Adaptive Large Neighborhood Search for the Static Electric Autonomous Dial-A-Ride Problem

Stavros Bakeas

29 May 2023

Abstract

The expected increase in electric autonomous vehicle(EAV) usage has sparked research into successfully adopting them in the context of ridesharing operations. This paper addresses the static version of the Electric Autonomous Dial-A-Ride problem (EA-DARP), which requires serving a number of transportation requests known in advance by a fleet of EAVs, while minimizing total excess ride time and operational costs. Each route must satisfy all the typical constraints found in the standard DARP, as well as not run out of battery and return to the ending depot with battery level higher than a lower bound. The vehicles are allowed to visit recharging stations to partially or fully charge, provided that they carry no passengers.

We propose an Adaptive Large Neighborhood Search(ALNS) algorithm enhanced with a heuristic procedure, which aids in solving highly constrained cases. The removal and insertion operators selected in each iteration depend on their performance in previous iterations. An insertion feasibility test has been designed in order to decrease computational time.

Computational experiments on benchmark instances were conducted in order to assess the performance of ALNS. Our algorithm managed to find the best known solution on four instances, improved the solutions of many and most importantly, found feasible solutions of previously unsolved instances. The results highlight the ability of the ALNS algorithm to tackle large highly-constrained instances.

Contents

1 Introduction

Vehicle electrification and the rapid development of self-driving technology are expected to transform urban mobility and transportation for the better. The elimination of the need of a driver means that the vehicles can be operated for longer periods of time and for many different purposes. Most importantly, the number of traffic accidents are expected to decrease, since the majority of them are caused by human misjudgement(Rojas-Rueda et al., 2020). Moreover, the use of shared electric autonomous vehicle(EAV) fleets in order to accommodate transportation requests could result in the reduction of traffic congestion and free street parking. As far as electric motors are concerned, fully electric vehicles do not emit any amount of carbon dioxide, which means that their widespread adoption could prevent a large percentage of air pollution related deaths (Pan et al., 2021). Lastly, electric vehicles are associated with less traffic related noise, which is related to health problems (Babisch et al., 2008, Van Kempen and Babisch, 2012).

Despite the potential environmental, societal and economical benefits, fully transitioning from conventional vehicles to EAVs remains a challenging task. According to the survey of Noel et al., 2019, the two main reasons consumers were not interested in purchasing an electric vehicle were the limited driving range and cost of ownership. The latter remains a source of discouragement for low income consumers, despite the subsidies provided by the government and different organizations(Caulfield et al., 2022). Additionally, from the standpoint of a ridesharing platform, the decision making process regarding the assignment of vehicles to requests and scheduling becomes even more complex, since the state of charge and slow charging speed must be taken into account.

A significant number of previous works have modeled a ridesharing service as the Dial-A-Ride problem(DARP). The standard version of the problem was formulated by Cordeau and Laporte, 2003. Users specify origin and destination locations. They also specify a time window at either the origin or the destination, during which service must begin. In order to account for user and driver inconvenience, the maximum ride time of each user and route duration of each vehicle is limited. In recent years, generalizations with vehicles having different characteristics and multiple depots have also been studied (Braekers et al., 2014, Masmoudi et al., 2016, Malheiros et al., 2021).

In this paper, we model a shared EAV service as the static Electric Autonomous Dial A Ride(EA-DARP) problem. The utilization of an electric autonomous fleet introduces additional constraints to the problem. The route duration constraint is removed, since the vehicles are autonomous and the battery resource constraint is introduced. A vehicle can visit a charging station in order to replenish its battery, provided that no passengers are on board. A partial recharging policy is adopted, which means that the fuel replenished by a vehicle in a charging station is a linear function of the duration of its stay. Moreover, the battery level upon return to the ending depot must exceed a certain lower bound. We assume that the ridesharing company does not own the vehicles, but has instead rented them for a specific amount of time. For that reason, the vehicles are not necessarily available from the beginning to the end of the planning horizon and they must return to a specific destination depot.

The contribution of this paper is threefold: Firstly, we have designed a test

which determines whether or not the insertion of a request in a route violates the time window, battery or capacity constraints of any node in the route. Secondly, an ALNS algorithm with problem-specific removal and insertion operators has been designed. The algorithm uses the insertion feasibility test in order to avoid checking ride time feasibility and calculation of the excess ride time of neighboring solutions. Regarding the charging decisions, each vehicle recharges just enough to reach the next station(ending depot) without running out (reaching battery level below the lower bound). Thirdly, the performance of the algorithm is tested on artificial and real-world instances. The algorithm manages to find the optimal solution for fifteen instances and improve solution quality for eleven instances. Moreover, feasible solutions were found for seven previously unsolved instances.

The rest of the paper is organized as follows: In Section 2, an overview of the literature is presented. Section 3 contains the notation used throughout the paper as well as a mathematical formulation of the problem. Section 4 describes the necessary steps that must be executed before ALNS, which is explained in detail in Section 5. The design and results of extensive computational experiments are presented in Section 7. Section 8 concludes the paper and suggests possible extensions and improvements.

2 Related Work

In the first subsection, we provide a brief review of the Electric Vehicle Routing Problem(EVRP) literature. Landmark studies on the field, as well as studies which use a metaheuristic are reported. In the second subsection, papers that studied the standard Dial-A-Ride problem are reported. In the third subsection, the papers which combine the Dial-A-Ride problem with electric vehicles are reported.

2.1 Electric Vehicle Routing Problem

Conrad and Figliozzi, 2011 were the first paper to integrate the recharging operation in the Vehicle Routing Problem(VRP) model. The vehicles have limited driving range and can either partially charge up to a predefined percentage of the maximum battery or fully charge. Charging takes place at specific customer locations. Erdoğan and Miller-Hooks, 2012 studied the Green VRP, where vehicles could visit a recharging station during their trip in order to recharge to full battery capacity. The authors proposed two construction heuristic and a local search method and reported their performance on small and large instances.

2.1.1 Full Recharge

A vast literature exists on the EVRP and its variants, much of it assuming that the vehicles either charge fully and/or that the charging time is constant. Lin et al., 2016 provided a non-linear and a linear mathematical formulation of the problem. The authors assumed that the charging time is constant, independent of the battery level with which the vehicle arrived at the station. The main objective was the minimization of the weighted sum of the total travel time and energy consumption, which depends on the vehicle load at each location. Shao

et al., 2018 studied the EVRP with a realistic energy consumption model dependent on vehicle speed, which is considered a discrete decision variable, and the load. The vehicles charge fully upon visiting a charging station and the charging time is fixed. The authors used a hybrid genetic algorithm to solve the problem. Zhang et al., 2018 studied a version of the EVRP with a realistic energy consumption model. The energy consumed in each arc depends not only on the distance, but on the current load and speed of the vehicle as well. It is assumed that the vehicles reach their full battery capacity instantly upon visiting a charging station, without staying there for a certain amount of time. The objective was the minimization of the total energy consumption. The authors provided two algorithm to solve the problem i) an ant colony optimization algorithm which uses iterated local search and ii) an ALNS algorithm coupled with a local search. The former found close-to-optimal solutions for small size problem instances and outperformed ALNS both in small and large instances. Moreover, they also demonstrated the importance of using the energy consumption objective function instead of the one with the total distance traveled.

Many works considered the real-world constraint of time windows on the beginning of service at each node, resulting in the EVRP with Time Windows (EVRPTW). The time window constraint complicates the charging decision process. Schneider et al., 2014 were the first to study the EVRPTW. The vehicles were assumed to charge to full battery capacity when visiting a station, with the charging time being linearly proportional to the charged amount. Moreover, an upper bound was imposed on the number of customers that can be on board simultaneously. The primary objective was to minimize the number of vehicles, whereas the secondary one was to minimize the total traveled distance. The authors proposed a heuristic combining Variable Neighborhood Search(VNS) and Tabu Search(TS), which found the optimal solution to newly generated instances and performed well when applied to solve instances of related problems. Hiermann et al., 2016 studied a version of the problem in which the fleet composition was included in the decision making process. The quality of each vehicle was measured by the maximum battery level, load capacity, energy consumed per unit of distance traveled and recharging time per unit of energy replenished. Using vehicles of high quality incurs higher costs. The authors provided a mixed integer programming and a set partitioning model and apply a branch-and-price algorithm to the latter to solve instances of small size optimally. The pricing problem was solved using a bi-directional labeling algorithm. In order to tackle instances of larger size, the authors developed an ALNS algorithm. After the destroy and repair step, a local search procedure is applied for intensification. Then, a labeling algorithm was applied with the hope that a better configuration with regard to the charging stations is discovered. In the labeling algorithm, it is assumed that only one charging station node can be inserted between two customer nodes in order to decrease computational time. Zhao and Lu, 2019 studied a real-world fleet size and mix EVRP with load constraints that seeks to minimize the sum of total travel cost, charging cost, acquisition cost and waiting time. The first three terms of the objective function depend on the type of vehicle selected for each route. The depot is the beginning and the end of each and every route, but it is also allowed to be visited for the purpose of the vehicles loading/unloading parcels and/or recharging. Due to the complexity of the problem, the authors provided an ALNS algorithm to solve the problem. In each

iteration, after the removal and reinsertion steps, local search is applied. Then, the vehicle types are selected and the optimal charging station and intermediate depot visits are found for a fixed customer sequence using a labeling algorithm. Finally, in order to decrease the waiting times, a heuristic which computes the optimal departure time of each route is used. Using all the feasible routes found throughout the execution of the ALNS, a set partitioning model is constructed and solved using a mixed integer programming solver, which runs until either the optimal solution of the model is found or a maximum time limit has been reached.

2.1.2 Partial Recharge

Always charging an electric vehicle to its maximum battery capacity is not the most efficient strategy to follow in a real-world scenario due to the prolonged amount of time it would take. Therefore, many works have considered recharging the vehicles partially. Bruglieri et al., 2015 solves the EVRP with time windows using a VNS algorithm with local branching. Keskin and Çatay, 2016 formulated the EVRP with time windows and recharging stations as a 0-1 mixed integer linear program and used an ALNS algorithm to solve it. The authors designed three types of operators: Those that insert or remove customers exclusively, those that insert or remove charging stations exclusively and those that remove a customer together with its preceding or succeeding charging station. After the routes are modified, the charging time at each station is adjusted so that the battery constraints are satisfied and the vehicle returns to the depot without any battery. Hiermann et al., 2019 studied a complex version of the VRP with time windows, in which the fleet is composed of electric(BEV), plug-in hybrid(PHEV) and conventional vehicles(ICEV). Using internal combustion engine(ICE) results in higher travel cost. Hybrid vehicles can switch to ICE mode if their battery level is depleted, but can also charge at the known charging station locations. The authors proposed a hybrid genetic algorithm to solve the problem. For the evaluation of each route, the charging station sequence is defined by a labeling algorithm and the scheduling, vehicle mode selection and charging level at each station by greedy heuristics. The current solution is either mutated using a large neighborhood search algorithm with random operator selection or two individuals are selected from the feasible solution pool and a crossover operator is applied. A set partitioning model is also solved periodically to find a good solution by recombining feasible routes found during the search. Finally, an iterative improvement procedure is applied to improve the solutions. Macrina, Pugliese, et al., 2019 solved the Green VRP with a mixed fleet of conventional and electric vehicles. Regarding the conventional vehicles, an upper bound on the polluting emissions was considered. The authors used an iterated local search algorithm, which consists of a perturbation and local search phase. Macrina, Laporte, et al., 2019 extended the previous work by considering a realistic energy consumption model with acceleration and deceleration. They proposed a large neighborhood search algorithm with a tabu criterion, whose initial solution is provided by using a commercial solver for a specified time limit. Recently, Wang and Zhao, 2023 introduced the fleet size and mix EVRP with time windows. They provided a path-based formulation, which supports multiple visits to the same recharging station without needing to create a copy of the station. In order to solve the problem, they provided an Iterated Local

Search Algorithm, which uses LNS to perturb the current solution and Variable Neighborhood Descent to improve it. The routes of the locally optimal solution are stored at each iteration and are recombined every 25 iterations by solving a set partitioning model, forming a potentially better solution than the best one found. The route pool is emptied after the model is solved.

The studies described so far assumed that the recharge rate, which is the amount of battery replenished for every unit of time spent on the station, was either not considered, meaning that the vehicles charged to full capacity instantly upon visiting a station, or was constant. Felipe et al., 2014 considered an EVRP where the charging stations supported multiple technologies. The recharge rate was considered a discrete decision variable. Each visit was associated with a variable cost, which depends on the selected recharge rate and a fixed cost. The objective was to minimize the total recharging cost, while serving all requests and satisfying route duration, load and battery constraints. The authors proposed three heuristics to solve the problem: A constructive heuristic, VNS and Simulated Annealing(SA). Keskin and Çatay, 2018 extended their previous work by considering three charging modes when visiting a station. Fast charging modes were associated with a higher cost. They aimed to minimize the total charging cost while operating the least amount of vehicles. They added new components to the ALNS algorithm described in Keskin and Çatay, 2016 and used an exact method throughout, which optimizes the charging decisions of a route.

2.2 Dial-A-Ride

2.2.1 Standard Version

The standard version of the Dial-A-Ride(DARP) problem was first introduced by Cordeau and Laporte, 2003. Users want to be picked up from one location(origin) and be dropped off at another location(destination). Service at either one or both of these locations must start during a time period specified by the user(time window). Each vehicle responsible for the accommodation of the requests has limited passenger capacity and cannot travel for more than a specified upper bound(route duration). Moreover, once the passengers of a request enter a vehicle, they cannot stay at the vehicle for more than a specified upper bound(maximum ride time). The objective is to construct a set of routes which serve all requests and satisfy the constraints, while minimizing the total distance traveled. The authors provided a tabu search heuristic, which uses an evaluation procedure to determine the feasibility of a new solution.

Many exact approaches have been proposed for this version of the problem, meaning that the algorithm is guaranteed to find the optimal solution. Cordeau, 2006 provided a mixed integer programming formulation of the problem and a branch-and-cut algorithm to solve small to medium sized instances to optimality. They use valid inequalities for related routing problems as well as create new ones specific to the standard DARP. Ropke et al., 2007 provided a branch-and-cut algorithm and two new two-index[1] formulations of the Pickup and Delivery Problem with Time Windows(PDPTW), which can be extended to the DARP

[1]There is no index that represents the vehicles in the decision variables of the formulation, which translates into lower running times and solution costs when exact methods are applied

by adding maximum ride time constraints. The authors also introduced three families of valid inequalities. After running the algorithm on the instances created by Cordeau and Laporte, 2003, it was revealed that their approach was more effective than the one proposed by Cordeau, 2006 due to the fact that the larger instances were also solved to optimality. Ritzinger et al., 2016 provided dynamic programming approaches based on a giant tour representation of the problem. Small instances are solved using an exact approach, whereas for larger instances a restricted approach is used, in which only a subset of all the states are selected when transitioning from one stage of the algorithm to the next. The selection is made using a function which considers global information. Both approaches were also embedded into a large neighborhood search framework in order to rearrange requests.

