Exploring the Multi-Objective Dial-A-Ride Problem: An Analysis of Genetic Algorithms and MIP

Wytze Elhorst

Delft University of Technology

TU Delft

by

Wytze Elhorst

Student number:	4306228	
Project duration:	November, 2022 – Jan, 2024	
Thesis committee:	Prof. Neil Yorke-Smith	TU Delft, supervisor
	Prof. Pradeep Murukannaiah,	TU Delft

This thesis is confidential and cannot be made public until 05-01-2024.

TU Delft

Abstract

The Multi-Objective Dial-a-Ride Problem (DARP) poses significant challenges in the field of transportation optimization, requiring the simultaneous optimization of conflicting objectives such as travel costs, emission, and customer ride times. In this research, we analyse two distinct approaches for tackling the Multi-Objective DARP: Mixed-Integer Linear Programming (MIP) solvers and Genetic Algorithms (GAs). Through a series of experiments and performance evaluations on diverse problem instances, we assess the strengths and weaknesses of each method. We compare their efficiency, scalability, and ability to generate Pareto-optimal solutions. Additionally, the study explores the impact of algorithmic variations on the convergence and solution quality of the Genetic Algorithm. The results demonstrate that MIP solvers seem entirely unsuited for the generation of quality Multi-Objective Pareto fronts. Of the Genetic Algorithms, the algorithm extended with our proposed guiding heuristics in its genetic operators, manages to construct the best quality Pareto front, outperforming the other algorithms in both finding the best objective solution values, as well as pareto diversity and convergence. We discuss the practical implications of our findings, offering recommendations for researchers and practitioners in the realm of transportation optimization and emission reduction.

Contents

1 Introduction

1.1. Problem Statement

Climate change poses an imminent threat to our planet, primarily driven by the alarming levels of greenhouse gas emissions, with Carbon Dioxide (CO_2) being a major contributor [41]. To address this crisis, the Paris Agreement has set emission targets within the European Union, aiming for a 55% reduction in greenhouse gas emissions by 2030 [36]. A vital part in achieving these targets is dependant on the transportation sector, which is responsible for 21% of global emissions, with road vehicles alone contributing 15% [40].

One of the most straightforward methods of reducing road vehicle emission is to reduce the amount of vehicles on the road. If one wants to reduce the amount of road vehicles, while still allowing passenger and freight transport demands to be met, without major changes in transport architecture that would allow for different modes of transportation, vehicles will have to transport more passengers and freight. Road vehicles transport an average of only 1.67 passengers [5]. If one in ten vehicles would carry an extra passenger, an estimate of 7.74 billion gallons of petrol would be saved annually [28]. This suggests emission could be reduced by a significant amount by getting people to share their rides.

The Dial-A-Ride problem (DARP) [32], is an NP-hard optimization challenge aiming to determine the optimal routes for a fleet of vehicles based on pick-up and drop-off requests. These requests may involve passengers, freight, or a combination of both. Usual key objectives in solving the DARP include minimizing total travel costs, total route duration, and customer ride/waiting times [3]. This former objective of minimizing travel costs is particularly intriguing. A minimal profit-oriented route efficiently handles multiple requests simultaneously, seeking to maximize the handled travel demand while minimizing vehicles, time, and fuel requirements.

While profit-oriented DARP solutions offer a promising avenue to reduce CO_2 emissions, the question arises: Can this be reduced even further? Introducing emission reduction as an explicit objective in the DARP allows for a focused exploration of eco efficient solutions. It is crucial, however, to balance this with traditional DARP objectives, ensuring that routes remain cost-effective and customer waiting times are minimized. To tackle this multifaceted problem, an algorithm is needed that can integrate these multiple objectives and generate a diverse collection of non-dominated solutions—commonly known as a Pareto front. This front should illuminate the trade-offs between objectives, providing a comprehensive view of the solution space.

Given that the DARP is NP-hard, exact solutions are limited to small instances. Larger instances can only be solved by obtaining an approximation through heuristic algorithms. Thus the question arises whether it is better to simulate a Pareto front using solutions from an exact algorithm or to rely on a heuristic algorithm to approximate the Pareto front in terms of solution quality and computational efficiency.

To solve this problem, we consider two optimization approaches: an exact Mixed Integer Programming model and a Multi-Objective Genetic Algorithm. How can these approaches best be used to obtain a

wide variety of high quality solutions for the multi-objective DARP?

1.2. Research Question

The primary aim of this research is to address the following research question:

> *Which optimization approach, between an exact Mixed Integer Programming model and a Multi-Objective Genetic Algorithm, proves more effective in providing solutions to the multi-objective Dial-A-Ride problem?*

To delve into this overarching question, it will be divided into two sub-questions:

> *Sub-question 1: How effectively do both approaches explore the objective space?*

In addressing this sub-question, we will look at the solution diversity, objective solution quality, convergence speed, and general shape of the Pareto front.

> *Sub-question 2: Can problem-specific heuristics be incorporated within the Genetic Algorithm to improve its performance in addressing the multi-objective Dial-A-Ride problem?*

The overarching goal of this research is to generate an algorithm for the Dial-A-Ride problem that provides a variety of solutions based on the core DARP objectives and the additional objective of minimizing emissions. The algorithm should produce solutions that approximate exact optimal solutions and are well-distributed throughout the objective space, offering distinct and varied solutions. This will allow ride sharing services to choose a solution that best fits their company values and goals.

2

Previous Research

2.1. Problem variations

The Dial-A-Ride problem is a variation of the vehicle routing problem (VRP)[47]. The VRP knows many other variations, some of these can be seen as a sort of stepping stone between the VRP and the DARP, while other variations expand upon the DARP. These variations all share a common goal of finding optimized routes and schedules based on pick up and drop off requests and are all NP-hard problems. The main difference between these can be found in the specifics of what they transport and how the optimal routes are determined. To get a quick overview of what each variation concerns, the most relevant VRP variations, along with some of their sub-variations, are described below.

2.1.1. Vehicle Routing Problem

The basic goal of a Vehicle Routing problem, as stated in its name, is to determine optimal routes for a set of vehicles to serve a group of customers. This description is kept vague on purpose, allowing variations of this problem to freely add their own specifics. An example of this is the Pick-Up and Delivery Vehicle Routing Problem (PDVRP), which concerns the distribution and collection of goods to and from customers [45]. This problem adds the first step towards the DARP by introducing the concept of picking up and delivering the same objects in a single route. Another VRP is the Vehicle Routing Problem with Time Windows [31] (VRPTW), which introduces time windows to the to be delivered objects which the final schedule has to comply to. These two concepts (Pick-up and delivery & Time windows) became important factors in later Dial-A-Ride variations.

2.1.2. Dial-A-Ride Problem

The goal of the DARP [14] is to design vehicle routes and schedules that can handle pick up and drop off requests from human passengers. Since this problem involves real people and no freight, more emphasis is put on keeping the passengers satisfied. Time windows, when transporting real people, are a lot shorter compared to VRPs that only concern freight. As a result a DARP has to balance reducing user inconvenience against minimizing the operating costs. Another factor the DARP introduces is that of vehicle capacity. Unlike most freight, people can not be stacked efficiently into vehicles, thus a vehicle can only hold a definite amount of people based on their capacity.

2.2. Algorithms

Most DARP algorithms can be divided into three main parts [14]:

- Dividing the requests into clusters. The requests in these clusters will then be assigned to the same vehicle.
- Determining the route every vehicle takes. This is done by sequencing the requests in each cluster.
- The scheduling of all the different requests of every route.

These steps can either be solved separately or in parallel, which is largely dependant on the algorithm used. A major distinction between DARP algorithms is whether they are static or dynamic algorithms. A static algorithm has full knowledge of all requests in advance, allowing static algorithms to cluster, sequence and schedule separately without worry that changes could occur that would affect a previous step. On the other hand, a dynamic algorithm is able to receive new requests at any point. This means the clustering, sequencing and scheduling must be able to be altered at any point, as the new requests will have to be added to the existing structure.

2.2.1. Single-Vehicle

A Single-Vehicle DARP uses only a single vehicle to handle all requests. Clustering of these requests into multiple routes is thus unnecessary and can be left out. Although using only one vehicle is often inefficient and nonexistent in practice, the Single-Vehicle problem can be used as a sub-step for multi-vehicle algorithms [38]. Solutions for this problem are thus usually designed to be used as a subroutine in more complex algorithms. One of the first solutions to the Single-Vehicle DARP is one presented by Psaraftis [37]. This version uses dynamic programming to generate exact optimal solutions. This version was originally designed for static use, but was also extended to work dynamically as well. Instead of using requests time windows (which is commonly used in modern DARP definitions) this algorithm instead uses maximum position shift, which means the order in which the requests are handled in the route can only alter so much from the order in which these request were recieved. Later variations of this approach [38] do incorporate the more modern time windows. While both these algorithms are able to generate global optimal solutions, the main issue lies in their run time complexity of $\mathcal{O}(n^3\ 3^n)$, with n being the amount of customers/requests. Due to this the largest instance that had been solved only had 9 requests. While this number of requests is far too low to function as a stand alone algorithm, it could still be used in conjunction with a clustering algorithm that divides the request in small enough groups.

Sexton and Bodin [44] ran with this idea of using the Single-Vehicle DARP after clustering a multi-vehicle DARP. This algorithm uses an insertion heuristic to solve the routing problem and then solves the associated scheduling problem. They iterate between these two parts using Benders decomposition [46]. This division makes the algorithm able to handle larger instances, with it being tested on several real-life cases with up to 20 customers.

