., Radakrishnan, S., and Court, M. (2014). A new fuzzy logic approach to capacitated dynamic dial-a-ride problem. *Fuzzy Sets and Systems*, *255*, 30–40.

Madsen, O. B., Ravn, H. F., and Rygaard, J. M. (1995). A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of operations Research*, *60*(1), 193–208.

Mitrovic-Minic, S., Krishnamurti, R., Laporte, G., et al. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, *38*(8), 669–685.

Muelas, S., LaTorre, A., and Peña, J.-M. (2013). A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city. *Expert Systems with Applications*, *40*(14), 5516–5531.

Paquette, J., Cordeau, J.-F., Laporte, G., and Pascoal, M. M. (2013). Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological*, *52*, 1–16.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers and Operations Research*, *37*(6), 1129–1138.

Parragh, S. N., Doerner, K. F., Hartl, R. F., and Gandibleux, X. (2009). A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks: An International Journal*, *54*(4), 227–242.

Parragh, S. N., Pinho de Sousa, J., and Almada-Lobo, B. (2015). The dial-a-ride problem with split requests and profits. *Transportation Science*, *49*(2), 311–334.

Potvin, J.-Y., and Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, *46*(12), 1433–1446.

Psaraftis, H. N. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, *14*(2), 130–154.

Psaraftis, H. N. (1983). An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation science*, *17*(3), 351–357.

Ropke, S., Cordeau, J.-F., and Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks: An International Journal*, *49*(4), 258–272.

Ropke, S., and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, *40*(4), 455–472.

Santos, D. O., and Xavier, E. C. (2015). Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications*, *42*(19), 6728–6737.

Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, *4*(2), 146–154.

Schilde, M., Doerner, K. F., and Hartl, R. F. (2011). Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers and operations research*, *38*(12), 1719–1730.

Schneider, M., Sand, B., and Stenger, A. (2013). A note on the time travel approach for handling time windows in vehicle routing problems. *Computers and Operations Research*, *40*(10), 2564–2568.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, *48*(4), 500–520.

Shaheen, S., and Cohen, A. (2019). Shared ride services in north america: definitions, impacts, and the future of pooling. *Transport reviews*, *39*(4), 427–442.

Swagerman, A. (2019). *Number of all-electric cars has doubled.* Retrieved 2019-05-10, from `https://www.cbs.nl/en-gb/news/2019/19/number-of-all-electric-cars-has-doubled`

Toth, P., and Vigo, D. (1997). Heuristic algorithms for the handicapped persons transportation problem. *Transportation science*, *31*(1), 60–71.

Toth, P., and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, *15*(4), 333–346.

Vallée, S., Oulamara, A., and Cherif-Khettaf, W. R. (2020). New online reinsertion approaches for a dynamic dial-a-ride problem. *Journal of Computational Science*, *47*, 101199.

Wong, K.-I., Han, A., and Yuen, C. (2014). On dynamic demand responsive transport services with degree of dynamism. *Transportmetrica A: Transport Science*, *10*(1), 55–73.

10 Appendices

Appendix A: Static Results

Tables 3-5 below present the objective value found within a period of 60 seconds, when running the VNS/TS search procedure using the full (F), minimal (M) and conservative (C) recharging strategies on the benchmark instances with different levels of vehicle charge (10%, 40% and 70% respectively) required at the destination depot. The optimality gaps of the values with respect to the optimal solution are presented for all instances for which this value is known.

Table 3: Objective values and optimality gaps for all benchmark instances with battery requirement $r = 0.1$

Policy	Full (F)		Minimal (M)		Conservative (C)		
Instance	Obj.	Gap (%)	Obj.	Gap (%)	Obj.	Gap (%)	BKS
`a2-16`	320.70	8.99	320.70	8.99	320.70	8.99	294.25
`a2-20`	348.07	0.94	348.07	0.94	348.07	0.94	344.83
`a2-24`	442.41	7.09	442.41	7.09	442.41	7.09	413.12
`a3-18`	300.63	0.05	300.63	0.05	300.63	0.05	300.48
`a3-24`	359.57	4.28	359.57	4.28	359.57	4.28	344.83
`a3-30`	523.04	5.70	523.04	5.70	523.04	5.70	494.84
`a3-36`	616.72	5.75	621.27	6.53	633.90	8.70	583.19
`a4-16`	282.68	0.00	282.68	0.00	282.68	0.00	282.68
`a4-24`	406.55	8.41	406.94	8.51	406.94	8.51	375.02
`a4-32`	504.17	3.85	504.17	3.85	504.17	3.85	485.5
`a4-40`	603.29	8.18	603.29	8.18	603.29	8.18	557.69
`a5-40`	514.79	3.29	514.79	3.29	514.79	3.29	498.41
Average		**4.71**		**4.78**		**4.96**	