Due to the fact that the Dial-A-Ride problem is NP-Hard, larger instances cannot be solved to optimality in a reasonable amount of time. Consequently, many studies considered metaheuristic algorithms, which are not guaranteed to find the optimal solution, but are expected to produce solutions of high quality. S. N. Parragh et al., 2010 provided a Variable Neighborhood Search(VNS) algorithm. In each iteration, a random neighbor of the incumbent solution is generated and an intra-route local search procedure is applied to the neighbor if it is promising. If the new solution is better than the incumbent, the next iteration starts with the new solution and the first neighborhood. Otherwise, the next larger neighborhood is used. If the largest neighborhood has been reached, the search starts over with the smallest one. The authors used the scheme of Cordeau and Laporte, 2003 for route evaluation, but if a violation cannot be repaired or there are no more violations, the procedure stops. The authors compared their approach to the one of Cordeau and Laporte, 2003 and improved the solutions on some instances. Jain and Van Hentenryck, 2011 proposed a Large Neighborhood Search algorithm to solve the problem. Starting from an initial solution, a number of randomly selected assigned requests are removed. Then, a constraint-programming search algorithm tries to insert the removed requests and it returns the first solution found which serves all requests in a feasible manner. A worse solution has a small probability of being accepted. This procedure is repeated until the maximum number of iterations or a predefined time limit have been reached. Computational experiments on benchmark instances demonstrated the superiority of the proposed approach compared to a standard LNS algorithm, the Tabu Search of Cordeau and Laporte, 2003 and the VNS of S. N. Parragh et al., 2010 and also prove the importance of the probabilistic acceptance criterion. S. N. Parragh and Schmid, 2013 proposed a column generation approach. At first, the initial column pool and a feasible initial solution are created. Then, in each iteration, the linear relaxation of the restricted set covering problem is solved and VNS is applied to each route in order to add more columns to the pool. Periodically, Large Neighborhood Search is applied to the solution found by solving the restricted set covering problem for a given number of iterations. In each iteration of the LNS, the operators that will be applied are selected randomly. Häme and Hakula, 2013 proposed a Hyperlink-Induced Topic Search(HITS) algorithm in order to find a feasible solution for highly constrained instances. In order to create a route for one vehicle, depth-first search is applied, where the nodes that will be explored first are the ones with high hub score, which represents the number of nodes that can

be reached from that node. In order to reduce computational time, the number of backtracks are limited. Chassaing et al., 2016 proposed an evolutionary local search algorithm. A randomized constructive heuristic initializes the search and local search is applied. Then, due to the randomized nature of the mutation operators, several neighboring solutions are generated and local search is applied in each of them. The best neighbor is selected for the next iteration. Note that the probability of a local search operator being selected heavily depends on the number of times its application improved a solution in past iterations.

Another important issue is checking the feasibility of a route in a short amount of time. Hunsaker and Savelsbergh, 2002 proposed a heuristic which runs in linear time. A limit on the waiting time at each visited node is also considered. Cordeau and Laporte, 2003 proposed an exact eight-step scheme which runs in quadratic time in the worst case. Haugland and Ho, 2010 proposed an exact approach running in loglinear time, which is a modification of the heuristic of Hunsaker and Savelsbergh, 2002. Tang et al., 2010 proposed a quadratic time worst-case procedure to correct the algorithm of Hunsaker and Savelsbergh, 2002, which occasionally declared feasible routes as infeasible. Firat and Woeginger, 2011a transformed the problem of whether or not a route is feasible to a shortest path problem in a vertex-weighted interval graph, which can be solved in linear time. Chassaing et al., 2016 compared the approach of Firat and Woeginger, 2011b to the one of Cordeau and Laporte, 2003 and confirmed the theoretical superiority of the former through computational experiments. Gschwind and Drexl, 2019 created a feasibility test that checks whether the insertion of a request in a route satisfies the time window and ride time constraints in amortized constant time. The authors compared their test with the eight-step scheme used in Cordeau and Laporte, 2003. Both the procedures were used in an ALNS algorithm. Their proposed approach was proven to be faster.

Some papers studied the standard formulation of the problem, but integrated quality-of-service related metrics into the objective function to be minimized. Jorgensen et al., 2007 added terms such as excess ride time, waiting time with passengers on board and total work time. They also considered time window, ride time and route duration constraints as soft, meaning that their violation is allowed but penalized. A cluster-first route-second method was proposed, where the clustering phase is handled by a genetic algorithm and the routing phase by a nearest neighbor heuristic. Kirchler and Wolfler Calvo, 2013 aimed to minimize the weighted sum of all of the above objectives, as well as early arrivals and number of unassigned requests. They proposed a Granular Tabu Search algorithm. During the local search phase, the only moves that are considered are those with reduced cost lower than the granular threshold, which adapts throughout the search. The reduced cost information is derived by solving an assignment subproblem, which produces clusters of requests which are similar temporally and spatially. Results showed that the algorithm is capable of finding adequate solutions in a short amount of time.

2.2.2 Heterogeneous Vehicles and/or Multiple Depots

Many constraints and features that occur in real-world healthcare transportation services are not present in the standard formulation of the problem, re-

sulting in a large number of works including them. Rekiek et al., 2006 studied a DARP in the context of handicapped people transportation. Each vehicle is heterogeneous with regard to capacity and each one is available at different times. The goal is to maximize service quality while minimizing the number of vehicles used. A genetic algorithm coupled with local improvement heuristics was used to solve real-world instances. Melachrinoudis et al., 2007 provided a mixed-integer programming formulation for the transportation service of a health-care organization. Each request contains the trip from an origin location to a treatment center and from the treatment center back to the origin location. Moreover, the vehicles do not necessarily start and end their routes at the same location and they are heterogeneous in terms of passenger capacity. The goal was to minimize the weighted sum of total transportation cost and total user inconvenience, which is represented as the sum of excess ride time, early or late deliveries and late pickups. A commercial solver could not solve instances with four requests in a reasonable time period and tabu search was used. S. N. Parragh, 2011 modeled the transportation operations of the Austrian Red Cross as the Heterogeneous Dial-A-Ride problem(HDARP). Patients may need to use a stretcher or a wheelchair seat. Moreover, they may need to be accompanied by someone. Each vehicle supports four seat types: staff seat, patient seat, stretcher and wheelchair space. One patient type can be assigned to different seat types when needed, but not to all. For example, a regular patient can sit on the stretcher, but not on the wheelchair seat. The goal was to minimize the sum of the total distance traveled and the total waiting time with passengers on board. The authors provided a three-index and two-index formulation of the problem and conducted numerical experiments on medium-sized instances using a Branch & Cut algorithm and VNS algorithm. Braekers et al., 2014 extended the work of S. N. Parragh, 2011 by adding multiple depots in the formulation of the HDARP, resulting in the MD-HDARP. The authors proposed a Branch & Cut algorithm and an efficient Deterministic Annealing algorithm and also created new benchmark instances. Computational experiments demonstrated the effectiveness of both algorithms for solving both MD-HDARP and HDARP instances. Masmoudi et al., 2016 introduced the Multi-Depot Multi-Trip Heterogeneous DARP(MD-MT-HDARP), in which the driver's rest is taken into account. Each driver performs two trips, one before the lunch break at his/her depot and one afterwards. A coffee break is included in both trips. Three metaheuristics were proposed: Hybrid Bees with Deterministic Annealing, Hybrid Bees with Simulated Annealing and Adaptive Large Neighborhood Search. The algorithms performed well on the newly generated instances of the problem and also for instances of the variant without breaks. Detti et al., 2017 provided a mixed-integer programming formulation for a multi-depot heterogeneous DARP arising from a real-world health application. In the formulation, many different vehicle types exist and the patients declare their preference to be served by vehicles of either a specific type or a subset of types. Moreover, the utilization of a specific vehicle type incurs different fixed costs and costs per unit of distance traveled. The authors have defined a complex objective function, which includes the waiting time of the patients at healthcare facilities. Two algorithms were proposed: A Tabu Search and Variable Neighborhood Search. Malheiros et al., 2021 proposed a hybrid algorithm to solve the MD-HDARP introduced by Braekers et al., 2014. The algorithm combines iterated local search with a set partitioning approach. In order to reduce computational time, the authors im-

plemented three speedup mechanisms. Their approach was effective in solving both HDARP and MD-HDARP instances.

2.2.3 Special Variants

Häll et al., 2009 introduced an exact formulation for the Integrated Dial-A-Ride problem, in which a fixed schedule routing service(e.g. bus, train) can be used along with the drivers to carry out parts of the trips of the passengers. The formulation is strengthened by arc elimination, variable substitution and subtour elimination constraints. The commercial solver managed to solve instances of small size. S. Parragh et al., 2014 considered a version of the problem, in which transportation requests with more than one passenger can be split and be treated as different requests. Each request is associated with a monetary reward and the goal is to maximize the revenue, which is equal to the difference between the total reward gathered by served requests and the total routing costs. A branch-and-price algorithm was capable of solving small-to-medium sized instances. For larger instances, variable neighborhood search was used. Computational experiments demonstrated that the strategy of splitting is most effective when the requests are gathered around few locations. Masson et al., 2014 considered a version of the problem in which the passengers of a request may be picked up by one vehicle and dropped off by another. The transferring operation happens at designated nodes called transfer points. The authors used an ALNS algorithm to solve the problem. Since the routes are interdependent, evaluating the feasibility of a solution is equivalent to finding no negative length circuit in a distance graph, which represents a simple temporal problem. Many necessary and sufficient conditions are checked before the full evaluation in order to reduce computational time. Results suggested that the use of transfer points can lead to significant savings. Pfeiffer and Schulz, 2022 removed the time window and maximum ride time constraints from the standard formulation, but the minimization of the excess ride time is included in the objective function. The maximum deviation from the direct travel time is penalized so that no customer experiences an excessively long trip. The authors provided an ALNS algorithm as well as a dynamic programming algorithm.

2.3 Dial-A-Ride with Electric Vehicles

The first part includes previous studies which are based on the seminal work of Bongiovanni et al., 2019, whereas the papers included in the second part contain different formulations of the problem.

2.3.1 Travel Distance & Excess Ride Time

Bongiovanni et al., 2019 were the first to formulate the electric autonomous dial-a-ride problem(EADARP). The destination depot each vehicle must return to is considered a decision variable. Moreover, the battery level upon return to the destination depot must be greater than a percentage γ of the maximum. The authors presented a three-index and a two-index mathematical programming formulation for the problem. In the latter formulation, the index to identify each vehicle is not used, resulting in less decision variables. However, the vehicle-specific constraints need to be modeled implicitly. The objective was

the minimization of the weighted sum of total travel time and total user excess ride time. B& C failed to find feasible solutions for instances with 4 or more vehicles and more than 24 requests for all settings of γ (0.1,0.4 and 0.7). Based on Bongiovanni et al., 2019, Su et al., 2021 used a Deterministic Annealing Local Search algorithm, combined with an exact method for calculating the minimum excess user ride time for a route in linear time after preprocessing. They found feasible solutions for all previously unsolved instances tested in Bongiovanni et al., 2019 and found new best solutions for some solved ones, due to mistakes in the formulation of Bongiovanni et al., 2019). These mistakes were corrected by Su et al., 2022, which proposed a column generation approach, improving some heuristic solutions of Su et al., 2021. Lastly, Bongiovanni, Kaspi, et al., 2022 proposed a two-phase metaheuristic to solve the dynamic version of the problem. In the first phase, a greedy insertion algorithm is applied to accommodate new requests. In the second phase, the current routing plan is re-optimized by applying a machine-learning based Large Neighborhood Search algorithm(MLNS). The destroy-repair couple that is selected in each iteration is based on the prediction of a machine-learning model about its performance. The model is trained on a large simulation-produced data set. Their approach outperforms the classic LNS algorithms on small-scale instances. However, solving the prediction problem in each iteration is a computationally expensive task.

2.3.2 Other Objectives

Masmoudi et al., 2018 was the first work that considered a fleet of electric vehicles for the Dial-A-Ride problem. Whenever a vehicle does not have enough fuel to reach the next node, it must visit a station or the depot to swap its depleted battery with a fully charged one. Passengers are allowed to be in the vehicle during the swapping process. The objective is the minimization of the total energy consumption. The authors proposed three evolutionary algorithms enhanced with Variable Neighborhood Search. Hoche et al., 2020 introduce new constraints to the EADARP, where the charging stations can be used by one vehicle at a time and are available for specific time intervals. Moreover, the fuel consumed while each vehicle is searching for parking in a location is taken into account. The objective is the maximization of the total earnings, where the value of a request is equal to the number of passengers multiplied by the direct travel time from its origin to its destination. The authors solve the problem by creating an initial solution using a greedy heuristic and then improving the solution using simulated annealing local search. Ma et al., 2021 solve a variation of standard DARP with EAVs, where the vehicles swap their depleted batteries with fully charged ones in battery swapping stations. The novelty of their approach is that the speed with which the vehicle will traverse an arc is a discrete decision variable. The objective function is a weighted sum of the total travel distance, total travel time and total energy consumption, which is heavily dependent on the speed and load. The authors tackle the problem using an ALNS algorithm. After the reconstruction step, the battery swapping stations are reinserted and the optimal speed in each traveled arc is determined.

3 Problem Description

Our problem(EADARP) can be defined on a complete directed graph $G(V, A)$ where $V = \mathcal{P} \cup D \cup B^+ \cup B^- \cup S$ the set of vertices and $A = \{(u, v)|u, v \in V, u \neq v\}$ the set of arcs. Set $\mathcal{P} = \{1, 2, \ldots, n\}$ contains the origin nodes and $D = \{n+1, n+2, \ldots, 2n\}$ the destination nodes of n transportation requests. The i-th request is denoted as $(i, n+i), i \in \mathcal{P}, n+i \in D$ and it is associated with a maximum ride time $L^{\max}_{i,n+i}$, which represents an upper bound on the amount of time the passengers of the request are willing to travel for. Let $\mathcal{V}$ the set of available vehicles. Each vehicle $h \in \mathcal{V}$ has a maximum battery capacity $bat^h_{\max}$ and a maximum passenger capacity $Q^h_{\max}$. The rate of discharge is denoted as ξ_h. Let $o^+_h \in B^+$ the starting depot and $o^-_h \in B^-$ the ending depot of vehicle h. The vehicle must depart from o^+_h not before e_h and must arrive at o^-_h not after l_h, with battery level not lower than $\gamma \cdot bat^{max}_h$, where $0 \leq \gamma \leq 1$. Let S be the set of charging stations. Each station $s \in S$ is associated with recharge rate r_s. Note that a charging station cannot be visited by a vehicle with passengers on board. Each node $v \in V$ is associated with

- a load q_v, which represents the number of passengers to be picked up at v, with $q_v = 0, \forall v \in S \cup B^+ \cup B^-$ and $q_v > 0 \wedge q_{n+v} = -q_v \forall v \in \mathcal{P}$.
- a non-negative service duration d_v, which is the duration of the loading/unloading operation, with $d_v = 0, \forall v \in S \cup B^+ \cup B^-$ and $d_v > 0, \forall v \in \mathcal{P} \cup D$.
- a time window $[e_v, l_v]$, which represents the time period during which service must begin at v. For charging stations and depots, it is set to $[0, T]$, where T is the end of the planning horizon. For origin and destination nodes, it depends on the request type. The passengers of an *outbound* request want to travel from home to a destination. An *inbound* request represents a trip back home. If a request is inbound(outbound) the time window at its destination(origin) is set to $[0, T]$ and the origin(destination) is the *critical* vertex. In reality, a strict time window exists, which depends on the origin(destination) and will be defined in a latter section for the purpose of speeding up our algorithm.