Another dynamic programming approach by Desrosiers, Dumas, and Soumis [19] reformulates the problem using integer programming. This approach allows for the utilization of constraints, making the algorithm a lot more efficient by removing infeasible states. Algorithms of up to 40 requests have been solved using this approach and, while not tested, the author claims theoretically much larger problems could even be solved.

A more modern approach [23] uses an adaptive insertion algorithm which allows the algorithm to scale to any problem size. At low problem sizes, when inserting a new customer to the existing route, the algorithm will look at all feasible sequences, generating a global optimal solution. When the problem size gets too large, the amount of feasible sequences grows exponentially, thus the algorithm adapts by only looking at a certain amount of feasible solutions for each new insertion, resulting in a locally optimal solution.

2.2.2. Multi-Vehicle

Being able to use multiple vehicles, Multi-vehicle DARPs are a lot more useful in practice, as they are able to satisfy a larger and more realistic demand. As a result many different approaches have been developed to tackle these multi-vehicle problems. Furthermore these approaches tend to divide the overall task into two distinct phases: clustering and scheduling. Meaning a solution usually consists of a combination of multiple algorithms, thus getting a clear overview of all algorithms and their effectiveness can get quite complicated. Luckily there are papers that specifically aim to give an overview of all these existing approaches. Cordeau and Laporte [15] give an overview of everything up to 2007 and Ho et al. [24] continue showcasing further research developments since then. Still, too many approaches exist to go over all of them, so instead of looking at all particular approaches that have been developed, a larger focus will be on the general algorithms that these approaches use, as many approaches seem to employ similar algorithms. Only the particularly interesting and effective approach will then be explored

to a larger extend.

Most static multi-vehilce DARPs start by dividing all requests into clusters. The goal of these clusters is to sort the requests in such a way that for each cluster a efficient route can be constructed that is feasible to complete for a single vehicle. Hence a cluster ideally contains requests that share similar locations and time windows. After the creation of the clusters the routes can then be scheduled. Since the routes are only meant for a single vehicle, algorithms used to solve the single-vehicle DARP can be used for this step. An example of this can be found by Bodin and Sexton [6] using their previous single-vehicle algorithm [44] for this exact purpose. The most commonly used algorithms to generate clusters and/or schedules are variations of heuristic insertion algorithm [20, 4], set partitioning using branch-and-cut algorithms [7], genetic algorithms [39, 29], Tabu Search [13, 34], Large & Variable Neighborhood Search [43, 35] and Simulated annealing [8].

One thing to note is that most approaches tend to use heuristics and are therefore approximation algorithms. The most notable exceptions to this are the branch-and-bound algorithms, which are exact algorithms and thus deliver an optimal solution. The price of an exact solution comes in the form of its complexity, which is exponential in the worst case as the DARP is NP-hard. One of the earlier exact solution approaches by Ropke, Cordeau, and Laporte [42] manages to solve instances containing up to 8 cars and 96 requests. Interestingly, Ho et al. [24] show that currently the largest solved DARP instance by exact methods is still on 8 cars and 96 requests [22]. This shows that these exact methods are limited to solve instances of this size or smaller. This approach by Gschwind and Irnich [22] which uses a Branch-and-Price-and-Cut algorithm has a run time of just under 15 minutes, making it most suitable for static algorithms where once the routes are determined, they will not be altered. For dynamic scenarios where rapid changes to the schedule need to occur, exact methods still prove to be insufficient on their own. So far, using an exact approach to solve the multi-objective DARP seems to have barely been researched [24]. Although this is likely due to most solving methods of these exact approaches (MIP, branch-and-bound, etc) being specifically tailored to minimize a single objective. For Multi-objective problems, the solver could run into issues due to the more complex trade-offs present between the multiple objectives.

2.3. Model & Objectives

Since there are many variations of the DARP as well as a multitude of algorithms capable of producing a solution, a single definitive model of the DARP does not exist. There are, however, certain characteristics that are so inherent to the problem that almost any variation and algorithm will make use of these. The data used as input for the problem contains some of these common characteristics. As many DARP use a list of available vehicles, a list of customer requests containing locations (both for pick-up and drop-off requests), as well as some sort of mechanism to enforce a time limit, such as a list of time windows for each request. Another shared characteristic is the output: a route for each vehicle in the form of a list containing the locations of all requests that particular vehicle has to handle in order, both starting and ending at the depot. In order to not only generate a feasible route list that adheres to the time constraints, but also a (close to) optimal one, these approaches aim to minimize some sort of objective. In this section these different objectives will be analysed to find out their relevance to our objective of minimizing CO_2 emissions.

2.3.1. Route Duration, Distance & Profit

Most approaches use the objective of minimizing either the overall route duration [19] or the route distance [13]. Another objective is the maximization of profit [18]. These three objectives provide very similar solutions, as the shortest route is usually also the fastest. Maximizing profit can also be seen as minimizing operational costs. Since the operational costs are largely the wage of the drivers and the cost of fuel [7], the shortest route generally uses the least gas and the fastest route allows drivers to serve the most amount of customers per hour. Hence the similarities between these objectives. Since fuel usage is also directly correlated to CO_2 emission, these objectives are already very much aligned with the objective of emission reduction. However, none of these objectives explicitly align with the minimization of fuel usage and are at best merely an approximation, as there are a multitude of other factors that ultimately determine fuel usage and thus emission. Regardless, it could be interesting to compare the eco-efficiency of these approximations with more specialized approaches.

2.3.2. Customer Satisfaction

Another common DARP objective is to minimize the dissatisfaction of the customers (or maximize the satisfaction). The exact meaning of customer satisfaction can vary, but it commonly means to minimize the waiting and travel time for each customer or the difference between desired and actual drop-off times. This objective seem to always be combined with another objective, as is the case for Psaraftis [38] and Sexton and Bodin [44] which both minimize a combination of customer dissatisfaction and overall route duration. Since this objective is seemingly never used on its own, one can conclude that customer satisfaction by itself does not provide adequate solutions. The main benefit of including customer satisfaction is to make the ride service more appealing to the customer. Looking at the survey by Cordeau and Laporte [15] an interesting observation can be made that customer dissatisfaction is more common in earlier approaches. A possible reason for this could be that in the early days ride sharing services were still novel, so these services had to be more appealing to customers in order to attract them. While this objective does not directly align with the minimization of emission, disregarding this objective entirely could result in customers preferring to use other ride services instead.

2.3.3. Emission

There are already a couple of DARPs with the objective of directly minimizing emissions. While a lot less common compared to the other objectives, regard for this objective has risen over the last few years especially. Hu, Zheng, and Liao [25] created a multi-objective model which includes fuel consumption as one of its three objectives, the other two being operational cost and service quality, which they later improved upon [26]. In order to bind these three quite varying objectives to a single parameter, they chose to implement speed levels for the vehicles. A higher speed level would benefit the service quality (by reducing customer waiting time) and reduce the operational costs (by reducing travel time). A lower speed level would reduce fuel consumption as travelling at lower speeds is more fuel efficient. Besides this multi-objective model they also tested each of these objectives individually, the results of their models is shown in Figure2.1.

	Travel time	Waiting time	Fuel consumption	S-1
Model	A	B	C	A33_B33_C33
Objective Value				
Travel time (min)	94.29	142.98	186.17	125.65
Waiting time (min)	52.44	0.97	33.66	11.94
Fuel consumption (L)	2.14	2.69	0.99	1.57

Figure 2.1: Table displaying the main results of four different models by Hu, Zheng, and Liao [26]. Model A, B & C have a single objective of minimizing travel time, waiting time or fuel consumption respectively. Model S-1 minimizes the combination of these three objectives with equal weights

Model C and Model S-1 are of particular interest as these include fuel consumption as their objective. Comparing these two shows why a multi-objective model could be preferred over a model with fuel consumption as its sole objective. While model C does logically have the least amount of fuel consumed, it heavily compensates for that by having a significant increase in both travel time and waiting time. Model A and B share the same tendency, minimizing their objective at the cost of the other parameters. Model S-1 has the relative lowest combination of all three. The weights of these objectives can be further altered to fine tune the model based on which objective is deemed the most important. Since this importance is quite subjective it mainly depends on outside factors, for example rises in fuel cost could make fuel consumption more important and bad customer reviews probably indicate more weight should be put behind service quality.

Instead of using multiple objectives, an alternative might be to put constraints on certain factors to keep a limit on how much these factors can be ignored in favour of the main objective. Chen, Hu, and Wu [10] came up with an approach for an exact solution that had two objectives: Minimizing travel time & fuel consumption. Additionally, certain constraints are set such as time window length and maximum ride duration to keep their solution in check. Furthermore, they deploy a traffic simulation model to estimate emissions. As can be expected both travel time and fuel consumption are reduced as the constraints are loosened. So while this approach also requires a balance between these main DARP objectives, setting constraints, such as allowing customers to wait no more than X minutes, it could be

easier to manage compared to arbitrarily altering weights until the desired values are achieved.

A non exact approach is that by Abedi et al. [2], which uses a heuristic (tabu search). What is interesting about this approach is not its objectives, which are minimizing Operating costs and fuel consumption, but rather they have some alternative ways of reducing this fuel consumption. Since their approach is particularly aimed at elderly and disabled customers, who sometimes require special seating, they deploy heterogeneous vehicles (different types and sizes). Thus for each type of vehicle they utilize, they determine how much fuel is used to travel a certain distance. While this is very specific to their target demographic, a more general version of this could be used by simply regarding vehicle sizes, as smaller vehicles drive more efficiently, at the cost of less capacity.