Table 4: Objective values and optimality gaps for all benchmark instances with battery requirement $r = 0.4$

Policy	Full (F)		Minimal (M)		Conservative (C)		
Instance	Obj.	Gap (%)	Obj.	Gap (%)	Obj.	Gap (%)	BKS
`a2-16`	323.26	9.86	322.10	9.46	322.10	9.46	294.25
`a2-20`	353.31	1.60	349.44	0.49	350.28	0.73	347.73
`a2-24`	466.48	7.62	445.95	2.88	445.95	2.88	433.46
`a3-18`	300.63	0.05	300.63	0.05	300.63	0.05	300.48
`a3-24`	361.38	4.80	361.38	4.80	361.38	4.80	344.83
`a3-30`	546.98	10.54	526.82	6.46	541.39	9.41	494.84
`a3-36`	NA	NA	618.13	5.10	618.13	5.10	588.12
`a4-16`	282.68	0.00	282.68	0.00	282.68	0.00	282.68
`a4-24`	416.93	11.18	411.13	9.63	411.13	9.63	375.02
`a4-32`	507.94	4.54	504.17	3.76	507.94	4.54	485.9
`a4-40`	605.80	8.63	612.52	9.83	614.26	10.14	557.69
`a5-40`	513.49	3.02	509.11	2.15	509.11	2.15	498.41
Average		**5.62**		**4.55**		**4.91**	

Table 5: Objective values and optimality gaps for all benchmark instances with battery requirement $r = 0.7$

Policy	Full (F)		Minimal (M)		Conservative (C)		
Instance	Obj.	Gap (%)	Obj.	Gap (%)	Obj.	Gap (%)	BKS
`a2-16`	347.49	16.36	336.30	12.61	336.30	12.61	298.63
`a2-20`	368.61	NA	382.35	NA	382.35	NA	NA
`a2-24`	NA	NA	NA	NA	NA	NA	444.63
`a3-18`	316.72	4.28	306.29	0.85	307.88	1.37	303.71
`a3-24`	363.14	3.55	365.41	4.20	369.25	5.29	350.69
`a3-30`	NA	NA	NA	NA	NA	NA	NA
`a3-36`	NA	NA	NA	NA	NA	NA	599.75
`a4-16`	286.72	1.43	285.64	1.05	283.42	0.26	282.68
`a4-24`	429.71	12.58	422.57	10.71	414.16	8.51	381.7
`a4-32`	561.85	14.38	519.04	5.66	519.04	5.66	491.23
`a4-40`	NA	NA	597.26	NA	597.26	NA	NA
`a5-40`	541.70	2.55	563.29	6.64	563.29	6.64	528.21
Average		**7.88**		**5.96**		**5.76**	

Appendix B: Demand Distribution of Earliest Pickup Times

Figure 11 presents the demand distribution of the earliest pickup time for instance `a5-50`. For each hour of the planning horizon ($T = 600$), the number of requests whose earliest pickup times that fall within the given hour are reported.

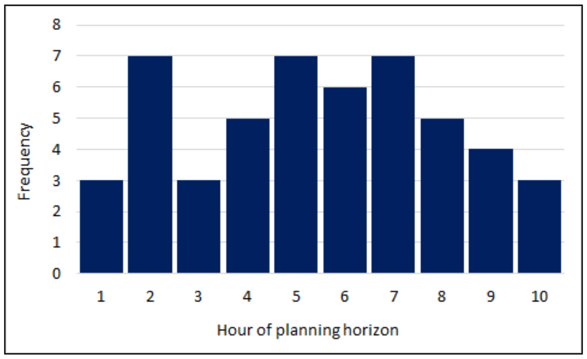


Figure 11: Distribution of earliest pickup times for each hour of the planning horizon for instance `a5-50`