Each arc $a \in V$, is associated with a travel distance/time t_a. Note that $L^{\max}_{i,n+i} \geq t_{i,n+i}, \forall i \in \mathcal{P}$, Each vehicle $h \in \mathcal{V}$ consumes $\beta^h_a = \xi_h \cdot t_a$ battery when traversing arc a. Note that all the above mentioned costs are symmetric and we assume that the triangle inequality holds. Let $x^h_{i,j}$ binary decision variable, representing whether vehicle $h \in \mathcal{V}$ traverses arc $(i, j) \in A$. Let b^h_v the time vehicle $h \in \mathcal{V}$ begins service at node $v \in V$,bat^h_v the battery level of h upon arrival at v and Q^h_v the number of passengers in h after departing from v. Let w^h_v be the waiting time of h at node v before the beginning of service. Let E^h_s the time h stays to refuel at charging station $s \in S$. The formulation, inspired by Bongiovanni et al., 2019, is the following:

$$\min \lambda_1 \sum_{h \in \mathcal{V}} \sum_{i,j \in V} x^h_{i,j} t_{i,j} + \lambda_2 \sum_{i \in \mathcal{P}} R_i \tag{1}$$

subject to:

$$\sum_{i \in \mathcal{P} \cup S \cup \{o_h^-\}} x^h_{o_h^+,i} = 1, \forall h \in \mathcal{V} \tag{2}$$

$$\sum_{i \in D \cup S \cup \{o_h^+\}} x^h_{i,o_h^-} = 1, \forall h \in \mathcal{V} \tag{3}$$

$$\sum_{h \in V} \sum_{i \in D \cup S \cup \{o_+^h\}} x^h_{i,j} \leq 1, \forall j \in S \tag{4}$$

$$\sum_{\substack{j \in V \\ j \neq i}} x^h_{i,j} - \sum_{\substack{j \in V \\ j \neq i}} x^h_{j,i} = 0, \forall h \in \mathcal{V}, i \in \mathcal{P} \cup D \cup S \tag{5}$$

$$\sum_{h \in \mathcal{V}} \sum_{\substack{j \in \mathcal{P} \cup D \\ j \neq i}} x^h_{i,j} = 1, \forall i \in \mathcal{P} \tag{6}$$

$$\sum_{\substack{j \in V \\ j \neq i}} x^h_{i,j} - \sum_{\substack{j \in V \\ j \neq n+i}} x^h_{j,n+i} = 0, \forall h \in \mathcal{V}, i \in \mathcal{P} \tag{7}$$

$$b^h_{o_h^+} \geq e_h, \forall h \in \mathcal{V} \tag{8}$$

$$b^h_{o_h^-} \leq l_h, \forall h \in \mathcal{V} \tag{9}$$

$$b^h_i + d_i + t_{i,n+i} \leq b^h_{n+i}, \forall h \in \mathcal{V}, i \in \mathcal{P} \tag{10}$$

$$e_i \leq b^h_i \leq l_i, \forall h \in \mathcal{V}, i \in V \tag{11}$$

$$b^h_{n+i} - (b^h_i + d_i) \leq L^{\max}_{i,n+i} \tag{12}$$

$$R_i \geq b^h_{n+i} - (b^h_i + d_i) - t_{i,n+i}, \forall h \in \mathcal{V}, i \in \mathcal{V} \tag{13}$$

$$b^h_i + d_i + t_{i,j} - M_{i,j} \cdot (1 - x^h_{i,j}) \leq b^h_j, \forall h \in \mathcal{V}, i \in V, j \in V, i \neq j \tag{14}$$

$$Q^h_i + q_j + G^h_{i,j} \cdot (1 - x^h_{i,j}) \geq Q^h_j, \forall h \in \mathcal{V}, i \in V, j \in V, i \neq j \tag{15}$$

$$Q^h_i + q_j - G^h_{i,j} \cdot (1 - x^h_{i,j}) \leq Q^h_j, \forall h \in \mathcal{V}, i \in V, j \in V, i \neq j \tag{16}$$

$$Q^h_i \geq \max(0, q_i), \forall h \in \mathcal{V}, i \in \mathcal{P} \cup D \tag{17}$$

$$Q^h_i \leq \min(Q^h_{\max}, Q^h_{\max} + q_i), \forall h \in \mathcal{V}, i \in \mathcal{P} \cup D \tag{18}$$

$$Q_i^h = 0, \forall h \in \mathcal{V}, i \in B^+ \cup B^- \cup S \tag{19}$$

$$bat^h_{o^h_+} = bat^h_{\max}, \forall h \in \mathcal{V} \tag{20}$$

$$bat^h_i - \beta_{i,j} + bat^h_{\max} \cdot (1 - x^h_{i,j}) \geq bat^h_j, \forall h \in \mathcal{V}, i \in V \backslash S, j \in V \backslash \{o^h_+\}, i \neq j \tag{21}$$

$$bat^h_i - \beta_{i,j} - bat^h_{\max} \cdot (1 - x^h_{i,j}) \leq bat^h_j, \forall h \in \mathcal{V}, i \in V \backslash S, j \in V \backslash \{o^h_+\}, i \neq j \tag{22}$$

$$bat^h_s + E^h_s \cdot r_s - \beta_{s,j} + bat^h_{\max} \cdot (1 - x^h_{s,j}) \geq bat^h_j, \forall h \in \mathcal{V}, s \in S, j \in \mathcal{P} \cup B^- \cup S, s \neq j \tag{23}$$

$$bat^h_s + E^h_s \cdot r_s - \beta_{s,j} - bat^h_{\max} \cdot (1 - x^h_{s,j}) \leq bat^h_j, \forall h \in \mathcal{V}, s \in S, j \in \mathcal{P} \cup B^- \cup S, s \neq j \tag{24}$$

$$bat^h_s + E^h_s \cdot r_s \leq bat^h_{\max}, \forall h \in \mathcal{V}, s \in S \tag{25}$$

$$bat^h_{o^h_-} \geq \gamma \cdot bat^h_{\max}, \forall h \in \mathcal{V} \tag{26}$$

$$E^h_s \leq b^h_i - t_{s,i} - b^h_s + M^h_{s,i}(1 - x^h_{s,i}), \forall s \in S, i \in \mathcal{P} \cup S \cup \{o^h_-\}, h \in \mathcal{V}, i \neq S \tag{27}$$

$$E^h_s \geq b^h_i - t_{s,i} - b^h_s - M^h_{s,i}(1 - x^h_{s,i}), \forall s \in S, i \in \mathcal{P} \cup S \cup \{o^h_-\}, h \in \mathcal{V}, i \neq S \tag{28}$$

$$x^h_{i,j} \in \{0, 1\}, \forall h \in \mathcal{V}, i \in V, j \in V \tag{29}$$

$$bat^h_i \geq 0, \forall h \in \mathcal{V}, i \in V \tag{30}$$

$$E^h_s \geq 0, \forall h \in \mathcal{V}, s \in S \tag{31}$$

The objective function (1) minimizes the weighted sum of total travel distance and total excess user ride time, which is defined in (13) as the difference between the actual ride time of the request and the direct travel time from its origin to the destination. Constraints (2),(3) ensure that the route of each vehicle will start from the origin depot and end at the destination depot. In case, for a vehicle $h \in \mathcal{V}$, $x^h_{o^+_h, o^-_h} = 1$, we assume the vehicle is not being utilized. Constraint (4) ensures that each charging station is visited exactly once. Flow conservation is

ensured by constraint (5). Constraint (6) ensures that each origin location is visited exactly once and constraint (7) ensures that the origin and destination locations of a request are visited by the same vehicle.

Constraints (8),(9) define the time period during which each vehicle is available to serve requests. We refer to this time period as the *shift* of the vehicle. Constraint (10) ensures that the origin of each request is visited before its destination. Constraint (11) limits the time period during which service can begin at each location and constraint (12) ensures that the maximum travel time of each request is not exceeded. Constraint (14) defines the start of service at each node, with $M_{i,j} = \max(0, l_i + d_i + t_{i,j} - e_j)$.

Constraints (15), (16) define the number of passengers on board at each visited node, with $G^h_{i,j} = \max(Q^h_{\max}, Q^h_{\max} + q_i)$. The minimum and maximum number of passengers on board are defined in constraints (17) and (18) respectively. Constraint (19) ensures that no passengers are on board when a vehicle the depot or charging stations.

Equality (20) ensures that the vehicles are full in terms of battery when their shift starts. Constraints (21)-(24) define the battery level at each node. Constraint (25) ensures that the vehicle cannot charge beyond its maximum capacity and constraint (26) sets a lower bound on the battery level of the vehicle upon its return to the ending depot. Constraints (27), (28) set upper and lower bounds for each vehicle on the duration of the charging operation at each station, with $M^h_{s,i} = \min(\frac{bat^h_{\max}}{r_s} - e_i + t_{s,i} + T, T)$ and T denoting the end of the planning horizon. Lastly, constraints (30) and (31) ensure that the battery level at all nodes and the charging time at stations are positive.

The formulation of Bongiovanni et al., 2019 has been customized in order to agree with our assumptions. The owner of the vehicle has defined a specific location to which he/she wants his vehicle to be returned to, meaning that the vehicle cannot simply return to the depot closest to its last visited location. Moreover, each person uses their vehicle at different times during the day. Consequently, the time period and duration for which each vehicle is available to serve transportation requests is also different. Note that the above formulation supports only one visit to each charging station. In order to allow another visit to each station, we would create another set S', which includes copies of the original charging stations, and the set $\overline{S} = S \cup S'$ would be used in the formulation.

The solution of a problem instance with two vehicles and three requests is illustrated in Figure 1. The origin depot of the first vehicle is also its destination depot, whereas for the second vehicle, the origin and destination depots are different. Assuming that each vehicle can travel up to 15 units of distance fully charged, the second vehicle charges in a charging station, once it serves the first requests, in order to be able to complete its scheduled route.

4 Preprocessing

This step takes place before the execution of the solution algorithm. It aims to narrow the feasible search space and thus reduce the computational time without sacrificing solution quality. The rules described in detail in Sections 4.1 and 4.2, namely time window tightening and arc elimination, were first proposed

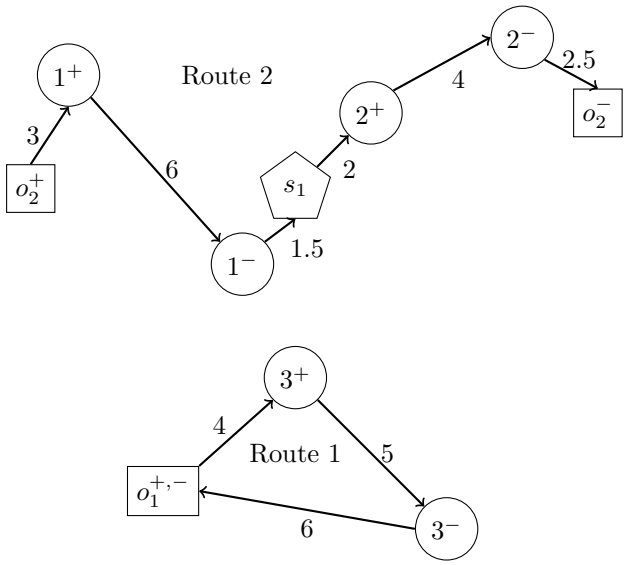


Figure 1: Problem Example

by Cordeau, 2006 for the purpose of the standard DARP and also used by Ma et al., 2021 for the EADARP.

4.1 Time Window Tightening

Let request $i \in \mathcal{P}$. If i is inbound, we set $e_{n+i} = \max\{0, e_i + d_i + t_{i,n+i}\}$ and $l_{n+i} = \min\{T, l_i + d_i + L^{\max}_{i,n+i}\}$. Otherwise, we set $e_i = \max\{0, e_{n+i} - d_i - L^{\max}_{i,n+i}\}$ and $l_i = \min\{T, l_{n+i} - d_i - t_{i,n+i}\}$. Using the time window information of the critical vertex, we can find the earliest and the latest time the vehicle can begin service at the non-critical vertex.

One could argue that the shift of each vehicle $h \in \mathcal{V}$ could also be shortened by setting $e_h = \max\{e_h, \min_{i \in \mathcal{P} \cup D}\{e_i - t_{o_h^+, i}\}\}$ and $l_h = \min\{l_h, \max_{i \in \mathcal{P} \cup D}\{l_i + d_i + t_{i, o_h^-}\}\}$. This step is not executed, due to the fact that shortening the vehicle shift leaves less time for the vehicle to charge, which would cause the evaluation scheme to incorrectly declare a route infeasible.

4.2 Arc Elimination

In this step, arcs that cannot be included in a feasible solution are identified. Let $I \subset A$ the set of all such arcs. An arc $(u, v) \in I$ if:

- $u \in B^+, v \in D$.
- $u \in \mathcal{P}, v \in B^-$.
- $u \in \mathcal{P}, v \in S$
- $u \in S, v \in D$
- $u \in D, v \in \mathcal{P} : v = u - n$.
- $u, v \in \mathcal{P} \cup D : |q_u + q_v| > \max_{h \in \mathcal{V}}(cap_h^{max})$
- $u, v \in V : e_u + d_u + t_{u,v} > l_v$

- arcs $(i,j),(j,n+i), i \in \mathcal{P}, j \in V : t_{i,j} + d_j + t_{j,n+i} > L^{\max}_{i,n+i}$

4.3 Infeasible Assignments

The assignment of request $i \in \mathcal{P}$ to vehicle $h \in \mathcal{V}$ is infeasible in terms of user inconvenience, if h departs from the starting depot the earliest time possible and the request is still served late.

$$\mathcal{V}_i^{\mathcal{R}} = \{h \in \mathcal{V} : (e_h + t_{o_h^+,i} > l_i) \vee (e_h + t_{o_h^+,i} + d_i + t_{i,n+i} > l_{n+i})\}$$

Similarly, the assignment of $i \in \mathcal{P}$ to vehicle $h \in \mathcal{V}$ is infeasible with respect to the vehicle return time constraint if, despite serving i as early as possible, h arrives at the ending depot late.

$$\mathcal{V}_i^{\mathcal{V}} = \{h \in \mathcal{V} : \max(e_i + d_i + t_{i,n+i}, e_{n+i}) + d_{n+i} + t_{n+i,o_h^-} > l_h\}$$

5 Adaptive Large Neighborhood Search

Large Neighborhood Search(LNS) was first proposed by Shaw, 1997 for solving different vehicle routing problems. Later, LNS reappeared under the name *Ruin and Recreate* by Schrimpf et al., 2000. The algorithm is characterized by a destruction step, in which certain elements are removed from the solution, and a reconstruction step, in which the removed elements are reinserted in a different way, resulting in a different solution than the original one. In LNS, a unique operator is used in each step. The adapted version of the algorithm(ALNS) was proposed by Ropke and Pisinger, 2006 for the Pickup and Delivery Problem with Time Windows. The difference between ALNS and its predecessor is the use of multiple removal and insertion operators. Moreover, an operator is more likely to be selected if it has performed well in previous iterations. The pseudocode of our proposed approach and the subroutines used is present in the appendix.