These approaches all seem to have their own methods for calculating fuel consumption and there does not seem to be a general best method, so in section 2.4 a more in-depth look at how to estimate fuel consumption will be had. An approach that uses only the single objective of minimizing fuel consumption also seems to not exist yet, besides the poorly optimized model used in Hu, Zheng, and Liao [26]. Even versions using constraints seem to still use at least two objectives, but perhaps a certain combination of constraints exist that allow a single-objective fuel reduction model to thrive.

2.4. Fuel Consumption

Section 2.3.3 shows that each existing research concerning eco-efficient DARPs had their own methods of measuring fuel consumption. Most of these methods seem to use simple methods, as exact fuel consumption calculations are far too complex, and are thus estimations instead of exact calculations. Furthermore they tend to only look at one aspect for possible fuel consumption reduction, such as the study by Abedi et al. [2] which only looks at utilizing different types of vehicles with different rates of fuel consumption at the cost of capacity. Hu, Zheng, and Liao [26] on the other hand only look at the difference speed makes in fuel consumption. While other solutions use traffic simulations to get an estimate. A study by Demir, Bektaş, and Laporte [17] reviews multiple studies concerning fuel consumption and concludes that there are many factors which affect fuel consumption for land vehicles. However, these factors rely on a lot of specific circumstances regarding the vehicle, driver and environment. Many of these factors are either unknown or so insignificant that they should only be regarded if it is vital to obtain the exact value. A good approximation of fuel usage can already be obtained by just looking at vehicle types alongside specific driving (speed) modes [21].

2.5. Multi-Objective Optimization

Multi-objective optimization (MOO) algorithms aim to solve problems that have multiple objectives (minimizing some values). Usually these objectives are conflicting, meaning the improvement of one objective comes at the cost of another objective. For these cases, the goal is to find a set of solutions that contain the best possible trade-offs between the objectives. This set of solutions, called the Pareto front, contains all solutions where one of the objectives can not be further minimized without increasing another one of the objectives. This front becomes a valuable tool for evaluating the exact costs, both monetary and environmental, associated with reducing the time customers spend waiting. Dial-a-ride services can use this information to strike a balance based on their specific priorities for each objective.

2.5.1. Weighted Aggregation

In the context of this specific variation of the Dial-a-Ride problem, three conflicting objectives must be minimized: operational costs, total emissions, and customer travel duration. Weighted aggregation provides a method to combine these objectives into a single objective value, aiming for overall minimization. This involves assigning weights to each objective and summing them to obtain a unified objective for optimization. Hu, Zheng, and Liao [27] solve a multi-objective model for the DARP using weighted aggregation. While achieving good results, their model only requires a single solution, instead of a full front of solutions. Their model, for instance, does not explicitly reveal how reductions in emissions may influence operational costs. However, by shifting the weights between runs to put more emphasis on certain objectives, multiple 'optimal' solutions can be obtained. These solutions could then be combined to simulate a pseudo-Pareto front.

2.5.2. Genetic Algorithms

On the other hand, real multi-objective optimization algorithms specifically aim to generate a Pareto front. Genetic Algorithms are some of the most used types of Multi-Objective optimization algorithms for the Dial-A-Ride Problem. A popularly used genetic algorithm is the Non-dominated Sorting Genetic Algorithm (NSGA). A NSGA is specifically tailored for and commonly used to solve multi-objective optimization problems [4]. While many approaches [48, 30] use NSGA-II, since it is faster [9] than both its predecessor (NSGA) and successor (NSGA-III), NSGA-III is the best at generating more diverse solutions sets [9]. Thus NSGA-III is ultimately the best suited for the goal of generating a diverse Pareto front.

2.5.3. Hybrid Heuristics

To tailor multi-objective Genetic Dial-A-Ride models to the specific constraints of DARP, many studies include additional heuristics, such as local search [33, 30] or other problem specific alterations to the genetic operators [48]. These heuristics guide the algorithm toward feasible and superior solutions, enhancing the adaptability of the algorithm to the intricacies of DARP.

Conclusively, the integration of weighted aggregation and multi-objective genetic algorithms, especially NSGA-III, presents a robust approach to solving the complex challenges posed by the DARP. These approaches offer valuable insights for dial-a-ride services to make informed decisions, considering diverse trade-offs in objectives.

3

Models & Algorithms

3.1. MILP Model

Several mixed integer linear programming models already exist for the DARP. However, most of these models describe either a base version of the DARP or their own specific alteration. Thus our model uses the three-index formulation from Cordeau and Laporte [15] as a basis, but is expanded to include the additional objectives. The method for calculating the cost has also been altered to be the sum of the drivers wages and the cost of the required fuel, instead of simply being based on the distance travelled.

Parameters	Explanation
n	Number of requests.
K	Number of vehicles.
L	Maximum route length.
V	Set of integers from 0 to $2n+1$ containing all requests. The first and final entry represent the depot, the entries in between are the union of $P\&D$
P	A set from 1 to n containing the pickup requests.
D	A set from $n+1$ to $2n$ containing the drop-off requests.
Q^k	The maximum capacity of vehicle k.
q_i	The load of request i (negative for drop-off requests).
d_i	Service time of request i.
e_i	Earliest available pickup for request i.
l_i	Latest possible drop-off for request i.
t_{ij}	Time (m) it takes to go from request i to j.
e^k_{ij}	Emission cost of vehicle k going from request i to j.
$w1$	Weight for objective 1: Operational cost.
$w2$	Weight for objective 2: Emission.
$w3$	Weight for objective 3: Customer ride duration.

Table 3.1: Parameters of the MILP Model

The model parameters and variables are explained in tables 3.1 & 3.2. The mathematical formulation of the model is as follows:

$$\text{Minimize} \quad c^* w_1 + e^* w_2 + r^* w_3 \tag{3.1}$$

$$\text{Subject to} \quad e^* = \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} e_{ij}^k x_{ij}^k \tag{3.2}$$

$$r^* = \sum_{k \in K} \sum_{i \in P} \sum_{j \in V} r_i^k x_{ij}^k \tag{3.3}$$

$$c^* = 16.15 c^w / 60 + 2.11 e^* / 2300 \tag{3.4}$$

$$c^w = \sum_{k \in K} \sum_{i \in V} u_{2n+1}^k x_{i,2n+1}^k - \sum_{k \in K} \sum_{i \in V} u_0^k x_{0i}^k \tag{3.5}$$

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k = 1 \qquad (i \in P) \tag{3.6}$$

$$\sum_{i \in V} x_{0i}^k = \sum_{i \in V} x_{i,2n+1}^k = 1 \qquad (k \in K) \tag{3.7}$$

$$\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{n+i,j}^k = 0 \qquad (i \in P, k \in K) \tag{3.8}$$

$$\sum_{j \in V} x_{ji}^k - \sum_{j \in V} x_{ij}^k = 0 \qquad (i \in P \cup D, k \in K) \tag{3.9}$$

$$u_j^k \geq (u_i^k + d_i + t_{ij}) x_{ij}^k \qquad (i, j \in V, k \in K) \tag{3.10}$$

$$w_j^k \geq (w_i^k + q_j) x_{ij}^k \qquad (i, j \in V, k \in K) \tag{3.11}$$

$$r_i^k \geq u_{n+1}^k - (u_i^k + d_i) \qquad (i \in P, k \in K) \tag{3.12}$$

$$u_{2n+1}^k - u_0^k \leq T_K \qquad (k \in K) \tag{3.13}$$

$$e_i \leq u_i^k \leq l_i \qquad (i \in V, k \in K) \tag{3.14}$$

$$u_j^k x_{0j}^k \leq e_i \qquad (j \in V, k \in K) \tag{3.15}$$

$$t_{i,n+i} \leq r_i^k \leq L \qquad (i \in P, k \in K) \tag{3.16}$$

$$max(0, q_i) \leq w_i^k \leq min(Q_k, Q_k + q_i) \qquad (i \in V, k \in K) \tag{3.17}$$

$$x_{ij}^k = 0, 1 \qquad (i, j \in V, k \in K) \tag{3.18}$$

(1) is the objective function which uses weighted aggregation to minimize the values of the three objectives. (2) to (5) are the calculations for total emission, ride duration, operational costs and wages respectively. Constraints (6) to (9) ensure the vehicle routes cover all requests once and each route has corresponding pickup and drop-off requests and start and end at the depot. Constraints (10) to (13) determine the schedules and vehicle loads. (14) till (17) check whether these schedules have feasible times and loads.

Variables	Explanation
x_{ij}^k	Contains the routes. is equal to 1 if vehicle k travels from i to j, otherwise it equals 0.
u_i^k	Time at which vehicle k starts to service request i.
w_i^k	Load of vehicle k at request i.
r_i^k	Ride duration of customer of request i in vehicle k.
e^*	Total emission.
c^*	Total Operational cost.
r^*	Total ride duration.
c^w	Total cost of wages.

Table 3.2: Variables of the MILP Model

3.2. Genetic Algorithm

While the Genetic Algorithm follows all the same objectives and constraints as the MIP model, the solution output format has been altered to better fit the strengths of the Genetic Algorithm. Instead of the results being a 3D Boolean matrix as introduced by Cordeau and Laporte [15], the solution is instead stored as a list of integers of size 4n, as is done in the algorithm by Zelić et al. [48]. The first 2n integers represent the request their pick up (1 to n) and their corresponding drop off (n+1 to 2n) points. The second half contain the integer ID's of the vehicle assigned to the corresponding request in the first half. Four variations of the Genetic Algorithms have been developed. The differences between the variations are found in their genetic operators. Table 3.3 shows an overview of the different operations used per algorithm.

Table 3.3: The four algorithms and the different operators they use.

Name	Sampling	Crossover	Mutation
Genetic Algorithm (GA)	Simple	Random	Point Swap
Guided Crossover (GC) GA	Smart	Guided	Point Swap
Fully Guided (FG) GA	Smart	Guided	Guided
Half Guided (HG) GA	Smart	Both	Guided

The Guided Genetic operators aims to improve upon the regular GA by having operations more tailored to the multi-objective DARP. The main design philosophy behind the guiding operations to achieve better results are the following:

- A heavy emphasis on making the solutions feasible.
- Guiding the population in the directions of the objectives.
- Diversity in its operations to combat early convergence.
- Keep the runtime linear.

3.2.1. Sampling

The initial simple sampling of candidate solutions, inspired by Zelić et al. [48], goes as follows: First all requests are shuffled into a random order. Then, following this order, for each request all vehicles that have enough capacity to fit the current request are determined. If no such vehicles are available, the request is placed back at the end of the order list. Then for each eligible vehicle a priority value is calculated. This value is the absolute difference between the target time of the request (the midpoint of it time window) and the actual earliest time each vehicle can arrive at this request based on its previous route. If a drop off request is selected, it gets added to the route containing the corresponding pick up request, if the pick up request is yet to be scheduled, the drop off request is simply placed back at the end of the order list.

Smart Sampling

The initial sampling method was mostly focused on creating correct routes that do not exceed capacity limitations. However, since it did not regard any time limitations, it often generated infeasible solutions that broke the time window constraints. In order to generate feasible samples more often a smarter sampling method was implemented, this method takes the time windows into consideration as well. The smart sampling method is very similar to the original sampling method, but with a few changes. First the priority value of empty routes is set to the length of the request time window. This means that if the current request time window differs too much from existing routes, it will prefer to add it to an unused vehicle instead. This should prevent large time gaps in routes. Another change is that when a drop off request is added to the route. Instead of simply adding it at the end of the route, it only gets added if the end of this requests time window is larger than the start of the previous request's window. If this is not the case, the same check will be done against the request that comes before in the route until the check is successful. This prevents the drop off request from being placed in an unfeasible position time-wise. These changes ensure a greater chance of a feasible candidate solution being created.

3.2.2. Crossover

The first crossover method uses a simple Partial Route Exchange. For each vehicle, the child solution randomly picks a parent and copies the same route as the parent's vehicle. If the child inherits a route containing requests it already serves in one of its other vehicles, these requests are removed from the new route. After inheriting a route for each vehicle, any leftover requests that are not contained in the child's route are randomly added to the existing routes until every request is served.

Guided Crossover

The guided crossover method which has more emphasis on keeping the parents route structure and keeping the routes feasible. It takes a single route from one of its parents. The other routes are then inherited from the other parent, with any duplicate requests being removed from these routes. This approach is similar to the crossover method used by Chevrier et al. [11]. The leftover requests are randomly assigned to one of these routes. Where in the route these requests are placed is determined by the time windows of each request. The pick up requests get scheduled in front of the first request encountered whose time window starts at a later time. The drop off for this request is then placed right before the first encountered request whose time window ends later. If non of these are encountered, the request will be scheduled at the end of the route. In order to prevent requests from ending up purely sorted by their time windows. The crossover will add a random slack value to the time window of a request before determining where to put it. By inheriting only one route from one parent and the other routes from the second, only a few duplicate and leftover requests will occur. This means the routes keep can keep more of the parent's exact schedules and the few leftovers can be placed with more precise methods without increasing the run-time complexity too much.

A possible downside to a more structured operator is that it could possible converge too soon. The half guided genetic algorithm is a compromise of both crossover methods. By alternating between using the guided crossover for more stability and the initial random crossover more diverse solutions could possibly be explored.

3.2.3. Mutation

The basic mutation operator selects 2 requests randomly. Then it exchanges both the pick up and the drop off requests of both of these requests.

Guided Mutation

The guided mutation algorithm utilizes multiple mutation methods, that all have their own intended purpose. Each mutation one of these methods is chosen based on their probability. The methods with the highest probabilities are those that aim to diversify the solution. The methods whose aim is to optimize certain aspects of the solution have a lower probability. This leaves plenty of room for exploration while the optimization methods gently nudge the solution in the direction of (one of) the objectives. Each method is kept relatively simple to keep the algorithm efficient. The methods used for mutation are described in the table below:

Name	Description	Intended Purpose	Probability
Random Swap	Two requests are randomly chosen. Both their pick up and drop off requests are then swapped with one another	Indiscriminate exploration of objective space while keeping the same amount of requests each route	0.3
Random Migration	One request is randomly chosen, its pick up and drop off requests move to a random vehicle at a random location	Indiscriminate exploration of the objective space while changing the amount of requests in each route	0.3

Table continues on next page

Table continued from previous page

Name	Description	Intended Purpose	Probability
Distance Swap	Picks a random request from a random vehicle's route. For each non-depot neighbor of the request, calculate the route distance if these requests were to swap. If none of these swaps reduce the route distance, the original route is kept; otherwise, the swap that causes the largest reduction in distance is done. Corresponding pick up and drop off requests are not swapped.	Guiding the solution towards shorter routes, which directly benefits the objective of reducing emissions.	0.1
Timing Swap	Picks a random request from a random vehicle. If the previously scheduled request on its route has a later time window, the two requests swap. If this is not the case, the next scheduled request on the route is inspected. If its time window starts earlier than the picked request, they swap. Corresponding pick up and drop off requests are not swapped.	Generally guides the solution to have more consistent route schedules, reducing time gaps in the schedule making faster and thus cheaper routes	0.1
Duration reduction	Picks a random request from a non-empty vehicle. If it's a pick up request, check whether the next request scheduled is its corresponding drop off request; if not, swap the two. If the randomly picked request was a drop off request, check the request scheduled before it. If it's not its corresponding pick up request, swap them.	Pushes pick up and drop off requests closer to each other, reducing the ride time of that customer. Guiding the solution towards a lower total customer ride duration	0.1
Route Swap	Two randomly picked vehicles swap their routes.	Since our DARP has non-homogeneous vehicles, differing in both eco-efficiency and capacity. Swapping routes allows the solution to explore the same route schedules with different vehicles.	0.05

Table continues on next page

Table continued from previous page

Name	Description	Intended Purpose	Probability
Max Route Migration	Finds the route whose first and final requests' time windows have the largest difference. Then randomly remove either the first non-depot request alongside its corresponding drop off or the final drop off request and its corresponding pick up. Then pick a random vehicle and add the removed request to this vehicle's route, right before the first request in the schedule whose time window starts later. The drop off request is placed right after the pick up request.	Reduces the length of the longest route, reducing the overall route duration, shorter routes directly reduce the operational costs.	0.05

3.2.4. Repair

The repair function will make sure each vehicle will never exceed it's capacity during its route. To do this, for each route, at every request a check will be done to see if the requests load would exceed the vehicles capacity. If this is the case, the algorithm finds the next scheduled drop off request that has a pick up request earlier in the route. This drop off request is then moved ahead in schedule of the capacity exceeding request. If required, additional drop off requests are moved ahead in the same way until the current request no longer exceeds capacity.

The repair function also checks if any drop off requests appear before its corresponding pick up request. If so, these two are simply swapped.

4

Analysis

4.1. Experimental Setup

The exact MILP model is recreated in Minizinc and uses Gurobi 10.0.1 as its solver. The Genetic algorithm is implemented in Python 3.9, using Pymoo for the NSGA-III algorithm. All experiments are ran on a Windows PC (Intel Core i5-7300HQ CPU @ 2.50GHz with 8gB).

4.1.1. Test Data

The dataset used for the experiments uses data obtained from these [16] existing datasets. This dataset contains simulated DARP instances with spacial and temporal variations. The requests are generated with random locations and start times that fall within the desired service area and duration. Each request has the same time window of 60 minutes. From these different parameters, three scenarios have been created that aim to test different challenges and aspects of the DARP problem. The scenarios are as follows:

- Scenario 1: This scenario covers an area of 20km^2 over 8 hours. Having a small service area & duration to work with means problem instances are more likely to have a lot of feasible solutions. Requests will be closer to one another both geographically and time wise, allowing for many variations within routes to still be feasible. This scenario aims to test how well the algorithm can find and navigate through all these different variations to generate a suitable Pareto front.
- Scenario 2: This scenario covers the same area of 20km^2, but has a longer service time of 24 hours. As a result the requests are spaced further apart on the timeline. Routes are now constraint by their maximum route duration of 8 hours. There is also less variation possible in the order of the requests in the routes, as their time windows are less likely to overlap. This scenario tests how capable the algorithm is at exploring the solution space of instances with stricter time constraints and less feasible solutions.
- Scenario 3: This scenario covers a larger area of 60km^2 over 8 hours. A larger service area means the requests lie further apart from each other. Larger distances take longer to travel, hence instances in this scenario are also more constrained time wise. These larger distances also allow for more significant variations in route distances, putting emphasis on finding routes that can efficiently reach these requests.