5.1 Insertion Neighborhood

Algorithm 1 is used as a subroutine in the construction of the initial solution and the reconstruction step of the LNS. It returns a set of moves that assign a request to a solution. The moves are feasible with respect to the battery,capacity and time window constraints. The insertion of the request is tested in the routes of all vehicles which have not been declared infeasible in the preprocessing step(Lines 2-6). The increase in charging time at a station visited anytime before the request is included in the move, as well as the insertion of an additional charging station. If, for a given route, no battery-feasible insertion is found, the charging station that is reachable and gives the minimum detour is inserted after a zero-load node and the insertion of the request is tested again(Lines 7-23).

5.2 Initial Solution

The initial solution is created as follows: First, all requests are sorted in non-decreasing order based on the earliest time their origin node can be visited. Then, they are sequentially inserted at their best positions. If a request cannot be feasibly inserted, it is added in the rejected requests pool.

5.3 Solution Evaluation

During the search, solutions that do not include all requests are allowed but penalized. For each rejected request, the penalty is equal to the average travel distance of all feasible arcs multiplied by the total number of requests. The penalty value is sufficiently large in order to guide the search towards solutions that include all requests.

5.4 Destruction Step

We use the Worst Removal, Random Removal and a modified version of Relatedness Removal operators from Ropke and Pisinger, 2006 as well as an additional one, which is inspired by the zero-split neighborhood proposed by S. N. Parragh et al., 2010 for the standard DARP. These operators remove ζ requests from the solution, where ζ depends on the percentage ω of the assigned requests that will be removed. Note that, after the destruction step has been executed, charging stations that are not used in a route are also removed.

Random Removal

This operator selects ζ requests randomly and removes them from the solution. It favours diversification and has good synergy with the rest of the removal operators.

Worst Removal

First, all assigned requests are sorted in non-increasing order based on their contribution to the objective function value of the current solution. The selection of the requests to be removed is controlled by a randomness parameter $p \geq 1$, providing balance between diversification and intensification. If $p = 1$, random removal takes place. If $p \to \infty$, then the worst ζ requests are selected for removal every time the operator is used.

Worst Zero Split Removal

A zero-split segment(Su et al., 2022) is a sequence of nodes inside a route, where the vehicle carries no passengers upon arrival at the first node ,upon departure from the last node and is never empty in between. In Figure 2, an example with a route serving three requests is illustrated. The first segment consists of two requests, while the second segment involves only one request.

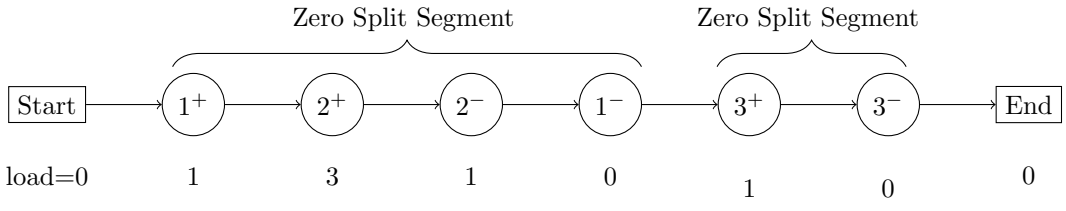


Figure 2: Example of a zero-split segment

This operator removes zero-split segments in a similar fashion to Worst Removal until a sufficient number of requests has been removed. Note that there is a possibility of this operator removing more than ζ requests. The cost of a zero-split segment is defined as the difference between the beginning of service at the last and first node. By removing long zero-split segments, the rearrangement of the requests might produce better solutions with respect to the total excess ride time.

Similarity Removal

It was first proposed by Shaw, 1997 and also used in Ropke and Pisinger, 2006. By removing similar requests, it is easy to shuffle them around in the reconstruction phase and better solutions might be found. The similarity measure $R_{i,j}$ between two requests i, j is defined as:

$$\begin{aligned} R_{i,j} = \Phi_1(t_{i,j} + t_{i+n,j+n}) + \\ \Phi_2(|e_i - e_j| + |l_i - l_j| + |e_{i+n} - e_{j+n}| + |l_{i+n} - l_{j+n}|) + \\ \Phi_3|q_i - q_j| \end{aligned}$$

The lower $R_{i,j}$ is, the more similar two requests are. Each term measures similarity in distance, time windows and load respectively and it is normalized so that $0 \leq R_{i,j} \leq \sum_{i=1}^{3} \Phi_i$, where $\Phi_i \in \mathbb{R}^+$ is the coefficient of the i-th term. In order to make the contribution of each term to the value of $R_{i,j}$ equal, we scale the term to take on values in the range $[0, 1]$. A key difference between our definition and the one in Ropke and Pisinger, 2006 is that the time windows of the origin and destinations are being compared instead of the service times, which means that the similarity measure between any two requests can be calculated in a preprocessing step and retrieved in constant time. The procedure of removing requests is the following: At first, a request r is chosen randomly, either from the assigned requests (if there are no unassigned ones) or from the removed pool. Then, the remaining assigned requests are sorted in non-decreasing order based on their similarity to r and ζ of them are removed. Again, the removal process is controlled by a randomness parameter.

5.5 Reconstruction step

In this step, we attempt to insert the requests removed from the destruction step as well as the rejected requests back into the current solution. The Greedy Insertion and Regret Insertion operators from Ropke and Pisinger, 2006 are employed together with a newly proposed operator.

Greedy Insertion

For each unassigned request, the cost of inserting it at its best position is found. At each iteration, the request with the least cost is inserted at its best position. In order to avoid unnecessary computations, the heuristic is composed of two phases. In the first phase, for each unassigned request, we store its best insertion position for each route, using an associative array. After the first insertion is completed, we do not create the associative array of the remaining requests from the beginning. The best insertion position for a route is updated in two cases:

i) The route was modified by the insertion and ii) The position is associated with a charging station that is already used in the solution.

Regret-k Insertion

This operator tries to improve upon the myopic insertion by considering the consequences of delaying the insertion of a request. At each iteration, the request with the highest regret value is inserted at its best position. Let $\theta_{r,k}$ the route for which request r has the k-th lowest insertion cost, denoted as $c_{r,\theta_{r,k}}$. Then, the regret value c_r^* for request r is equal to $\sum_{j=1}^{k}(c_{r,\theta_{r,j}} - c_{r,\theta_{r,1}})$. In case two requests have the same regret value, then the one whose insertion cost at its best position is lowest will be selected. If this operator is selected, k takes a random value between 2 and the number of available vehicles. The heuristic uses the same speed up mechanism used in Greedy Insertion.

Random Insertion

This operator encourages diversification and can help the algorithm escape local optima. A random unassigned request is selected and it is inserted at a random feasible position. If no feasible position is found for that request, it is added in the rejected pool.

5.6 Operator Selection Scheme

The procedure is similar for both destroy and repair operators, hence the notation that separates them is omitted. Let W_i the weight of operator i. The probability of selecting i is

$$p_i = \frac{W_i}{\sum_{j \in O} W_j}$$

where O is the set of available operators. The procedure is similar for both destroy and repair operators. Every N_W iterations, the weights are dynamically updated as follows:

$$W_i = \begin{cases} (1-a) \cdot W_i + a \cdot \frac{\Psi_i}{\upsilon_i}, \text{ if } \upsilon_i > 0 \\ W_i, \text{ otherwise} \end{cases}$$

where a is a control parameter that dictates the sensitivity of the weights in performance changes, Ψ_i the accumulated score of operator i and υ_i denotes the number of times i was selected. For the first N_W iterations, every removal and insertion operator has equal probability of being selected. The score Ψ_i is calculated as follows: If i was selected, ψ is added to Ψ_i, where:

$$\psi = \begin{cases} \psi_1, \text{ if the modified solution is the best one found} \\ \psi_2, \text{ if the modified solution was better than the current one} \\ \psi_3, \text{ if the modified solution was accepted} \\ 0, \text{ otherwise} \end{cases}$$

Ψ_i and υ_i are set to zero after the adjustment of the weights.

5.7 Rejected Reinsertion

If, after the reconstruction phase, there are still some requests that could not be feasibly inserted in the solution, Algorithm 7 is executed, which is inspired by Luo and Schonfeld, 2007. The similarity measure from Section 5.4 is used. The rejected request is swapped with its most similar assigned one. If the new route is feasible, the insertion of the most similar request is tested in all positions and it is inserted at the first feasible one found. If no feasible insertion is found, the solution returns to its original state and the rejected request is swapped with its second most similar request and so on. If all assigned requests have been tested and no feasible solution has been found, the request is finally rejected. This procedure is executed for all rejected requests and they are selected in the order by which they were tested for insertion in the previous step.

5.8 Acceptance Criterion

A new solution is accepted if it is cheaper with respect to the objective function than the incumbent. Otherwise, the new solution is accepted with probability $exp(\frac{\Delta}{T})$, where Δ the difference between the cost of the incumbent and the new solution and T the current temperature. Acceptance of non-improving solutions helps the algorithm escape local optima during the search. Similar to Ropke and Pisinger, 2006, the initial temperature T_{init} is set in such a way that the probability of accepting a solution $z\%$ worse than the first complete one is 50%. The temperature is gradually decreased according to a geometric cooling schedule. In each iteration, the current temperature is multiplied by the cooling rate $\rho \in [0, 1)$. The cooling rate ρ is set in such a way that, at the last iteration, the temperature is equal to 0.01.

6 Route Evaluation

In the reconstruction step, a set of possibly feasible insertions are found using Algorithm 1. However, neither the cost of each move has been calculated due to the presence of excess ride time in the objective function nor has it been confirmed that the insertions are feasible with respect to the maximum ride time constraint. In Section 6.1, the way in which the charging time is set at each visited station of a route is described in detail. In Section 6.2, the procedure which narrows down the number of feasible insertions is described in detail. In Section 6.3, we explain the way in which ride time feasibility is checked.

6.1 Charging Policy

The procedure described is repeated for all charging stations in the route in the order they were visited. Let s be the selected charging station and let s_{min} be the minimum battery level that the vehicle must depart from s with, in order to reach the next charging station (or ending depot if another visit to a station has not been scheduled) without violating the battery constraints. If s_{min} is less than the current battery level at s, no charging time is needed at that station. Otherwise, the charging time is set as the difference between s_{min} and the current battery level at s, divided by the recharge rate. The battery level at all subsequent nodes is updated.

6.2 Insertion Feasibility

In lines 6 and 13 of Algorithm 1, the insertion of a request to all positions of a route is tested and only feasible insertions regarding battery, capacity and time window constraints are appended to the move set. Let request r and feasible route $\theta = (v_1, \ldots, v_i, \ldots, v_j, \ldots, v_k)$, where v_1 is the starting and v_k the ending depot. We need to determine if the insertion of the origin r^+ after v_i and the destination r^- after v_j in θ results in another route θ', which might be infeasible only for the maximum ride time constraints. Obviously, the charging policy affects the check on both the time windows and the battery constraints. Algorithm 10 helps reduce the number of moves to be evaluated by the procedure of Section 6.3. At first, we check if any of the arcs added to the solution were eliminated in the preprocessing step(Lines 1-6). If not, the procedures described below are executed in the presented order. Note that if the insertion is deemed infeasible by any of the procedures, the algorithm terminates.

Capacity

If the sum of the number of passengers of request r and the load in any node between v_i and v_j exceeds the maximum passenger capacity of the vehicle, the insertion is infeasible. Note that if any charging station exists between nodes v_i and v_j, the insertion is also infeasible. This test can be performed in $O(k)$ in the worst-case, where k is the number of nodes in the route.

Time Windows

The time window feasibility test presented in Algorithm 11 is an extended version of the one used by Gschwind and Drexl, 2019. The concept of forward slack time proposed by Savelsbergh and Sol, 1995 for the Pickup and Delivery Problem with Time Windows(PDPTW) is used, which is basically the standard DARP without the ride time constraint. The vehicle notation is omitted. The formula is:

$$F_i^{\text{PDPTW}} = \min_{i \leq j \leq n} \{w_{i,j} + (l_i - b_i)\} \tag{32}$$

where $w_{i,j} = \sum_{x=i+1}^{j} w_x$ the cumulative waiting time between nodes $i+1$ and j. F_i represents the maximum amount of time by which the beginning of service can be delayed in i, such that i and all subsequent nodes are served before or at the end of their time window. Equation (32) can be rewritten as:

$$F_i^{\text{PDPTW}} = \begin{cases} l_h - b_i, \text{ if } i = k \\ \min\{l_i - b_i, w_{i+1} + F_{i+1}^{\text{PDPTW}}\}, \text{ otherwise} \end{cases} \tag{33}$$

The charging policy implies that, if a charging station precedes the origin(r^+) of the new request(Line 2), the insertion may delay the start of service at nodes that are visited after the station and we check if the nodes are served on time(Lines 3-10). If that is the case, we check if all the nodes between r^+ and r^-, which are included, are served on time(Lines 11-25). Lastly, we check that all the nodes visited after r^- are served on time(Line 27). A special case occurs when, despite the extra battery consumed, the closest preceding charging station of r^+ remains unnecessary(Line 29). In that case, the charging time

might increase at charging stations visited after r^- and the necessary checks are performed(Lines 30-36). If the cumulative waiting times are known (i.e. the sum of the waiting times between every two nodes), this test can be performed in constant time with a preprocessing step which would take $O(k^2)$ time. However, preliminary experiments suggested that no significant benefit in terms of computational time was provided. Only the forward time slack at every node has been calculated before the test is performed, which takes $O(k)$ time.

Battery

Let $\beta_{r,i,j}$ the added battery consumption caused by the insertion. The pseudocode for the test is presented in Algorithm 9. If a charging station precedes the origin of the request, the charging time will be increased. The insertion may violate the battery constraint only if the vehicle has reached its full battery capacity at the closest preceding station (Line 10). In case only the starting depot precedes the origin(Line 2), the new battery level is checked at the closest subsequent charging station or the ending depot(Lines 4-7). Since the order in which the nodes of a route are visited has already been computed, this test is performed in $O(|S|)$ time.

6.3 Scheduling Subproblem

If the maximum ride time constraint and excess ride time objective were not present in the formulation, the optimal strategy to ensure that time window constraints are satisfied would be to depart from the origin depot and begin service at each node as early as possible. However, this strategy can declare a route infeasible in terms of ride time constraints, while in reality the route would be feasible if a different strategy was used. An example is illustrated in Figure 3. A route that serves two requests is displayed. Above each arc of the route, its travel distance is displayed. For the sake of brevity and simplicity, we assume that the route satisfies capacity and battery constraints and that the service duration at each node is zero. Both requests are inbound and as a consequence the time windows at their destinations are open. The maximum ride time for the passengers of the first request is equal to 15. Below the route, two schedules are presented. The first schedule is generated by serving each node as soon as possible. The vehicle arrives at the origin of the second request early, so it has to wait for 4 units of time, while the passengers of the first request are on board. Despite the fact that the time window constraints are satisfied, the maximum ride time constraint for the first request is violated, since $23 - 5 = 18 > L_1^{max} = 15$. In the second schedule, the vehicle departs from the origin depot 5 units of time later. The vehicle arrives at the origin of the first request at the latest possible time, yet still respecting the time window. The time window at the origin of the second request is also respected. In this schedule the ride time constraint is also satisfied. The ride time of the first request is $24 - 10 = 14 < L_1^{max}$. Note that starting as early as possible from the depot and delaying the beginning of service at the origin of the first request would have the same effect. It is clear that a more sophisticated strategy is needed for setting the beginning of service at each node.