The amount of requests per instance ranges from 10 to 100, with 5 to 30 vehicles. These numbers are based on the real data [1] of a ride share company in Austin, TX.

4.2. Results

4.2.1. MIP Solver

The exact MIP model quickly proved to be unsuited for this Multi-Objective instance of the DARP. While it was able to solve very small instances, the MIP failed to generate a solution within a generous time limit of 20 minutes for instances above 12 requests. While the MIP was never expected to work well

on larger instances, the multi-objective model performs significantly worse than its single-objective counterpart, which managed to solve instances up to 36 requests [12]. Table4.1 shows the objective values found for these small instances by both the MIP and the GA. Even with only 400 generations, all but the original genetic algorithm managed to find the optimal solution the MIP gave. Since the MIP could not complete larger instances, further single solution comparison was unfortunately not possible.

Figure 4.1 shows a comparison of the pseudo-Pareto front generated by the MIP and the Pareto front obtained by the FGGA. Out of the 15 points measured by the MIP, 9 turned out to be distinct and only 5 solutions were not dominated by the front of the FGGA. Analysing 4.1 quickly shows the weakness of generating a Pareto front using Weighted Aggregation. While it managed to find a bunch of suitable solutions near the edges of the objective space, it fails to find all the different solutions in between these values. The solver tended to keep picking the same solution with similar weights, instead of finding new solutions with slightly altered values.

The failure of the MIP model to efficiently solve the multi-objective DARP is somewhat expected, considering the lack of existing research indicates that it has not been deemed worth to look into the effectiveness of this approach. Nevertheless if the multi-objective were to perform on par with the single-objective version, there might have still been merit in the approach. Being able to solve instances up to 36 requests is much more in line with the expected number of requests found in practice [1]. This would also have meant we would have been able to calculate how close the approximations of the GAs are to the exact solutions on these larger realistic instances. Unfortunately with the MIP only being able to solve instances of trivial size, it does not allow us to make meaningful conclusions regarding the solution quality of the GAs.

Table 4.1: Comparison of best found solutions between GA & MIP

Dataset			Operational Costs				Emission				Ride Duration			
Sce	**Req**	**Vehicles***	**GA**	**GA+**	**GAS**	**MIP**	**GA**	**GGA**	**FGGA**	**MIP**	**GA**	**GA+**	**GAS**	**MIP**
1	5	0,0,2	79.17	79.17	79.17	79.17	15.59	15.59	15.59	15.59	47	47	47	47
1	8	0,0,2	105.87	94.67	94.67	94.67	24.15	24.56	24.56	24.56	91	91	91	91
1	12	0,1,2	147,59	137,2	137,2	137,2	30,31	28,38	28,38	28,38	93	93	93	93

* * Vehicles are displayed in the format [big, medium, small]

4.2.2. Genetic Algorithms

To compare and analyze the quality and characteristics of the Pareto fronts generated by the GAs, the following metrics are determined:

- The amount of Non Dominated solutions in the Pareto front is the first important factor. A large number of solutions generally means the Pareto front has explored many diverse options. However, only looking at the amount can be misleading, a front with many solutions could largely be dominated by the better quality solutions of a smaller front. Thus the ideal amount of solutions is specific to the instance.
- The Best Solution (BSS) looks for the single 'best' solution in the front. This is determined by weighted aggregation. To determine the weights we look at the avarage objective values and use the weights to normalize them. Operational Costs and Customer Wait Duration seem to share similar values, while the emission value tends to only be about 20% of the other values. So the weights for the former two are both set to 0.2, while the latter has a weight of 1.0. Finally the score is once more normalized by dividing it by the number of requests in the instance. This final score gives an indication of how the quality of a solution is compared to another instance on the same dataset, with a lower score indicating lower (thus better) objective values.
- The S-metric evaluates the diversity of the Pareto front based on how well spaced out the solutions are. It is calculated by dividing the avarage distance (Euclidian) between consecutive solutions by the solution pair with the largest distance between them. The higher the S-metric value, the better spread out the solutions are on the Pareto front.
- The Δ-metric assesses the convergence and diversity of the Pareto front. To obtain the Δ-value, the avarage distance (Euclidian) of consecutive solutions on the front is calculated. A lower Δ-

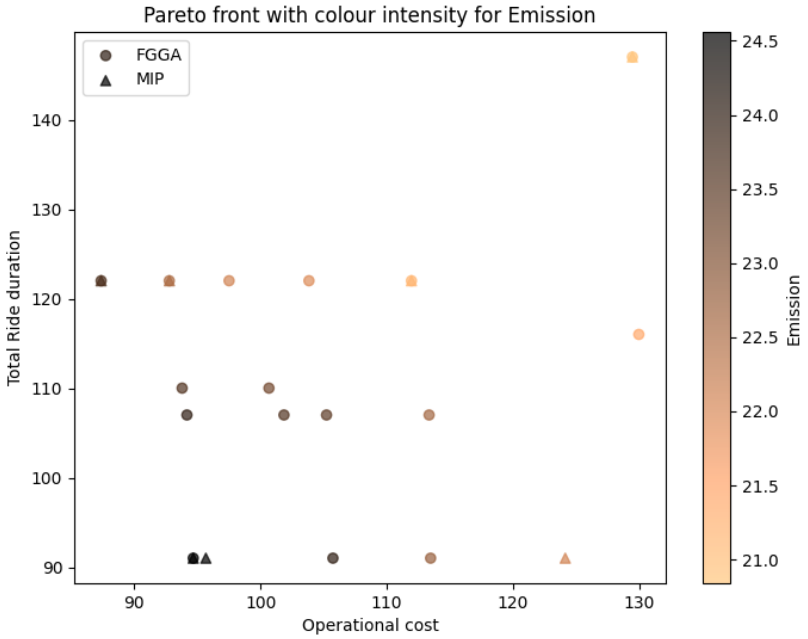


Figure 4.1: Combined Pareto fronts of the MIP and FGGA

metric value indicates the front has better convergence, while a higher score means there is diversity between solutions. A high spread and a low Δ-metric indicate the solution is both well spread and converged.

Pareto front Analysis

Table 4.2 gives an overview of the results obtained by each algorithm per dataset. The first feature of note is the original GA failing to find even a single solution for the larger instances of 75 and 100 requests. Indicating that the random nature of this algorithm struggles to find feasible solutions for these larger instances.

While the HGA is able to generate solutions for each instance, it is consistently outclassed by the GGA. Figure 4.2 shows the Pareto fronts of the same instance generated by both algorithms. The left figure shows both fronts in the same graph, while the right figure shows the combination of both fronts into a single front. While the former might suggest the HGA to explore a larger solution space, the latter figure shows that it is entirely dominated by the front of the GGA.

The FGGA outperforms all other algorithms across the board. Consistently obtaining the lowest Best Solution Score, as can be seen in 4.3a. Its lower Δ-metric compared to the other solutions (especially at larger instances), indicate the front has converged the most, while still maintaining a large number of solutions and high spread. Figure 4.4 shows both the Pareto fronts of both the FGGA and the GGA for the same instance, each solution from the GGA is dominated by the front of the FGGA.

Since the MIP is unable to solve larger instances, the approximation quality can not be exactly calculated by comparing the values in the front to the exact solutions the MIP would have given. However, since the BSS is normalized for each instance size and we do know the Pareto front with n=10 contains the exact optimal solutions. We can compare the BSS of this front to those of larger instances. While the original GA and HGA have BSS that slowly increase, showing their solutions get worse relatively the larger the instance size. The GGA and especially the FGGA see a reduction in their BSS as the instances get larger. While this does not give an absolute answer, as the exact score is dependant on the specific instance and a larger number of requests naturally allow for more efficient routes. It does show that these algorithms can find solutions with objective values comparable to and even exceeding the smaller instances.