In this paper, we use the eight-step scheme of Cordeau and Laporte, 2003. The procedure minimizes maximum ride time violations, implying that if it

produces an infeasible schedule, no feasible schedule exists. Before we explain the procedure in detail, a closer look at the concept of forward time slack is necessary. It is slightly different than the one in Equation (32) in the sense that it considers the maximum ride time constraint. The formula is:

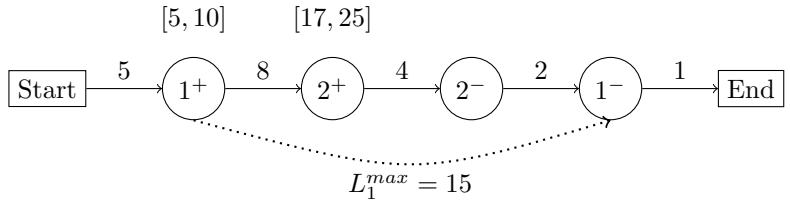
$$F_i^{\text{DARP}} = \min_{i \leq j \leq k} \{w_{i,j} + (\min\{l_j - b_j, L_j^{max} - L_j\})^+\} \tag{34}$$

where L_j is equal to the actual ride time of the request, if j is an origin location. Note that $L_j^{max} = \infty$ and $L_j = 0$ when j is not an origin node. Equation (34) represents the maximum amount of time by which service can be delayed at node i, such that the time window constraint of i and all subsequent nodes is respected and the maximum ride time violation of any request whose origin precedes i and destination lies after i is not increased. The procedure mentioned above is the following. First, a schedule in which the vehicle departs the earliest possible time from the origin depot and each node is served as early as possible is created Then, the duration of the route is minimized by delaying the departure from the origin depot. Finally, the maximum ride time constraint violations are minimized by delaying the beginning of service at all origin nodes as much as possible. S. N. Parragh et al., 2010 modified the procedure so that when no maximum ride time violations exist in any step of the procedure, it terminates. We do not follow that approach due to the fact that we aim to minimize excess ride time together with total distance traveled, which means we need to reduce ride times to the fullest extent.

6.4 Updating Data

The following steps are performed in the order presented, any time a modification on a route is finalized:

1. Set the charging time at each charging station of the route as described in Section 6.1.
2. Compute the load and battery level at each node.
3. Compute the zero-split segments of the route, which will speed up the Worst Zero-Split Removal operator described in Section 5.4.
4. Perform the eight-step scheme of Cordeau and Laporte, 2003. Note that the information regarding the as-early-as-possible schedule are not dis-



Service=0	5	17	21	23	24
Service=5	10	18	22	24	25

Figure 3: Incorrect ride time infeasibility declaration example

carded, but stored and used for the time window feasibility check described Algorithm 11.

5. Compute the total excess ride time of the route.

Note that the charging time at the stations of a route does not affect the ride time of any request, due to constraint (19), which states that no passengers are allowed to be in the vehicle while it is charging.

7 Computational Experiments

The algorithms were implemented in C++ and compiled in Visual Studio 2019. The tests were performed using an AMD A9-9420 Radeon R5 CPU with base speed equal to 3 GHz and with 6GB of RAM. The full implementation and code are in the following link: https://github.com/stbakeas/EADARP.git.

7.1 Test Instances

In order to test our method, we run the algorithm on existing benchmark instances for the version of the problem introduced by Bongiovanni et al., 2019, which vary in size and problem characteristics.

7.1.1 Benchmark Instances

The data set consists of three types of instances: type-a and type-u instances introduced by Bongiovanni et al., 2019 and type-r instances introduced by Su et al., 2021.

Type-a instances were introduced by Cordeau, 2006 for the standard version of the problem but were extended for the purpose of the EADARP. The number of vehicles range between 2 and 5 and the number of requests between 16 and 50. Regarding type-r instances, they were generated by Ropke et al., 2007 and were extended by Su et al., 2021 for the EADARP. The number of vehicles range from 6 to 8 and the number of requests from 60 to 96. The following apply for both types of instances:

- Each node is located inside a $[-10, 10]^2$ square. The travel distance between two points is equal to the Euclidean distance between them.
- Half of the requests are inbound and half are outbound.
- The origin(destination) of inbound(outbound) requests must be served within 15 minutes after the declared earliest time.
- Vehicle capacity is set to 3. For each origin location $i \in \mathcal{P}$, $q_i = 1$.
- The maximum ride time is set to 30.

All instances are supplemented with charging stations, recharging and discharging rates as well as vehicle battery capacity. Both the maximum and initial battery level of each vehicle is set to 14.85 KWh. All recharging and discharging rates are set to 0.055 KWh per minute. Three charging stations were placed in positions (-4,4),(0,0) and (4,4) respectively.

Type-u instances were generated using real-world ridesharing data of Uber Technologies in San Francisco. First, origin/destination locations were extracted by GPS logs. Then, the travel distance between them was calculated, assuming a vehicle is traveling at constant speed of 35 km/h. The locations of the charging stations were retrieved using the Alternative Fueling Station Locator. More details on the way in which the instances were generated can be found in Bongiovanni et al., 2019[2]. The number of requests range from 16 to 50 and the number of vehicles from 2 to 5, just like in type-a instances. Maximum ride time was set to eight minutes. There are a total of five charging stations, whose

[2]The generated instances were found in the following link: https://luts.epfl.ch/wpcontent/uploads/2019/03/e_ADARP_archive.zip

locations differ and coincide with the locations of the depots. Maximum battery capacity is set to 3.5 KWh. The discharge rate is set to 0.0715 KWh.

7.2 Notation

Each instance is named using the form $xK - n$, where $x \in \{a, u, r\}$ denotes the instance type, K denotes the number of vehicles and n denotes the number of requests.

The proposed algorithm largely depends on randomization. Therefore, in order to come up with a good estimate about its performance, the algorithm is run 10 times using different seeds. For each instance, the following values are reported:

- FeasRatio, denoting the number of runs where a feasible solution was found.
- *BKS* denotes the cost of the best known solution for the instance.
- *BC* denotes the cost of the best solution found by ALNS after all 10 runs.
- *BC%* denotes the relative change between *BKS* and *BC*.
- *AC* denotes the average value of the solution costs found by ALNS after all 10 runs.
- *AC%* denotes the relative change between *BKS* and *AC*.
- *CPU(s)* denotes the computational time for the execution of all 10 runs in seconds.
- *AvgIter* denotes the iteration in which the best solution was found on average.

The relative change formula is:

$$\frac{Value - BKS}{BKS} \cdot 100\%$$

If one of the above mentioned values cannot be reported, we report it as *NA*(Not Available). Similar to Bongiovanni et al., 2019, each instance will be examined separately for the different values of γ, which are $0.1, 0.4$ and 0.7. Each charging station can be visited only once. Moreover, $(\lambda_1, \lambda_2) = (0.75, 0.25)$.

7.3 Parameter Tuning

The performance of the ALNS algorithm is controlled by many factors. Among those are certain parameters whose value must be defined beforehand. Assuming n parameters and a configuration $Y = (y_1, y_2, \ldots, y_n), y_i \in Y_i, \forall i = 1, 2, \ldots, n.$ of those, where Y_i is a discrete set of values that the i-th parameter can take, the problem lies in finding the configuration that produces the best results with respect to certain criteria. The optimal strategy to solve this problem would be to execute the algorithm many times on a sufficiently large training set of instances for each configuration and proceed with the best one. Due to the large number of parameters, this would be computationally expensive. For example, we assume a training set of 6 instances and that for each configuration, 10 runs are performed for each instance. There are 11 parameters to be set and the discrete set of each consists of 3 values. Therefore, there are 3^{11} configurations, resulting in a total of $3^{11} \cdot 10 \cdot 6 = 10.628.820$ runs.

We will resort to a sub-optimal but more computationally efficient strategy in order to find a good setting. The first step of our strategy involves providing an initial setting, which is based either on intuition or preliminary experimentation, and deciding which parameters will remain fixed for the rest of the process. Afterwards, the parameters to be set are divided into two groups, assuming that parameters of the same group are interdependent and of different groups are not. Fixing the values of one group, the best configuration for the other group is found. Given the best values of the first group, the best configuration for the other is found. The resulting setting will be used for the rest of the paper. For more advanced tuning methods, we refer the interested reader to Montero et al., 2014 and Montgomery, 2017.

The initial setting and value range for each parameter is presented in Table 1. If a dash(-) is present in the Range column, it means that the parameter will remain fixed. One group consists of the frequency of updating the operators(N_w) and the reaction factor(α), while the other consists of the removal randomness and percentage. No initial values are set to the parameters of the latter, due to the fact that they will be tuned first. The training set consists of all type-a instances with up to 3 vehicles and 36 requests and the algorithm is run 10 times for each instance. The quality of a configuration is measured by the average of the gaps between the average solution cost over the 10 runs and the cost of the best known solution of each instance. The total computational time is also taken into account, which is significantly affected by the number of iterations and removal percentage. Note that we repeat the process for each value of γ, resulting in different best configurations Note that the rejected request penalty described in Section 5.3 was taken into account, which might result in very large gaps if a feasible solution is not found for an instance after a run.

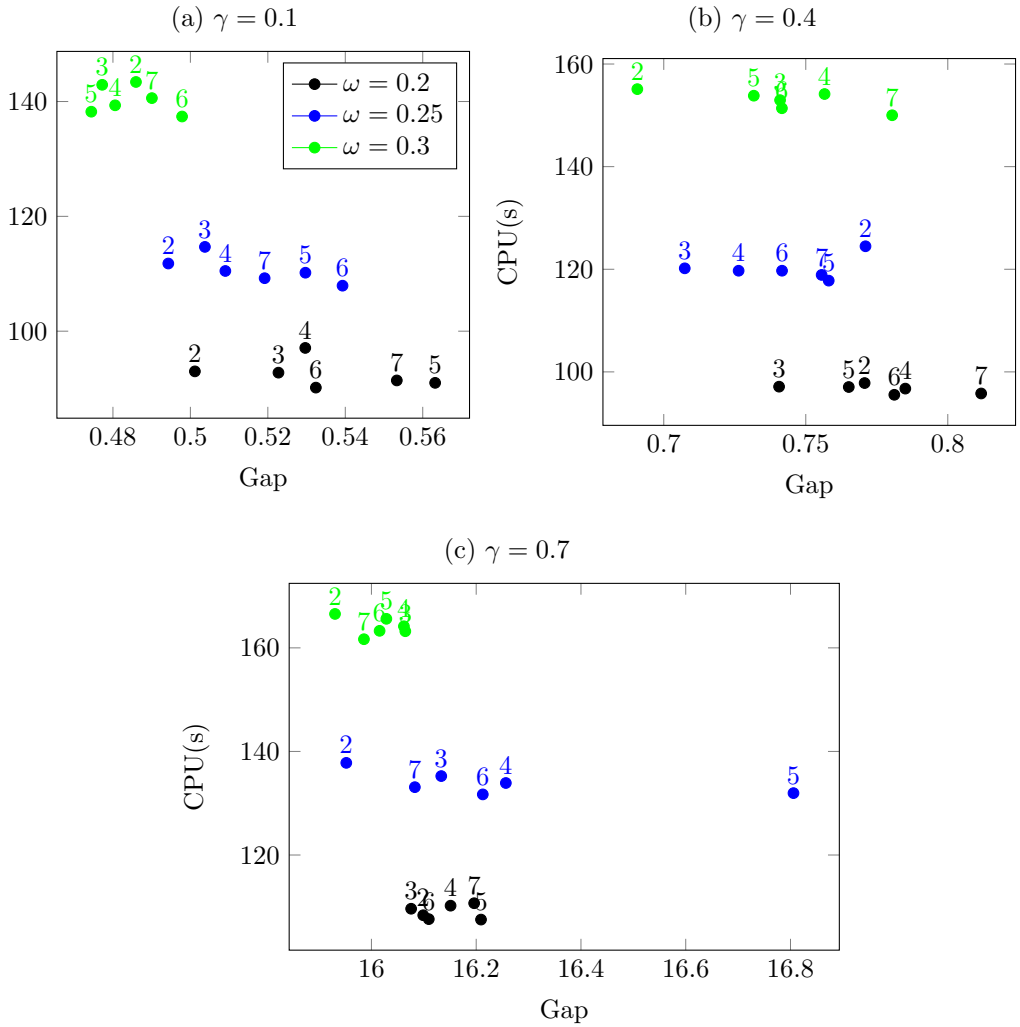
7.3.1 Removal Percentage and Randomness

The results of the experiment are shown in Figure 4. The removal randomness value is presented above every data point. Regarding the removal percentage ω, it is clear that the average solution quality improved when its value increased, but the computational effort was increased. Ropke and Pisinger, 2006 discovered that by increasing ω even more, at some point a plateau was reached and the solution cost started to increase again, due to the fact that the good attributes of a solution were lost. We did not test for larger values due to the increased computational time. For $\gamma = 0.7$, the configurations with $\omega = 0.2$ performed well. As for the removal randomness, it can be observed that the configurations that resulted in the smallest gaps were those with removal randomness values in the lower range. With this information in mind, we decided to proceed with $\omega = 0.25$ for all values of γ, with the removal randomness parameter p_{remove} equal to 3 for $\gamma = 0.4$ and equal to 2 for $\gamma = 0.1, 0.7$. We believe that the selected configurations strike a good balance between solution quality and computational effort. Note that performance of the algorithm is not shown for $\omega = 0.1, 0.15$ due to significant increases in the average gap.