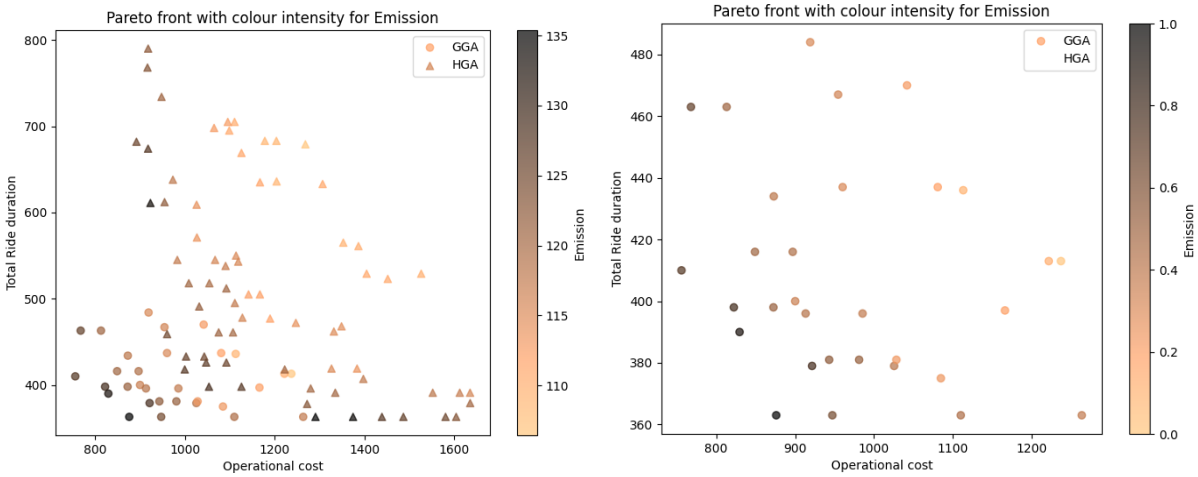
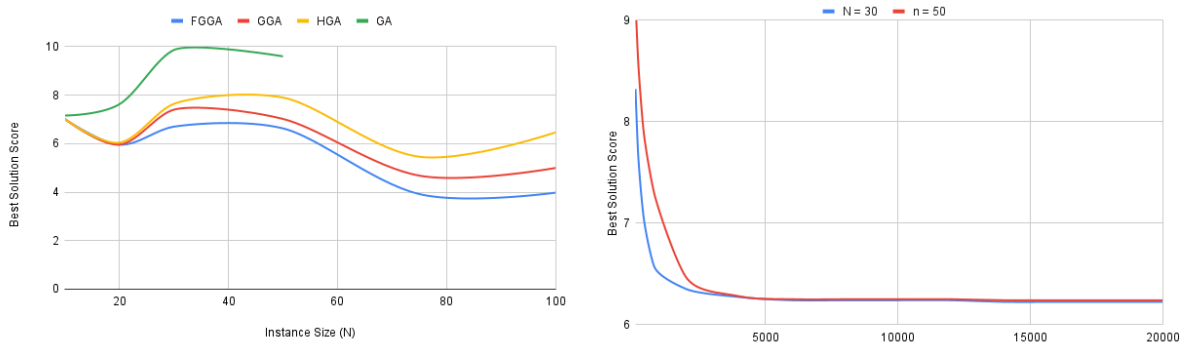


Figure 4.2: Pareto fronts of GGA & HGA displayed seperately (left) and combined (right).



(a) Best solution Score of each GA for different instance sizes.

(b) FGGA: Best Solution Score over the number of generations

Figure 4.3: BSS over instance size (a) and generations (b)

Convergence

Figure 4.3b shows the BSS achieved by the FGGA steadily drops until around 2000 generations, after which it will only decrease an insignificant amount (around 2% score reduction between 2000-20000 generations). Figure 4.5a shows the relation between runtimes for two similar instances of size 30 and 50. This figure shows the linear runtime relation between the two instances. 2000 generations takes around 4-6 minutes for instances between 30-50 generations, increasing linearly as the instance size goes up. Since the MIP is not able to run However, while the BSS stagnates after this point, the density and spread of the Pareto front does continue to increase. Figures 4.6 & 4.7 show the Pareto fronts at 2000, 10,000 and 20,000 respectively. Figure 4.5b shows how the size of the Pareto front (number of solutions) changes over generations. It seems the amount of solutions found increases until around 6000 generations, after which the number stays about the same. However the Pareto front at 20.000 generations is much more evenly spaced across the objective space. Showing that the quality of the Pareto front still increases during this time. For reference, the final Pareto front dominates 40/41 of the solutions at 2000 generations and 101/134 of the solutions at 10.000 generations. The combined Pareto of the instances of 18.000 and 20.000 generations is displayed in figure 4.7b. From the combined 170 solutions, 52 come from the 18,000 front while the remaining 118 originate from the 20,000 front. The fact that this former front contains 52 non-dominated solutions not included in the Pareto front of figure 4.7a indicates that even after 20,000 generations, there are still more optimal solutions available that the algorithm has yet to discover. However with neither the BSS or the size of the Pareto Front changing significantly, there is little to gain from further exploration, especially considering the impractically large amount this would require, as figure 4.5a shows 20,000 generations can already take up to one hour for realistic instance sizes.

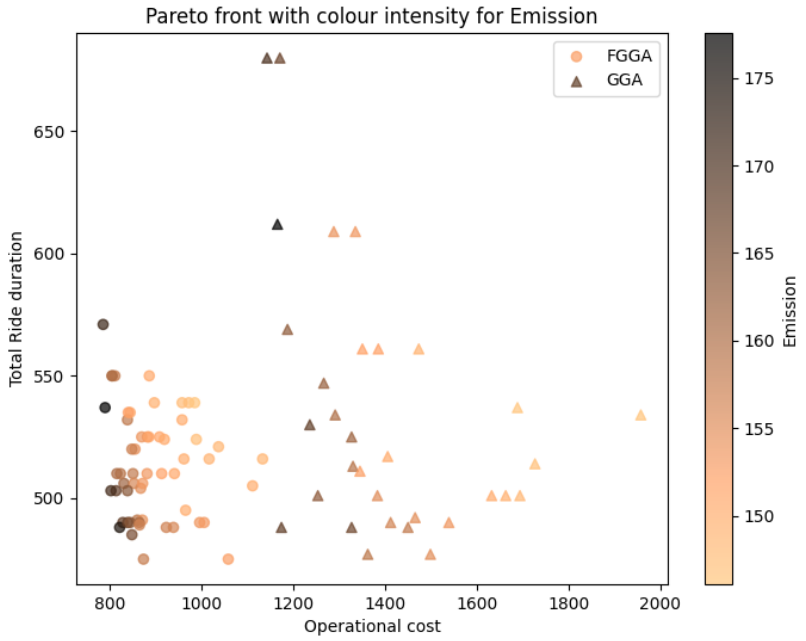
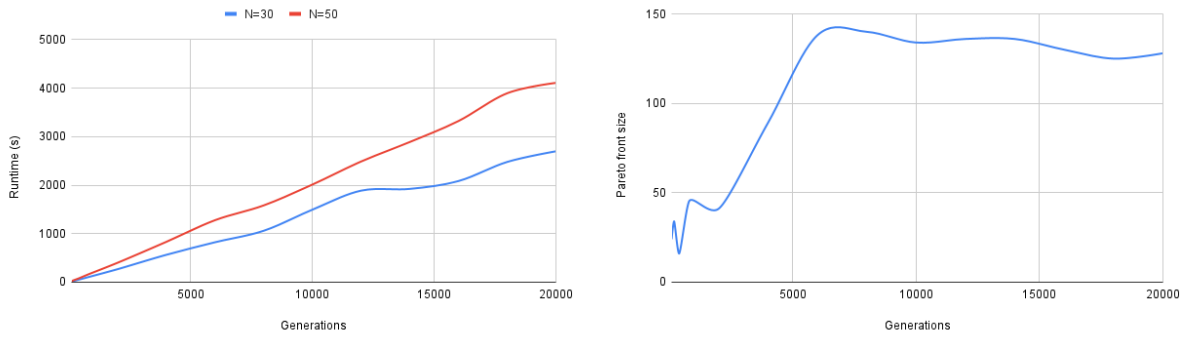


Figure 4.4: Pareto fronts of FGGA & GGA displayed seperately for N=100



(a) Runtime over generations

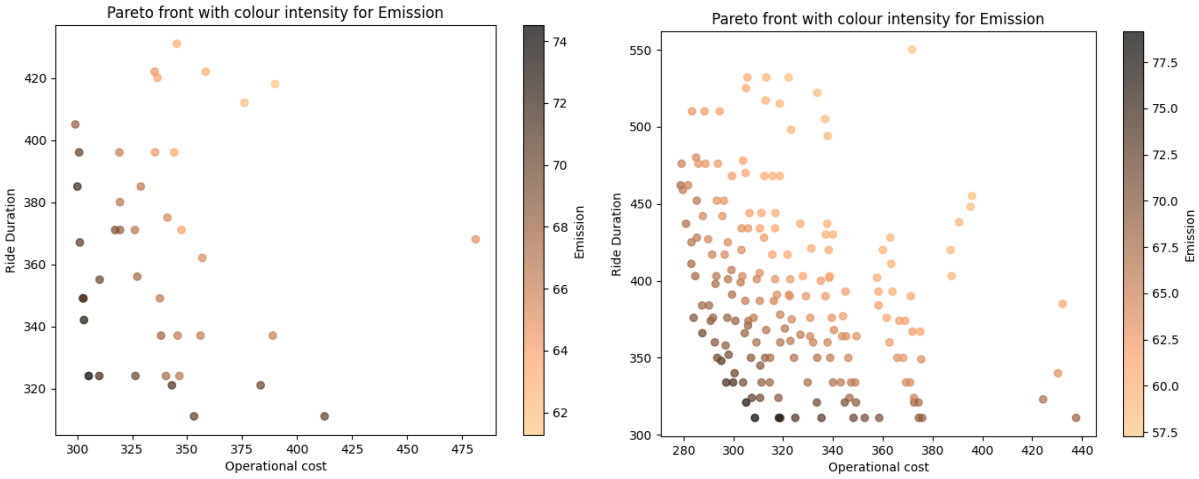
(b) Pareto front size over generations

Figure 4.5: Runtime (a) and Pareto front size (b) over generations for FGGA

Table 4.2: Pareto front Result Data

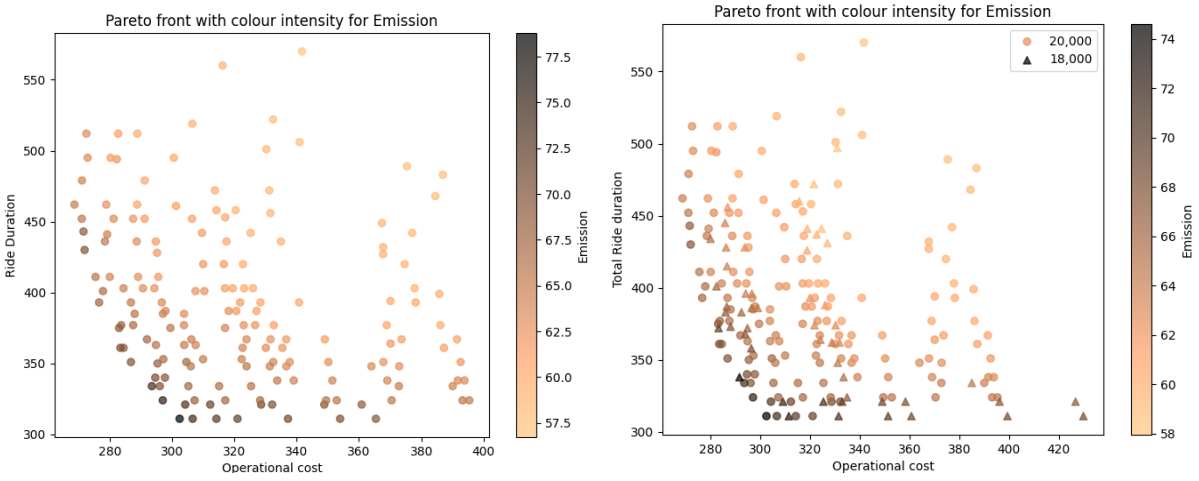
Scenario	N	K	Algorithm	Solutions	BSS	Spread	Δ
1	10	1,2,2	GA	11	7.16	0.93	36.72
			HGA	11	7.02	0.90	40
			GGA	14	7.02	0.87	27
			FGGA	7	7.02	0.91	19
1	20	2,2,2	GA	21	7.62	0.97	93
			HGA	11	6.04	0.94	24
			GGA	9	5.96	0.91	22
			FGGA	37	5.96	0.95	50
1	30	3,3,4	GA	29	9.86	0.98	121
			HGA	22	7.64	0.98	60
			GGA	28	7.4	0.98	61
			FGGA	24	6.7	0.97	87
1	50	8,8,8	GA	20	9.6	0.98	126
			HGA	35	7.9	0.99	137
			GGA	14	7.02	0.98	83
			FGGA	41	6.63	0.98	59
3	50	8,8,8	GA	-	-	-	-
			HGA	28	20.48	0.98	93
			GGA	32	19.15	0.97	87
			FGGA	22	17.81	0.98	83
1	75	8,8,8	GA	-	-	-	-
			HGA	70	5.46	0.99	278
			GGA	31	4.68	0.98	174
			FGGA	58	3.92	0.98	113
2	75	8,8,8	GA	-	-	-	-
			HGA	35	8.8	0.99	326
			GGA	14	7.43	0.98	268
			FGGA	41	6.6	0.99	206
1	100	10,10,10	GA	-	-	-	-
			HGA	49	6.47	0.97	267
			GGA	32	5.0	0.99	235
			FGGA	56	3.98	0.98	99

All instances were run with the following GA parameters: population = 500, offspring = 50, gen = 2000.



(a) 2000 generations **(b)** 10,000 generations

Figure 4.6: Pareto fronts of FGGA (N=30) over 2000 (left) and 10,000 generations (right).



(a) 20,000 generations **(b)** Combined Pareto fronts

Figure 4.7: Pareto front of FGGA (N=30) over 20000 generations (a) and the combined Pareto front of 4.6b and 4.7a (b).

5

Conclusion

In this research we explored the possibilities of using both MIP as well as GAs to effectively find solutions for the Multi-Objective DARP. To determine the effectiveness of these solutions we looked at how well both approaches are able to explore the objective space to generate a diverse Pareto front as well as the individual quality of the solutions found in the front. The results show that the MIP is an unsuitable option for both solving the multi-objective DARP as well as generating Pareto fronts. While the latter was to be expected, as MIP will only look for a single best solution, so the front would have to be manually generated by shifting the weights for each objective. This often resulted in the MIP still finding the same solutions, failing to find neighbouring non-dominated solutions with similar values. However these points could still have served as a reference to measure the approximation of the Pareto fronts generated by the GAs. However the MIP had such issues finding solutions while regarding multiple contrasting objectives, that its efficiency suffered significantly compared to the single-objective model it was based on. As a result it is only able to solve small instances of up to 12 requests, which is too low for non-trivial comparison. For the genetic algorithms we explored how well GAs with more general off the shelf operators perform against our own approach which uses heuristics in its operators that guide the solutions towards each specific objective. The GAs that deploy more general genetic operators might still struggle to find good and feasible solutions for the heavily constrained Dial-A-Ride Problem. Adding heuristics to the genetic operators that guide the solution towards the specific desired objectives and ensuring these operators produce feasible routes gets rid of this issue. Our proposed Fully Guided Genetic Algorithm manages to generate Pareto fronts containing high quality solutions in a couple of minutes, scaling linearly. After which the algorithm continues to improve the quality and diversity of the Pareto front if allowed to continue. Although the improvements seem to be minimal after this point, their presence combined with the inability to measure the exact approximation of the GA, means it is entirely possible that the GA has unknown limitations with regards to exploring the entire objective space. Although the Pareto fronts seem generally diverse, the heuristics in the guided genetic operations could very well fail to explore certain niche solutions as they get guided towards other local optima.

Since the guiding heuristics used are numerous and very problem specific, there is still a lot of room for further adjustments and fine tuning to other problems. In future research, these adjustments could be implemented to allow it to run on the many other variations of the DARP. Although this would require some altering of the operators based on the new specific problem objectives, it does mitigate the need for an exact solution to compare it to, as this would allow (part of the) the algorithm to be tested on existing benchmark data sets. Since there is no exact measure of the approximation of the GAs, the algorithm For a practical application, the algorithm can be adjusted to accurately reflect the specific parameters of a real Ride Sharing company. This Ride Sharing company can use the resulting Pareto front to pick a solution based on their own value preference for the objectives. Allowing them to set limits for what they want for these values and still having multiple route variations they can choose from that have these limits.

References

[1] Apr. 2017. URL: https://data.world/andytryba/rideaustin.

[2] Mehdi Abedi et al. “A Regional Multi-Objective Tabu Search Algorithm for a Green Heterogeneous Dial-A-Ride Problem”. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. 2019, pp. 2082–2089. DOI: 10.1109/CEC.2019.8790003.

[3] Niels Agatz et al. “Optimization for dynamic ride-sharing: A review”. In: *European Journal of Operational Research* 223.2 (2012), pp. 295–303. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2012.05.028. URL: https://www.sciencedirect.com/science/article/pii/S0377221712003864.

[4] Majid Aldaihani and Maged M. Dessouky. “Hybrid scheduling methods for paratransit operations”. In: *Computers & Industrial Engineering* 45.1 (2003), pp. 75–96. ISSN: 0360-8352. DOI: https://doi.org/10.1016/S0360-8352(03)00032-9. URL: https://www.sciencedirect.com/science/article/pii/S0360835203000329.

[5] *Avarage Vehicle Occupancy*. 2009. URL: https://nhts.ornl.gov/tables09/fatcat/2009/avo%5C_TRPTRANS%5C_WHYTRP1S.html.

[6] Lawrence Bodin and Thomas Sexton. “The multi-vehicle subscriber dial-a-ride problem”. In: *TIMS Studies in Management Science* 26 (Jan. 1986).

[7] R. Borndörfer et al. “Telebus Berlin: Vehicle Scheduling in a Dial-a-Ride System”. In: *Computer-Aided Transit Scheduling*. Ed. by Nigel H. M. Wilson. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 391–422. ISBN: 978-3-642-85970-0.

[8] Kris Braekers, An Caris, and Gerrit K. Janssens. “Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots”. In: *Transportation Research Part B: Methodological* 67 (2014), pp. 166–186. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2014.05.007. URL: https://www.sciencedirect.com/science/article/pii/S0191261514000800.

[9] Pranava Chaudhari et al. “Comparison of NSGA-III with NSGA-II for multi objective optimization of adiabatic styrene reactor”. In: *Materials Today: Proceedings* 57 (2022). International Chemical Engineering Conference 2021 (100 Glorious Years of Chemical Engineering & Technology), pp. 1509–1514. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2021.12.047. URL: https://www.sciencedirect.com/science/article/pii/S221478532107718X.

[10] Li-Wen Chen, Ta-Yin Hu, and Yu-Wen Wu. “A bi-objective model for eco-efficient dial-a-ride problems”. In: *Asia Pacific Management Review* 27.3 (2022), pp. 163–172. ISSN: 1029-3132. DOI: https://doi.org/10.1016/j.apmrv.2021.07.001. URL: https://www.sciencedirect.com/science/article/pii/S1029313221000683.

[11] Rémy Chevrier et al. “Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: Application to demand responsive transport”. In: *Applied Soft Computing* 12.4 (2012), pp. 1247–1258. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2011.12.014. URL: https://www.sciencedirect.com/science/article/pii/S1568494611004984.

[12] Jean-François Cordeau. “A Branch-and-Cut Algorithm for the Dial-a-Ride Problem”. In: *Operations Research* 54 (June 2006), pp. 573–586. DOI: 10.1287/opre.1060.0283.

[13] Jean-François Cordeau and Gilbert Laporte. “A tabu search heuristic for the static multi-vehicle dial-a-ride problem”. In: *Transportation Research Part B: Methodological* 37.6 (2003), pp. 579–594. ISSN: 0191-2615. DOI: https://doi.org/10.1016/S0191-2615(02)00045-0. URL: https://www.sciencedirect.com/science/article/pii/S0191261502000450.