7.3.2 Reaction Factor and Update Frequency

The results are demonstrated in Figure 5. The computational time is not significantly affected by the update frequency, hence it is not reported. For $\alpha = 0$, we

Figure 4: Tuning removal percentage and randomness

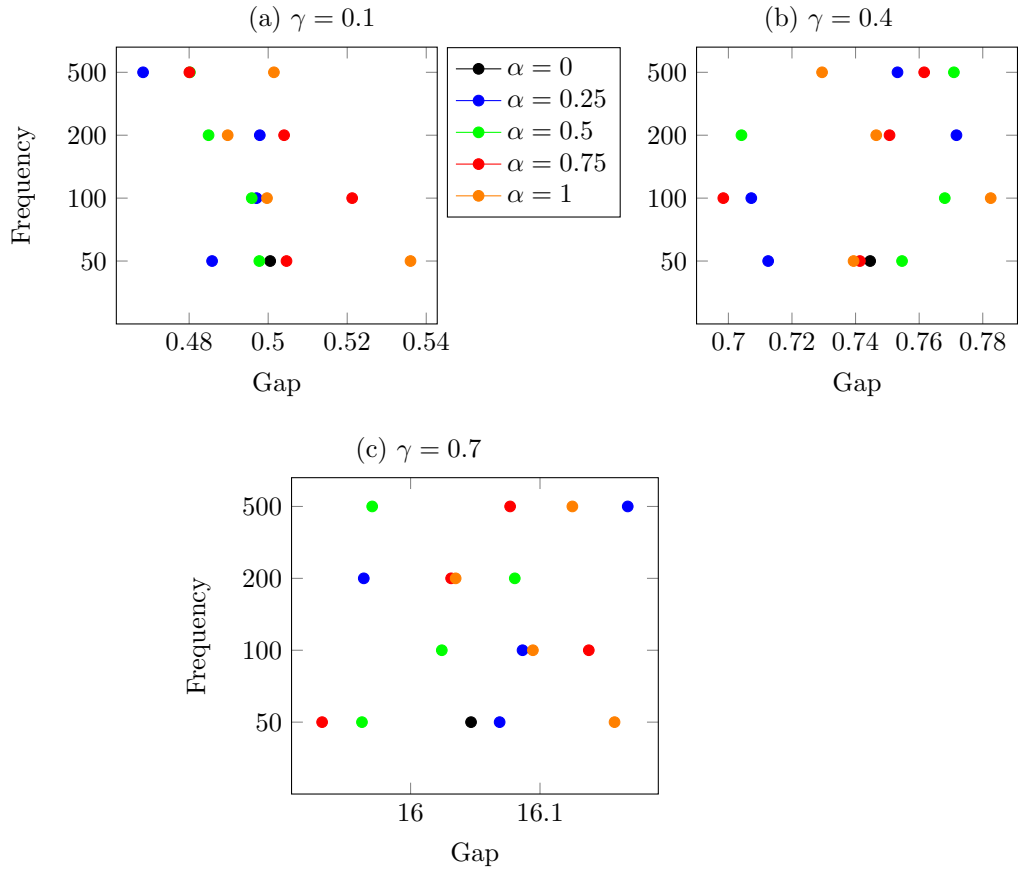


did not test for different update frequency values, due to the fact that the likelihood of an operator being selected remains the same throughout the execution of the algorithm. Note that the more constrained the instances become(i.e. γ value increases), the lower the update frequency value of the best configuration. With that in mind, the update frequency N_w is set to 500 for $\gamma = 0.1$, 100 for $\gamma = 0.4$ and 50 for $\gamma = 0.7$. Regarding the reaction factor α, it is set to 0.75 regardless of the value of γ.

7.4 Performance Evaluation

We conducted experiments on the benchmark instances in order to evaluate the performance of ALNS under $\gamma \in \{0.1, 0.4, 0.7\}$. If an ALNS solution is equal in terms of quality with the best known solution for an instance, it is marked in

Figure 5: Tuning reaction factor(α) and operator weight update frequency(N_w)



bold. If the ALNS solution is better, an asterisk is also added.

7.4.1 Type-a Instances

In Table 2, the results of our algorithm are presented compared to the best known solution of each instance. Our algorithm finds the best known solution for instance a2-24, when $\gamma = 0.1$ and for instance a4-16,for all values of γ. Surprisingly, our algorithm managed to find new best solutions for the last three instances when $\gamma = 0.7$ with improvements equal to 0.92%, 0.96% and 2.11% respectively. In fact, for instance a5-50-0.7, the average solution cost over 10 runs was smaller than the cost of the best known by 1.15%.

Regarding the rest of the instances, the average gap when $\gamma = 0.1, 0.4$ is less than 1%. When $\gamma = 0.7$ the instances for which the average gap exceeds 1% are: a2-16 (1.46%), a3-18(2.36%) and a4-40(1.66%). For instances a2-20 and a2-24, no feasible solution was found, despite running ALNS 10 times.

Regarding the computational time, the maximum time spent to run the algorithm for an instance is approximately 11 minutes. However, almost half of the iterations were redundant, suggesting that the iteration number could be

decreased without sacrificing solution quality.

7.4.2 Type-u Instances

Previous works that solved this type of instances assumed that the destination depot each vehicle returns to is a decision variable. Therefore, in order to be able to compare our results, we have designed a backtracking algorithm for the purpose of selecting the destination depot for each route, which is executed after the rejected reinsertion step in the ALNS. After randomly selecting an available vehicle, the closest destination depot to the last node of its route is inserted. Then, another vehicle is chosen in the same fashion. The algorithm terminates when all routes have been assigned a destination depot in a feasible manner and the total distance traveled has been decreased. If the insertion of the closest available depot of the last node of a route is infeasible or a feasible solution does not decrease the total cost, the algorithm backtracks, reverting a decision it previously made and making a new one (i.e. assigning the second closest available depot to the route of a vehicle). For the purpose of this algorithm, for each destination location and charging station, the destination depots are sorted in non-increasing order based on their distance to the previously mentioned nodes in a preprocessing step.

In Table 3, the results of our proposed approach on the Uber Instances for different values of γ are shown. Our algorithm managed to find the best known solution for eleven instances in total, some of which have been proven optimal. A small improvement in the best known solution for instance u4-48-0.1 was observed. For $\gamma = 0.1, 0.4$, the gap between the best solution found by our algorithm and BKS remained below 1% for almost all instances, except for u2-20-0.4, which was equal to 3.25%. For $\gamma = 0.7$, gaps up to 8% were reported. Moreover, the algorithm found a feasible solution 50% of the time for instances u2-16 and u4-48. No feasible solution was found after 10 runs for instance u2-24, for $\gamma = 0.4, 0.7$.

7.4.3 Type-r Instances

Table 4 contains the results of the execution of our algorithm on type-r instances. Our algorithm managed to find feasible solutions for seven previously unsolved instances when $\gamma = 0.7$. A feasible solution was found in all ten runs for four out of those seven instances, whereas for the remaining three instances, namely a6-60, a7-70 and a8-80, a feasible solution was found seven,two and nine times respectively. In addition, our algorithm found new best solution for seven instances, namely, a5-60,a6-60 and a8-96 when $\gamma = 0.1, 0.4$ and also a6-72, when $\gamma = 0.4$. Specifically, for instances a6-72-0.4 and a7-84-0.4 the improvements were substantial, 3.26% and 5.12% respectively. The average solution cost found for these instances was smaller than BKS as well. Regarding the rest of the instances, the average gap was above 1% for four instances when $\gamma = 0.1$ and also for four instances when $\gamma = 0.4$. The highest average gap reported was 2.1% for instance a8-64-0.7.

7.4.4 Discussion on overall algorithmic performance

After experimenting on all benchmark instances, certain patterns have emerged regarding the behaviour of our proposed approach. It is safe to say that the algorithm is effective for finding feasible solutions for larger and highly constrained instances. New best solutions were found for many type-r instances and also previously unsolved instances were solved. However, the algorithm was proven to be weaker in finding feasible solutions for smaller but still highly constrained instances. It can also be observed that the average iterations needed to find the best solution increase as the size of the instance increases. The increase is not linear, since the individual characteristics of each instance also matter. Nevertheless, it would be wise to set the number of iterations of each run according to the size of the instance. With the above information in mind, we can conclude that, although our algorithm did not manage to find or improve the best known solution for all instances, the results highlighted the room for improvement that exists in this area of research.

8 Conclusion

In this paper, an ALNS algorithm has been designed to solve the dial-a-ride problem with electric autonomous vehicles. The additional constraints unique to the EADARP are the limited driving range and the lower bound on the battery of the vehicles upon returning to the depot. Recharging stations can be visited for a specific number of times in order to replenish battery, but passengers inside the vehicles are not allowed. The objective function is a weighted sum of the total distanced traveled and excess user ride time (Bongiovanni et al., 2019). After creating an initial solution, we iteratively remove some requests and then reinsert them to the solution using different operators. After the removal step, the charging stations that do not contribute to maintaining the battery feasibility of the route are removed. In order to reduce computational effort, an insertion feasibility test is applied to avoid the evaluation of infeasible neighbors.

The efficiency of the proposed algorithm is investigated on benchmark instances. The best solution found by our algorithm is compared to the best known solutions reported in Su et al., 2021 and Su et al., 2022. The algorithm managed to find the optimal solution of fifteen instances, improve the best known solution on eleven and find feasible solution on seven instances that previous works failed to solve. Small gaps to the best known solutions are reported for the rest of the instances.

Due to limited time, this study has potential limitations. The route evaluation scheme of Cordeau and Laporte, 2003 used provides a schedule that is feasible but not necessarily optimal with respect to the user excess ride time. A more sophisticated scheduling algorithm similar to the one proposed by Bongiovanni, Geroliminis, et al., 2022 could be used instead. Moreover, in this paper we assume that the fuel replenished in charging stations and consumed by the vehicles is calculated using a linear function of the charging time and travel distance respectively, which is not always realistic. Therefore, it would be interesting to integrate a non-linear charging function(Montoya et al., 2017) and a realistic energy consumption model(Zhang et al., 2018, Ma et al., 2021) to the problem and observe the effects on solution quality. In addition, in spite of the fact that the computational time is not prohibitive, it could be further improved by trying to find the best insertion position of a request for all vehicles in parallel. Finally, one could study the multi-objective version of the problem, ignoring the weighted sum function and instead try to find an approximation of the Pareto front.

A Appendix

A.1 Algorithms

Algorithm 1 Detect Feasible Moves

Input: Unassigned rider request r, Solution s, Available vehicles $E \subseteq \mathcal{V}$
Output: Set of moves M

$M \leftarrow \emptyset$
for vehicle $h \in E$ **do**
    $M_h \leftarrow \emptyset$
    if $h \notin \mathcal{V}_r^{\mathcal{V}} \cup \mathcal{V}_r^{\mathcal{R}}$ **then**
        Let θ_h^s be the route of h in s
        Find all feasible insertions of r in θ_h^s and add them to M_h
        if no battery-feasible insertion has been found **then**
            for each zero-load node $v \in \theta_h^s$ **do**
                Find all the available charging stations that can be reached from v without running out of battery
                if such a station exists **then**
                    Let cs be the station that gives the minimum detour
                    if the insertion of cs after v does not violate time window constraints **then**
                        Find all feasible insertions of r in θ_h^s (that include the insertion of cs after v) and add them to M_h
                        if at least one feasible insertion was found **then**
                            Break
                        end if
                    end if
                end if
            end for
        end if
    end if
    $M \leftarrow M \cup M_h$
end for

Algorithm 2 Worst Removal

Input: Solution s, Removal Percentage ω, Randomness parameter p
Output: Solution s', Set D of requests

$s' \leftarrow s$
$D \leftarrow \emptyset$
Let R_s the set of assigned requests of s, sorted in non increasing order based on the contribution of each request in the cost of s
Let $count \leftarrow \lceil \frac{\omega \cdot |R_s|}{100} \rceil$ the number of requests that will be removed from s
$i \leftarrow 0$
while $i < count$ **do**
    Generate random number k, where $0 \leq k < 1$
    $index \leftarrow \lfloor k^p \cdot |R_s| \rfloor$
    $r \leftarrow R_s[index]$
    $s' \leftarrow Remove(r, s')$
    $D \leftarrow D \cup \{r\}$
    $R_s \leftarrow R_s \backslash \{r\}$
    $i \leftarrow i + 1$
end while

Algorithm 3 Worst Zero Split Removal

Input: Solution s, Removal Percentage ω, Randomness parameter p
Output: Solution s', Set D of requests

$s' \leftarrow s$
$D \leftarrow \emptyset$
Let Z_s the set of zero-split segments of s
Sort Z_s in non increasing order, based on the segment duration
Let $count \leftarrow \lceil \frac{\omega \cdot |R_s|}{100} \rceil$ the number of requests that will be removed from s
$i \leftarrow 0$
while $i < count$ **do**
    Generate random number k, where $0 \leq k < 1$
    $index \leftarrow \lfloor k^p \cdot |Z_s| \rfloor$
    $z \leftarrow Z_s[index]$
    for each request $r \in z$ **do**
        $s' \leftarrow Remove(r, s')$
        $D \leftarrow D \cup \{r\}$
        $i \leftarrow i + 1$
    end for
    $Z_s \leftarrow Z_s \backslash \{z\}$
end while

Algorithm 4 Similarity Removal

Input: Solution s, Removal Percentage ω, Randomness parameter p
Output: Solution s', Set D of requests

$s' \leftarrow s$
$D \leftarrow \emptyset$
$i \leftarrow 0$
Let R_s the requests that are served in solution s
Let $count \leftarrow \lceil \frac{\omega \cdot |R_s|}{100} \rceil$ the number of requests that will be removed from s
if s has no rejected requests AND $count > 0$ **then**
    Let r be a random request from the assigned ones
    $s' \leftarrow Remove(r, s')$
    $D \leftarrow D \cup \{r\}$
    $R_s \leftarrow R_s \backslash \{r\}$
    $i \leftarrow i + 1$
else
    Let r a random rejected request
end if
Sort R_s in non decreasing order based on their similarity to r
while $i < count$ **do**
    Generate random number k, where $0 \leq k < 1$
    $index \leftarrow \lfloor k^p \cdot |R_s| \rfloor$
    $r' \leftarrow R_s[index]$
    $s' \leftarrow Remove(r', s')$
    $D \leftarrow D \cup \{r'\}$
    $i \leftarrow i + 1$
    $R_s \leftarrow R_s \backslash \{r'\}$
end while

Algorithm 5 Greedy Insertion

Input: Partial Solution s, Set of removed requests D
Output: Solution s'

$s' \leftarrow s$
Let H associative array, which stores for each removed request, its best insertion move for each route
for request $r \in D$ **do**
    for vehicle $h \in \mathcal{V}$ **do**
        $M_h \leftarrow$ DetectFeasibleMoves(r, s', h)
        if $M_h \neq \emptyset$ **then**
            $m^* \leftarrow \min_{m \in M_h} m(s').cost$
            $H[r][h] \leftarrow m^*$
        else
            $H[r][h] \leftarrow \infty$ ▷ Infeasible move
        end if
    end for
end for
while $D \neq \emptyset$ **do**
    Let $m_{best} \leftarrow \min_{m \in H} m(s').cost$ the globally best move
    if $m_{best}(s').cost = \infty$ **then**
        for $r \in D$ **do**
            Add r to the rejected requests of s'
            $D \leftarrow D \backslash \{r\}$
        end for
    else
        $s' \leftarrow m_{best}(s')$
        Let r_{best} the request associated with m_{best}
        $D \leftarrow D \backslash \{r_{best}\}$
        ▷ Associative Array Update Case i)
        for $r \in D$ **do**
            $M_h \leftarrow$ DetectFeasibleMoves$(r, s', m_{best}$.vehicle$)$
            if $M_h \neq \emptyset$ **then**
                $m^* \leftarrow \min_{m \in M_h} m(s').cost$
                $H[r][m_{best}$.vehicle$] \leftarrow m^*$
            else
                $H[r][m_{best}$.vehicle$] \leftarrow \infty$ ▷ Infeasible Move
            end if
        end for
        ▷ Associative Array Update Case ii)
        for $r \in D$ **do**
            for $m \in H[r]$ **do**
                if m.station is visited more times than allowed in s' **then**
                    $M \leftarrow$ DetectFeasibleMoves$(r, s', m$.vehicle$)$
                    if $M \neq \emptyset$ **then**
                        $m^* \leftarrow \min_{m \in M} m(s').cost$
                        $H[r][m$.vehicle$] \leftarrow m^*$
                    else
                        $H[r][m$.vehicle$] \leftarrow \infty$ ▷ Infeasible Move
                    end if
                end if
            end for
        end for
    end if
end while

Algorithm 6 Regret Insertion

Input: Partial Solution s, Set of removed requests D, Random number $k \in [2, |\mathcal{V}|]$
Output: Solution s'