[14] Jean-François Cordeau and Gilbert Laporte. “The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms”. In: *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1.2 (June 2003), pp. 89–101. ISSN: 1619-4500. DOI: 10.1007/s10288-002-0009-8. URL: https://doi.org/10.1007/s10288-002-0009-8.

[15] Jean-François Cordeau and Gilbert Laporte. “The dial-a-ride problem: models and algorithms”. In: *Annals of Operations Research* 153.1 (Sept. 2007), pp. 29–46. ISSN: 1572-9338. DOI: 10.1007/s10479-007-0170-8. URL: https://doi.org/10.1007/s10479-007-0170-8.

[16] Darppaper. *DARPDATASET*. 2016. URL: https://github.com/Darppaper/DARPDATASET.

[17] Emrah Demir, Tolga Bektaş, and Gilbert Laporte. “A review of recent research on green road freight transportation”. In: *European Journal of Operational Research* 237.3 (2014), pp. 775–793. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2013.12.033. URL: https://www.sciencedirect.com/science/article/pii/S0377221713010175.

[18] Guy Desaulniers et al. “VRP with Pickup and Delivery”. In: Jan. 2002, pp. 225–242. ISBN: 978-0-89871-498-2. DOI: 10.1137/1.9780898718515.ch9.

[19] Jacques Desrosiers, Yvan Dumas, and F. Soumis. “A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-A-Ride Problem with Time Windows”. In: *American Journal of Mathematical and Management Sciences* 6 (Feb. 1986). DOI: 10.1080/01966324.1986.10737198.

[20] Y. Dumas, Jacques Desrosiers, and F. Soumis. “Large scale multi-vehicle dial-a-ride problems”. In: (Jan. 1989).

[21] A Esteves-Booth et al. “A review of vehicular emission models and driving cycles”. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 216.8 (2002), pp. 777–797.

[22] Timo Gschwind and Stefan Irnich. “Effective Handling of Dynamic Time Windows and Its Application to Solving the Dial-a-Ride Problem”. In: *Transportation Science* 49.2 (2015), pp. 335–354. DOI: 10.1287/trsc.2014.0531. eprint: https://doi.org/10.1287/trsc.2014.0531. URL: https://doi.org/10.1287/trsc.2014.0531.

[23] Lauri Häme. “An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows”. In: *European Journal of Operational Research* 209.1 (2011), pp. 11–22. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2010.08.021. URL: https://www.sciencedirect.com/science/article/pii/S0377221710005588.

[24] Sin C. Ho et al. “A survey of dial-a-ride problems: Literature review and recent developments”. In: *Transportation Research Part B: Methodological* 111 (2018), pp. 395–421. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2018.02.001. URL: https://www.sciencedirect.com/science/article/pii/S0191261517304484.

[25] Ta-Yin Hu, Guan-Chun Zheng, and Tsai-Yun Liao. “A multi-objective model for dial-a-ride problems with service quality and eco-efficiency”. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317630.

[26] Ta-Yin Hu, Guan-Chun Zheng, and Tsai-Yun Liao. “Multi-Objective Model for Dial-a-Ride Problems with Vehicle Speed Considerations”. In: *Transportation Research Record* 2673.11 (2019), pp. 161–171. DOI: 10.1177/0361198119848417.

[27] Ta-Yin Hu, Guan-Chun Zheng, and Tsai-Yun Liao. “Multi-Objective Model for Dial-a-Ride Problems with Vehicle Speed Considerations”. In: *Transportation Research Record* 2673.11 (2019), pp. 161–171. DOI: 10.1177/0361198119848417. eprint: https://doi.org/10.1177/0361198119848417. URL: https://doi.org/10.1177/0361198119848417.

[28] Sheldon H. Jacobson and Douglas M. King. “Fuel saving and ridesharing in the US: Motivations, limitations, and opportunities”. In: *Transportation Research Part D: Transport and Environment* 14.1 (2009), pp. 14–21. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2008.10.001. URL: https://www.sciencedirect.com/science/article/pii/S1361920908001168.

[29] R Jorgensen, Jesper Larsen, and K Bergvinsdottir. “Solving the Dial-a-Ride problem using genetic algorithms”. In: *Journal of the Operational Research Society* 58 (Oct. 2007). DOI: 10.1057/palgrave.jors.2602287.

[30] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. “The bi-objective covering tour problem”. In: *Computers Operations Research* 34.7 (2007), pp. 1929–1942. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2005.07.022. URL: https://www.sciencedirect.com/science/article/pii/S0305054805002480.

[31] Brian Kallehauge et al. “Vehicle Routing Problem with Time Windows”. In: *Column Generation*. Ed. by Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. Boston, MA: Springer US, 2005, pp. 67–98. ISBN: 978-0-387-25486-9. DOI: 10.1007/0-387-25486-2_3. URL: https://doi.org/10.1007/0-387-25486-2_3.

[32] Baoxiang Li et al. “The Share-a-Ride Problem: People and parcels sharing taxis”. In: *European Journal of Operational Research* 238.1 (2014), pp. 31–40. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2014.03.003. URL: https://www.sciencedirect.com/science/article/pii/S0377221714002173.

[33] Arnaud Liefooghe et al. “On the Integration of a TSP Heuristic into an EA for the Bi-objective Ring Star Problem”. In: *Hybrid Metaheuristics*. Ed. by María J. Blesa et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 117–130. ISBN: 978-3-540-88439-2.

[34] Emanuel Melachrinoudis, Ahmet B. Ilhan, and Hokey Min. “A dial-a-ride problem for client transportation in a health-care organization”. In: *Computers & Operations Research* 34.3 (2007). Logistics of Health Care Management, pp. 742–759. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2005.03.024. URL: https://www.sciencedirect.com/science/article/pii/S0305054805001188.

[35] N. Mladenović and P. Hansen. “Variable neighborhood search”. In: *Computers & Operations Research* 24.11 (1997), pp. 1097–1100. ISSN: 0305-0548. DOI: https://doi.org/10.1016/S0305-0548(97)00031-2. URL: https://www.sciencedirect.com/science/article/pii/S0305054897000312.

[36] *Paris Agreement*. UNTC XXVII 7.d. Dec. 12, 2015.

[37] Harilaos Psaraftis. “A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem”. In: *Transportation Science* 14 (May 1980), pp. 130–154. DOI: 10.1287/trsc.14.2.130.

[38] Harilaos N. Psaraftis. “An Exact Algorithm for the Single Vehicle Many-to-Many Dial-A-Ride Problem with Time Windows”. In: *Transportation Science* 17.3 (Aug. 1983), pp. 351–357. DOI: 10.1287/trsc.17.3.351. URL: https://ideas.repec.org/a/inm/ortrsc/v17y1983i3p351-357.html.

[39] REKIEK, B and DELCHAMBRE, A and Saleh, Hussain. “Handicapped Person Transportation: An application of the Grouping Genetic Algorithm”. eng. In: *ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE* 19.5 (2006), 511–520. ISSN: 0952-1976.

[40] Hannah Ritchie. *Cars, planes, trains: Where do CO2 emissions from transport come from?* Oct. 2020. URL: https://ourworldindata.org/co2-emissions-from-transport.

[41] Hannah Ritchie, Max Roser, and Pablo Rosado. “CO□ and Greenhouse Gas Emissions”. In: *Our World in Data* (2020). https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions.

[42] Stefan Ropke, Jean-François Cordeau, and Gilbert Laporte. “Models and branch-and-cut algorithms for pickup and delivery problems with time windows”. In: *Networks* 49.4 (2007), pp. 258–272. DOI: https://doi.org/10.1002/net.20177. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.20177. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20177.

[43] Stefan Ropke and David Pisinger. “An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows”. In: *Transportation Science* 40.4 (2006), pp. 455–472. DOI: 10.1287/trsc.1050.0135. eprint: https://doi.org/10.1287/trsc.1050.0135. URL: https://doi.org/10.1287/trsc.1050.0135.

[44] Thomas Sexton and Lawrence Bodin. “Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I. Scheduling”. In: *Transportation Science* 19 (Nov. 1985), pp. 378–410. DOI: 10.1287/trsc.19.4.378.

[45] A. Serdar Tasan and Mitsuo Gen. “A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries”. In: *Computers & Industrial Engineering* 62.3 (2012). Soft Computing for Management Systems, pp. 755–761. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2011.11.025. URL: https://www.sciencedirect.com/science/article/pii/S0360835211003482.

[46] Z Caner Taskın. “Benders decomposition”. In: *Wiley Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Malden (MA)* (2010).

[47] Max van der Tholen et al. “The Share-A-Ride Problem with Integrated Routing and Design Decisions: The Case of Mixed-Purpose Shared Autonomous Vehicles”. In: *Computational Logistics*. Ed. by Martijn Mes, Eduardo Lalla-Ruiz, and Stefan Voß. Cham: Springer International Publishing, 2021, pp. 347–361. ISBN: 978-3-030-87672-2.

[48] Stjepan Zelić et al. “Solving the Dial-a-Ride Problem Using an Adapted Genetic Algorithm”. In: *AIxIA 2021 – Advances in Artificial Intelligence*. Ed. by Stefania Bandini et al. Cham: Springer International Publishing, 2022, pp. 689–699. ISBN: 978-3-031-08421-8.