$s' \leftarrow s$
Let H associative array, which stores for each removed request, its best insertion move for each route
Initialize H as in Algorithm 5
while $D \neq \emptyset$ **do**
    Let m_{best} the globally best move and r_{best} the request associated with the move
    $m_{best}.cost \leftarrow \infty$
    $\text{regret}_{\max} \leftarrow -\infty$
    for request $r \in D$ **do**
        Sort set of moves $H[r]$ in non-increasing order of cost
        $\text{regret} = \sum_{i=0}^{k}(H[r][i].cost - H[r][0].cost)$
        if $\text{regret}_{\max} < \text{regret}$ **then**
            $\text{regret}_{\max} \leftarrow \text{regret}$
            $m_{best} = H[r][0]$
            $r_{best} = r$
        else if $\text{regret}_{\max} = \text{regret}$ and $H[r][0].cost < m_{best}.cost$ **then**
            $m_{best} = H[r][0]$
            $r_{best} = r$
        end if
    end for
    if $m_{best}(s').cost = \infty$ **then**
        for $r \in D$ **do**
            Add r to the rejected requests of s'
            $D \leftarrow D \backslash \{r\}$
        end for
    else
        $s' \leftarrow m_{best}(s')$
        Let r_{best} the request associated with m_{best}
        $D \leftarrow D \backslash \{r_{best}\}$
        Update H as in Algorithm 5
    end if
end while

Algorithm 7 Rejected Reinsertion

for each rejected request r from s **do**
  Let R_s the set of all assigned requests of s, sorted in non decreasing order based on their similarity to r
  failed $\leftarrow 1$
  for $r' \in R_s$ **do** ▷ From most similar to least
    Let h be the vehicle r' is assigned to and θ_h^s its route in s
    $\theta_h^s \leftarrow \theta_h^s$.Swap$(r', r)$
    if θ_h^s is feasible **then**
      $s' \leftarrow s \cup \{\theta_h^s\}$
      $M \leftarrow$ DetectFeasibleMoves$(r', s', \mathcal{V})$
      if $M \neq \emptyset$ **then**
        $m^* \leftarrow \min_{m \in M} m(s').cost$
        if $m^*(s').cost \neq \infty$ **then**
          $s \leftarrow m^*(s')$
          failed $\leftarrow 0$
          Break
        end if
      end if
    end if
  end for
  if failed **then**
    Finalize the rejection of r in s
  end if
end for

Algorithm 8 Adaptive Large Neighborhood Search

Input: Initial Solution s_{init}, Number Of Iterations N_{iter}, Temperature Control z
Output: Solution s_{best}

$T \leftarrow \frac{-z \cdot s_{init}.cost}{\ln 0.5}$ ▷ Initial Temperature
$\rho \leftarrow (\frac{0.01}{T})^{1/N_{iter}}$ ▷ Cooling Rate
$s_{best}, s_{curr} \leftarrow s_{init}$
while termination criterion has not been met **do**
  Select removal and insertion operator (see Section 5.6)
  $s' \leftarrow$ Insertion(Removal(s_{curr}))
  RejectedReinsertion(s') ▷ Section 5.7
  $g \leftarrow$ random number between 0 and 1
  if $s'.cost < s_{curr}.cost$ **or** $exp(\frac{s_{curr}.cost - s'.cost}{T}) > g$ **then**
    $s_{curr} \leftarrow s'$
  end if
  if $s'.cost < s_{best}.cost$ **then**
    $s_{best} \leftarrow s'$
  end if
  Update scores and weights for removal and insertion operators
  $T \leftarrow T \cdot \rho$
end while

Algorithm 9 is Insertion Battery Feasible

Input: Request r, Nodes v_i, v_j
Output: Feasibility of insertion with respect to battery constraints

Let $v_{left}(v_{right})$ the closest charging station preceding(succeeding) i and j.
if $v_{left} = \emptyset$ **then** ▷ Only the starting depot precedes i
    $limit = 0$
    if $v_{right} = \emptyset$ **then** ▷ Only the ending depot succeeds j
        $limit = \gamma \cdot bat_{\max}$
        $right \leftarrow$ ending depot index
    end if
    return $bat_{right} - \beta_{r,i,j} \geq limit$
end if
return $bat_{left+1} + \beta_{left,left+1} + \beta_{r,i,j} \leq bat_{max}$

Algorithm 10 Feasibility Test

Input: Request r, Nodes v_i, v_j
Output: Feasibility of inserting r^+ after v_i and r^- after v_j

▷ Infeasible Arcs
if $(v_i, r^+) \in I$ or $(r^-, v_{j+1}) \in I$ **then return** false
end if
if $i \neq j$ **then**
    if $(r^+, v_{i+1}) \in I$ or $(v_j, r^-) \in I$ **then return** false
    end if
end if
▷ Constraint check
Perform capacity feasibility test
if insertion violates capacity constraints **then return** false
end if
Perform time window feasibility test
if insertion violates time window constraints **then return** false
end if
Perform battery feasibility test
if insertion violates battery constraints **then return** false
end if
return true

Algorithm 11 Time Window Feasibility Test

Input: Request r, Nodes v_i, v_j
Output: Feasibility of inserting r^+ after v_i and r^- after v_j

$b'_i = b_i$
Let $left$ be the index of the closest charging station preceding v_{i+1}. If no charging station precedes v_{i+1}, it is equal to 0
if $left \neq 0$ **then**
    Let E_{left} the charging time needed to reach the next station or ending depot after the insertion of r
    if $left < i$ **then**
        $\delta_{left+1} = (b_{left} + E_{left} + t_{left,left+1} - b_{left+1})^+$ ▷ $x^+ = max(0, x)$
        if $\delta_{left+1} > F^{\text{PDPTW}}_{left+1}$ **then return** false
            $b'_i = b_i + (\delta_{left+1} - w_{left+1,i})^+$
        end if
    end if
end if
Let d equal to E_{left} if $left = i$, d_i otherwise
$b'_{r^+} = \max\{e_{r^+}, b'_i + d + t_{i,r^+}\}$
if $b'_{r^+} > l_{r^+}$ **then return** false
end if
if $i = j$ **then**
    $b'_{r^-} = \max\{e_{r^-}, b'_{r^+} + d_{r^+} + t_{r^+,r^-}\}$
else
    $\delta_{i+1} = (b'_{r^+} + d_{r^+} + t_{r^+,i+1} - b_{i+1})^+$
    if $\delta_{i+1} > F^{\text{PDPTW}}_{i+1}$ **then return** false
    end if
    $b'_j = b_j + (\delta_{i+1} - w_{i+1,j})^+$
    $b'_{r^-} = \max\{e_{r^-}, b'_j + d_j + t_{j,r^-}\}$
end if
if $b'_{r^-} > l_{r^-}$ **then return** false
end if
$\delta_{j+1} = (b'_{r^-} + d_{r^-} + t_{r^-,j+1} - b_{j+1})^+$
if $\delta_{j+1} > F^{\text{PDPTW}}_{j+1}$ **then return** false
end if
if $E_{left} = 0.0$ **then**
    for each charging station node $v_s, s > j$ **do**
        $b'_s = b_s + (\delta_{j+1} - w_{j+1,s})^+$
        Let $E'_s(E_s)$ the new(old) charging time at station s
        if $E'_s - E_s > 0.0$ **then**
            $\delta_{s+1} = (b'_s + E'_s + t_{s,s+1} - b_{s+1})^+$ **return** $\delta_{s+1} \leq F^{\text{PDPTW}}_{s+1}$
        end if
    end for
end if
return True

A.2 Tables

Table 1: Initial Parameter Setting

Parameter	**Value**	**Range**
Number of Iterations N_{iter}	10000	-
Operator Weight Update Frequency(iterations) N_w	100	$\{50, 100, 200, 500\}$
Reaction Factor α	0.5	$\{0, 0.25, 0.5, 0.75, 1\}$
New global solution score ψ_1	10	-
Better solution score ψ_2	7	-
Simulated Annealing Acceptance Score ψ_3	5	-
Removal Randomness p_{remove}	-	$[2, 7]$
Removal Percentage ω	-	$\{0.1, 0.15, 0.2, 0.25, 0.3\}$
Start Temperature Control z	0.05	-
Distance Similarity Coefficient Φ_1	1	-
Time Window Similarity Coefficient Φ_2	1	-
Load Similarity Coefficient Φ_1	1	-

Table 2: Results of ALNS on type-a instances under different values of γ

$\gamma = 0.1$		ALNS-RR(10 runs)					
Instance	BKS	BC	BC%	AC	AC%	CPU(s)	AvgIter
a2-16	237.38	238.20	0.34	238.20	0.34	44.842	345
a2-20	279.08	281.00	0.69	281.00	0.69	92.998	371
a2-24	346.21	**346.21**	0	346.48	0.08	122.643	680
a3-18	236.82	238.726	0.8	238.726	0.8	53.858	276
a3-24	274.8	275.185	0.14	275.301	0.18	102.519	1191
a3-30	413.27	414.878	0.39	415.272	0.48	182.022	2587
a3-36	481.17	483.862	0.56	485.293	0.86	222.016	4636
a4-16	222.49	**222.49**	0	**222.49**	0	37.723	281
a4-24	310.84	311.478	0.21	311.478	0.21	90.139	1453
a4-32	393.96	395.104	0.29	395.104	0.29	152.214	2503
a4-40	453.84	457.84	0.88	458.669	1.06	254.607	2850
a4-48	554.54	557.632	0.56	559.182	0.84	448.464	5914
a5-40	414.51	415.624	0.27	416.517	0.48	265.852	3976
a5-50	559.17	560.071	0.16	562.546	0.6	467.647	4285
$\gamma = 0.4$	BKS	BC	BC%	AC	AC%	CPU(s)	AvgIter
a2-16	237.38	238.198	0.34	238.411	0.43	49.636	400
a2-20	280.7	282.624	0.69	282.624	0.69	95.493	121
a2-24	347.04	349.358	0.67	350.427	0.98	135.639	1701
a3-18	236.82	238.726	0.8	238.726	0.8	64.098	145
a3-24	274.8	275.185	0.14	275.185	0.14	104.603	739
a3-30	413.34	414.979	0.4	415.466	0.51	175.6	2482
a3-36	482.75	487.829	1.05	490.099	1.52	238.657	4244
a4-16	222.49	**222.49**	0	**222.49**	0	37.968	353
a4-24	311.03	311.478	0.14	311.478	0.14	99.59	993
a4-32	394.26	395.407	0.29	395.546	0.33	162.97	2160
a4-40	453.84	457.84	0.88	458.089	0.94	266.195	2675
a4-48	554.6	561.466	1.24	562.771	1.47	511.653	4008
a5-40	414.51	415.624	0.27	416.626	0.51	280.849	3526
a5-50	559.51	563.139	0.65	567.056	1.35	531.021	5202
$\gamma = 0.7$	BKS	BC	BC%	AC	AC%	CPU(s)	AvgIter
a2-16	240.66	244.162	1.46	244.941	1.78	47.603	544
a2-20	293.27	NA	NA	NA	NA	129.886	1475
a2-24	353.18	NA	NA	NA	NA	186.345	888
a3-18	240.58	246.263	2.36	246.366	2.41	69.606	522
a3-24	275.97	278.099	0.77	278.373	0.87	113.915	1925
a3-30	424.93	426.385	0.34	432.177	1.71	170.243	1283
a3-36	494.04	497.227	0.65	500.401	1.29	228.973	2774
a4-16	223.13	**223.13**	0	**223.13**	0	45.963	238
a4-24	316.65	319.465	0.89	319.74	0.98	93.318	1791
a4-32	397.87	399.019	0.29	399.811	0.49	154.497	1721
a4-40	467.72	475.487	1.66	482.294	3.12	252.219	3392
a4-48	582.22	**576.892***	-0.92	583.077	0.15	445.512	4254
a5-40	424.26	**420.203***	-0.96	426.01	0.41	244.837	5377
a5-50	603.24	**590.514***	-2.11	**596.324***	-1.15	460.214	2786

Table 3: Results of ALNS on Uber(Type-u) instances under different values of γ

$\gamma = 0.1$		ALNS-RR(10 runs)					
Instance	BKS	BC	BC%	AC	AC%	CPU(s)	AvgIter
u2-16	57.61	**57.61**	0.0	**57.61**	0.0	125.207	260
u2-20	55.59	**55.59**	0.0	56.51	1.66	182.351	607
u2-24	90.73	91.54	0.89	92.33	1.76	189.571	1501
u3-18	50.74	**50.74**	0.0	50.82	0.17	143.955	1114
u3-24	67.56	67.67	0.16	68.01	0.66	199.04	2357
u3-30	76.75	**76.75**	0.0	76.8	0.06	541.147	1224
u3-36	104.04	104.06	0.02	105.18	1.1	523.285	1555
u4-16	53.58	**53.58**	0.0	53.96	0.71	212.12	670
u4-24	89.83	89.96	0.14	90.71	0.98	222.48	3299
u4-32	99.29	99.89	0.61	100.32	1.04	359.805	2797
u4-40	133.11	133.64	0.4	134.58	1.1	420.99	3644
u4-48	147.75	**147.68***	-0.05	148.49	0.5	837.766	4836
u5-40	121.86	122.41	0.45	123.54	1.38	619.421	2702
u5-50	142.82	144.04	0.85	144.51	1.19	1040.18	4805
$\gamma = 0.4$	BKS	BC	BC%	AC	AC%	CPU(s)	AvgIter
u2-16	57.65	**57.65**	0.0	57.98	0.58	77.714	547
u2-20	56.34	58.17	3.25	58.35	3.57	157.099	413
u2-24	91.24	NA	NA	NA	NA	232.135	265
u3-18	50.74	**50.74**	0.0	51.22	0.94	186.527	661
u3-24	67.56	**67.56**	0.0	68.25	1.02	255.131	2241
u3-30	76.75	76.86	0.14	77.07	0.42	418.855	1235
u3-36	104.06	104.49	0.41	105.95	1.81	364.718	2423
u4-16	53.58	**53.58**	0.0	53.82	0.44	182.627	597
u4-24	89.83	**89.83**	0.0	90.53	0.78	209.181	2792
u4-32	99.29	99.89	0.61	100.42	1.14	304.317	2704
u4-40	133.78	134.88	0.82	135.89	1.58	322.283	4261
u4-48	147.63	148.44	0.55	149.82	1.48	654.502	4341
u5-40	121.96	122.22	0.21	123.45	1.23	446.232	4816
u5-50	142.84	143.17	0.23	144.29	1.02	795.4	5119
$\gamma = 0.7$	BKS	BC	BC%	AC	AC%	CPU(s)	AvgIter
u2-16	59.19	63.93	8	NA	NA	99.908	382
u2-20	56.86	59.92	5.39	60.96	7.2	141.928	184
u2-24	92.17	NA	NA	NA	NA	268.69	704
u3-18	50.99	51.24	0.5	52.56	3.09	160.864	491
u3-24	68.39	68.74	0.52	69.19	1.17	162.021	2275
u3-30	77.41	78.74	1.72	79.07	2.15	303.726	1551
u3-36	105.79	109.01	3.05	110.36	4.32	366.3	2746
u4-16	53.87	**53.87**	0.0	54.62	1.4	76.784	1486
u4-24	89.96	91.02	1.18	91.51	1.72	126.142	2942
u4-32	99.5	100.69	1.2	101.51	2.02	235.146	2290
u4-40	136.08	137.26	0.87	137.94	1.37	280.366	4268
u4-48	152.58	156.66	2.68	NA	NA	740.655	5312
u5-40	124.01	124.3	0.24	125.13	0.9	361.012	3571
u5-50	143.51	145.74	1.55	146.85	2.33	688.023	5641

Table 4: Results of ALNS on type-r instances under different values of γ

$\gamma = 0.1$		ALNS-RR(10 runs)					
Instance	BKS	BC	BC%	AC	AC%	CPU(s)	AvgIter
a5-60	687.8	**685.04***	-0.4	688.72	0.13	646.869	4832
a6-48	506.45	508.13	0.33	508.73	0.45	361.201	4813
a6-60	692	**691.38***	-0.09	696.39	0.63	504.954	6941
a6-72	761.34	771.38	1.32	774.33	1.71	998.777	6852
a7-56	613.1	615.8	0.44	622.14	1.48	493.222	6612
a7-70	760.23	762.05	0.24	766.31	0.8	891.955	6519
a7-84	889.38	**881.31***	-0.91	**886.4***	-0.33	1180.07	6162
a8-64	632.22	637.95	0.91	642.53	1.63	653.917	6204
a8-80	788.99	794.93	0.75	800.58	1.47	1115.13	5376
a8-96	1053.11	**1046.32***	-0.64	**1049.3***	-0.36	1622.23	6909
$\gamma = 0.4$	BKS	BC	BC%	AC	AC%	CPU(s)	AvgIter
a5-60	697.97	**687.55***	-1.49	**693.04***	-0.71	702.61	6185
a6-48	506.45	507.58	0.22	508.49	0.4	378.608	4768
a6-60	689.46	692.03	0.37	696.48	1.02	588.168	8362
a6-72	799.6	**773.56***	-3.26	**778.27***	-2.67	1122.32	6814
a7-56	612.17	616.43	0.7	619.03	1.12	510.563	6252
a7-70	759.27	762.64	0.44	766.49	0.95	972.07	6247
a7-84	932.12	**884.4***	-5.12	**891.5***	-4.36	1591.36	7647
a8-64	632.22	641.33	1.44	645.48	2.1	700.278	6023
a8-80	788.99	795.85	0.87	800.04	1.4	1216.22	6454
a8-96	NA	**1048.96***	-	**1055.72***	-	1785.88	6723
$\gamma = 0.7$	BKS	BC	BC%	AC	AC%	CPU(s)	AvgIter
a5-60	NA	NA	-	NA	-	1029.07	4732
a6-48	NA	**517.12***	-	**523.52***	-	328.85	3645
a6-60	NA	**727.85***	-	NA	-	609.988	5644
a6-72	NA	NA	-	NA	-	1494.96	5115
a7-56	NA	**644.7***	-	**653.11***	-	445.76	5707
a7-70	NA	**849.83***	-	NA	-	1057.64	4609
a7-84	NA	NA	-	NA	-	2257.98	6664
a8-64	NA	**649.4***	-	**654.35***	-	565.246	6567
a8-80	NA	**843.25***	-	NA	-	1137.92	6630
a8-96	NA	NA	-	NA	-	3554.71	5308

References

Babisch, W., et al. (2008). Road traffic noise and cardiovascular risk. *Noise and Health*, *10*(38), 27.

Bongiovanni, C., Geroliminis, N., & Kaspi, M. (2022). A ride time-oriented scheduling algorithm for dial-a-ride problems [arXiv:2211.07347 [cs, math]]. https://doi.org/10.48550/arXiv.2211.07347

Bongiovanni, C., Kaspi, M., Cordeau, J.-F., & Geroliminis, N. (2022). A machine learning-driven two-phase metaheuristic for autonomous ridesharing operations. *Transportation Research Part E: Logistics and Transportation Review*, *165*, 102835. https://doi.org/10.1016/j.tre.2022.102835

Bongiovanni, C., Kaspi, M., & Geroliminis, N. (2019). The electric autonomous dial-a-ride problem. *Transportation Research Part B: Methodological*, *122*, 436–456. https://doi.org/10.1016/j.trb.2019.03.004

Braekers, K., Caris, A., & Janssens, G. K. (2014). Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, *67*, 166–186. https://doi.org/10.1016/j.trb.2014.05.007

Bruglieri, M., Pezzella, F., Pisacane, O., & Suraci, S. (2015). A variable neighborhood search branching for the electric vehicle routing problem with time windows. *Electronic Notes in Discrete Mathematics*, *47*, 221–228.

Caulfield, B., Furszyfer, D., Stefaniec, A., & Foley, A. (2022). Measuring the equity impacts of government subsidies for electric vehicles. *Energy*, *248*, 123588. https://doi.org/10.1016/j.energy.2022.123588

Chassaing, M., Duhamel, C., & Lacomme, P. (2016). An els-based approach with dynamic probabilities management in local search for the dial-a-ride problem. *Engineering Applications of Artificial Intelligence*, *48*, 119–133.

Conrad, R. G., & Figliozzi, M. A. (2011). The recharging vehicle routing problem. *Proceedings of the 2011 industrial engineering research conference*, *8*.

Cordeau, J.-F. (2006). A Branch-and-Cut Algorithm for the Dial-a-Ride Problem. *Operations Research*, *54*(3), 573–586. https://doi.org/10.1287/opre.1060.0283

Cordeau, J.-F., & 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. https://doi.org/10.1016/S0191-2615(02)00045-0

Detti, P., Papalini, F., & 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.

Erdoğan, S., & Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation research part E: logistics and transportation review*, *48*(1), 100–114.

Felipe, Á., Ortuño, M. T., Righini, G., & Tirado, G. (2014). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, *71*, 111–128.

Firat, M., & Woeginger, G. J. (2011a). Analysis of the dial-a-ride problem of hunsaker and savelsbergh. *Operations Research Letters*, *39*(1), 32–35.

Firat, M., & Woeginger, G. J. (2011b). Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Operations Research Letters*, *39*(1), 32–35. https://doi.org/10.1016/j.orl.2010.11.004

Gschwind, T., & 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. https://doi.org/10.1287/trsc.2018.0837

Häll, C. H., Andersson, H., Lundgren, J. T., & Värbrand, P. (2009). The integrated dial-a-ride problem. *Public Transport*, *1*, 39–54.

Häme, L., & Hakula, H. (2013). Routing by ranking: A link analysis method for the constrained dial-a-ride problem. *Operations Research Letters*, *41*(6), 664–669.

Haugland, D., & Ho, S. C. (2010). Feasibility testing for dial-a-ride problems. *Algorithmic Aspects in Information and Management: 6th International Conference, AAIM 2010, Weihai, China, July 19-21, 2010. Proceedings 6*, 170–179.

Hiermann, G., Hartl, R. F., Puchinger, J., & Vidal, T. (2019). Routing a mix of conventional, plug-in hybrid, and electric vehicles. *European Journal of Operational Research*, *272*(1), 235–248.

Hiermann, G., Puchinger, J., Ropke, S., & Hartl, R. F. (2016). The Electric Fleet Size and Mix Vehicle Routing Problem with Time Windows and Recharging Stations. *European Journal of Operational Research*, *252*(3), 995–1018. https://doi.org/10.1016/j.ejor.2016.01.038

Hoche, T., Barth, D., Mautor, T., & Burghout, W. (2020). Charging management of shared taxis: Neighbourhood search for the E-ADARP. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC45102.2020.9294446

Hunsaker, B., & Savelsbergh, M. (2002). Efficient feasibility testing for dial-a-ride problems. *Operations research letters*, *30*(3), 169–173.

Jain, S., & Van Hentenryck, P. (2011). Large neighborhood search for dial-a-ride problems. *International Conference on Principles and Practice of Constraint Programming*, 400–413.

Jorgensen, R. M., Larsen, J., & Bergvinsdottir, K. B. (2007). Solving the Dial-a-Ride problem using genetic algorithms. *Journal of the Operational Research Society*, *58*(10), 1321–1331. https://doi.org/10.1057/palgrave.jors.2602287

Keskin, M., & Çatay, B. (2018). A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & operations research*, *100*, 172–188.

Keskin, M., & Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, *65*, 111–127. https://doi.org/10.1016/j.trc.2016.01.013

Kirchler, D., & Wolfler Calvo, R. (2013). A Granular Tabu Search algorithm for the Dial-a-Ride Problem. *Transportation Research Part B: Methodological*, *56*, 120–135. https://doi.org/10.1016/j.trb.2013.07.014

Lin, J., Zhou, W., & Wolfson, O. (2016). Electric Vehicle Routing Problem. *Transportation Research Procedia*, *12*, 508–521. https://doi.org/10.1016/j.trpro.2016.02.007

Luo, Y., & Schonfeld, P. (2007). A rejected-reinsertion heuristic for the static Dial-A-Ride Problem. *Transportation Research Part B: Methodological*, *41*(7), 736–755. https://doi.org/10.1016/j.trb.2007.02.003

Ma, B., Hu, D., Chen, X., Wang, Y., & Wu, X. (2021). The vehicle routing problem with speed optimization for shared autonomous electric vehicles service. *Computers & Industrial Engineering*, *161*, 107614. https://doi.org/10.1016/j.cie.2021.107614

Macrina, G., Laporte, G., Guerriero, F., & Pugliese, L. D. P. (2019). An energy-efficient green-vehicle routing problem with mixed vehicle fleet, partial battery recharging and time windows. *European Journal of Operational Research*, *276*(3), 971–982.

Macrina, G., Pugliese, L. D. P., Guerriero, F., & Laporte, G. (2019). The green mixed fleet vehicle routing problem with partial battery recharging and time windows. *Computers & Operations Research*, *101*, 183–199.

Malheiros, I., Ramalho, R., Passeti, B., Bulhões, T., & Subramanian, A. (2021). A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem. *Computers & Operations Research*, *129*, 105196. https://doi.org/10.1016/j.cor.2020.105196

Masmoudi, M. A., Hosny, M., Braekers, K., & Dammak, A. (2016). Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, *96*, 60–80. https://doi.org/10.1016/j.tre.2016.10.002

Masmoudi, M. A., Hosny, M., Demir, E., Genikomsakis, K. N., & Cheikhrouhou, N. (2018). The dial-a-ride problem with electric vehicles and battery swapping stations. *Transportation Research Part E: Logistics and Transportation Review*, *118*, 392–420. https://doi.org/10.1016/j.tre.2018.08.005

Masson, R., Lehuédé, F., & Péton, O. (2014). The Dial-A-Ride Problem with Transfers. *Computers & Operations Research*, *41*, 12–23. https://doi.org/10.1016/j.cor.2013.07.020

Melachrinoudis, E., Ilhan, A. B., & Min, H. (2007). A dial-a-ride problem for client transportation in a health-care organization. *Computers & Operations Research*, *34*(3), 742–759.

Montero, E., Riff, M.-C., & Neveu, B. (2014). A beginner's guide to tuning methods. *Applied Soft Computing*, *17*, 39–51. https://doi.org/10.1016/j.asoc.2013.12.017

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

Montoya, A., Guéret, C., Mendoza, J. E., & Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, *103*, 87–110.

Noel, L., Zarazua de Rubens, G., Sovacool, B. K., & Kester, J. (2019). Fear and loathing of electric vehicles: The reactionary rhetoric of range anxiety. *Energy Research & Social Science*, *48*, 96–107. https://doi.org/10.1016/j.erss.2018.10.001

Pan, S., Fulton, L. M., Roy, A., Jung, J., Choi, Y., & Gao, H. O. (2021). Shared use of electric autonomous vehicles: Air quality and health impacts of future mobility in the united states. *Renewable and Sustainable Energy Reviews*, *149*, 111380. https://doi.org/10.1016/j.rser.2021.111380

Parragh, S., Pinho de Sousa, J., & Almada-Lobo, B. (2014). The Dial-a-Ride Problem with Split Requests and Profits. *Transportation Science*. https://doi.org/10.1287/trsc.2014.0520

Parragh, S. N. (2011). Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, *19*(5), 912–930.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, *37*(6), 1129–1138. https://doi.org/10.1016/j.cor.2009.10.003

Parragh, S. N., & Schmid, V. (2013). Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, *40*(1), 490–497. https://doi.org/10.1016/j.cor.2012.08.004

Pfeiffer, C., & Schulz, A. (2022). An ALNS algorithm for the static dial-a-ride problem with ride and waiting time minimization. *OR Spectrum*, *44*(1), 87–119. https://doi.org/10.1007/s00291-021-00656-7

Rekiek, B., Delchambre, A., & Saleh, H. A. (2006). Handicapped person transportation: An application of the grouping genetic algorithm. *Engineering Applications of Artificial Intelligence*, *19*(5), 511–520.

Ritzinger, U., Puchinger, J., & Hartl, R. F. (2016). Dynamic programming based metaheuristics for the dial-a-ride problem. *Annals of Operations Research*, *236*, 341–358.

Rojas-Rueda, D., Nieuwenhuijsen, M. J., Khreis, H., & Frumkin, H. (2020). Autonomous vehicles and public health. *Annual Review of Public Health*, *41*(1), 329–345. https://doi.org/10.1146/annurev-publhealth-040119-094035

Ropke, S., Cordeau, J.-F., & 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., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, *40*(4), 455–472. https://doi.org/10.1287/trsc.1050.0135

Savelsbergh, M., & Sol, M. (1995). The General Pickup and Delivery Problem. *Transportation Science*, *29*, 17–29. https://doi.org/10.1287/trsc.29.1.17

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

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record Breaking Optimization Results Using the Ruin and Recreate Principle. *Journal of Computational Physics*, *159*(2), 139–171. https://doi.org/10.1006/jcph.1999.6413

Shao, S., Guan, W., & Bi, J. (2018). Electric vehicle-routing problem with charging demands and energy consumption. *IET Intelligent Transport Systems*, *12*(3), 202–212.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems.

Su, Y., Dupin, N., Parragh, S. N., & Puchinger, J. (2022). A Column Generation Approach for the Electric Autonomous Dial-a-Ride Problem [arXiv:2206.13496 [math]]. Retrieved July 9, 2022, from http://arxiv.org/abs/2206.13496

Su, Y., Dupin, N., & Puchinger, J. (2021). A deterministic annealing local search for the electric autonomous dial-a-ride problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2023.02.012

Tang, J., Kong, Y., Lau, H., & Ip, A. W. (2010). A note on "efficient feasibility testing for dial-a-ride problems". *Operations Research Letters*, *38*(5), 405–407. https://doi.org/https://doi.org/10.1016/j.orl.2010.05.002

Van Kempen, E., & Babisch, W. (2012). The quantitative relationship between road traffic noise and hypertension: A meta-analysis. *Journal of hypertension*, *30*(6), 1075–1086.

Wang, W., & Zhao, J. (2023). Partial linear recharging strategy for the electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, *308*(2), 929–948.

Zhang, S., Gajpal, Y., Appadoo, S., & Abdulkader, M. (2018). Electric vehicle routing problem with recharging stations for minimizing energy consumption. *International journal of production economics*, *203*, 404–413.

Zhao, M., & Lu, Y. (2019). A Heuristic Approach for a Real-World Electric Vehicle Routing Problem [Number: 2 Publisher: Multidisciplinary Digital Publishing Institute]. *Algorithms*, *12*(2), 45. https://doi.org/10.3390/a12020045