

Multi-Objective Tabu Search approach for Single Customer Dial-a-Ride Problem in the Tourism Sector

Ana Catarina Thomaz Moura Morais

Master's Dissertation

Supervisor: Professor Maria Teresa Galvão Dias



FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Integrated Master in Industrial Engineering and Management

2018-07-02

Abstract

The problem under research arises in YellowFish Transfers company. YellowFish is an airport private transfers company in Algarve, Portugal and currently allocates its requests manually in a daily basis. This allocation process implies, regarding the booking set for a specific day and the available drivers, the allocation of every request to a vehicle. Although the process tries to be efficient reducing the operational costs, with days that can reach more than 500 requests, the efficiency and the total time of this task can be compromised once it is a completely manual procedure.

The present work models this problem as a single customer Dial-a-Ride Problem (DARP). A DARP is a vehicle routing problem involving humans as the load to be transported. Several works have been published in this area. While common models involve ride-sharing, in this one, only a "single" customer is transported at the time. However, features as multi-depot and heterogeneous fleet are found in the present application as in many other in literature.

Once this framework has a real application in the tourism sector and a company has always several concerns regarding its services, the model tackles not one objective but three. Regarding the solution approach, among the several methods in academia, a Tabu Search meta-heuristic is applied. Besides addressing a multi-objective DARP, the present Tabu Search samples the neighborhood and uses five different local search operators (not operating at the same time). Runs for three different scenarios of the algorithm are performed. One without a maximum idle time of a driver between requests and other two considering it. From those two scenarios, one tackles only the minimization of the kilometers during the Tabu Search process and the other one only the minimization of the salaries variance between drivers. The algorithm is developed and implemented in Python 3.6.5 language with Visual Studio Code as IDE.

After running the algorithm for 9 different days (with a small, medium and large number of requests), with the main goal of kilometers minimizations, the model has proved to be able to find solutions with less kilometers while keeping similar values to unoccupied seats of the vehicle with or without waiting time constraints. As a trade off, for these scenarios, the algorithm underperforms when the salaries variance between drivers is taken into account. The fleet size seems also to be directly related with salaries variance and inversely proportional to the kilometers. Due to the Pareto front solution that is obtained from joining the non-dominated set of solutions of the different scenarios for each instance, the company can select a compromise solution between the number of kilometers and the others objective functions. Finally, the use of proposed methodology increases the company competitiveness, potentially decreases the costs for the clients and reduces the Green-House Gas emissions because the vehicles need to do less kilometers. Another potential advantage related with the present work, is that using an integrated methodology, the company can close the booking 24 hours before the operation day instead of 48 hours currently used.

Acknowledgement

I remember being a child and have those homeworks of writing a composition about "What if...". Today, my "composition" is *What if I had to do my thesis alone?*

Without professor Teresa Galvão and professor Pedro Cardoso who would debug my mistakes? Without Hortênsio Fernandes who would give me the input data and *a priori* constraints?

Without my family who would be there to make me laugh about something stupid and completely random and make me hear strange music? Or give me the wise advices that I need even if I do not want to say that I need them? What pet would wake me up because he wants to eat?

And what if I did not have any friends? Who would give me a ride home after a night of study? Who would support (or mostly not) my crazy ideas? Who would be there side-by-side in good and bad moments?... Wait, these are almost marriage vows... Or not, because as Apicella et al. (2012) stated once, humans are unusual as a species in the extent to which they form longstanding, non-reproductive unions with unrelated individuals – namely, we have friends. With some I have played basketball, with others I spent my high school, with others my Erasmus, others I just procrastinated in *Gestionis Salonis* and, finally, with a really annoying group of people that loves polls, I spent my five years of university. From this last *pantagruélico* group of people, I must highlight Leque that was unlucky enough to have to work with me 95% of the days in the past 5 months.

These professors, family and friends, like variables in a program, have different natures, but without them the program would not run. Nevertheless, unlike what sometimes happens with a computer program, I just hope to have enough memory to keep them all in my heart. Last but not least, without a hardware it is impossible to run a software, so I must definitely thank to the coffee machine of the second floor of the industrial engineering department.

It is clear to me at this point: if I had to do this alone, it would be impossible. I wholeheartedly thank you all!

Ana Catarina Morais

*“When life gets you down, you know what you gotta do?
Just keep swimming.”*

*"Dori", *Finding Nemo**

Contents

1	Introduction	1
1.1	Problem Overview	1
1.2	Motivation	1
1.3	Structure of the Document	2
2	Literature Review	3
2.1	Vehicle Routing Problem	3
2.2	Dial-a-Ride Characteristics	4
2.2.1	DARP Formulation	5
2.2.2	DARP Common Features	5
2.2.3	DARP Objective Function	6
2.3	Solution Approaches	8
2.3.1	Exact Approaches	8
2.3.2	Heuristics and Meta-Heuristics Approaches	9
2.3.3	Hybrid Algorithms	13
2.3.4	Other Approaches	14
2.3.5	Benchmark Problems	14
3	Single Customer Dial-a-Ride Problem with Heterogeneous Fleet and Drivers	15
3.1	Differences Between the Proposed Model and a Classical DARP	15
3.2	Assumptions	15
3.3	Variables	16
3.3.1	Sets/indexes	16
3.3.2	Requests' Parameters	17
3.3.3	Resource parameters	17
3.4	Objective Function	18
3.5	Constraints	19
4	A Tabu Search approach for the multi-objective DARP	21
4.1	Tabu Search	21
4.1.1	Initial Solution	23
4.1.2	Neighborhood Definition	23
4.1.3	Aspiration Criterion	26
4.1.4	Termination Criteria	26
4.1.5	Diversification, Intensification and Tabu Tenure	26
4.1.6	Dealing with a Multi-Objective Function	27
4.2	Software Application	28
4.2.1	Python Language	28

4.2.2	Visual Studio Code	29
4.2.3	Algorithm Structure in Python	29
5	YellowFish Case Study	31
5.1	YellowFish data characteristics	31
5.2	Current Allocation Approach	31
5.3	Computational Experiments	33
5.3.1	Dataset	33
5.3.2	Analysis of the Current Manual Solution	34
5.3.3	Algorithm Results	36
6	Conclusions and Future Work	45
A	Appendix 1	47
B	Appendix 2	49
	References	51

List of Figures

2.1	Pickup and Delivery problem and possible solution example (Wang, 2014). . . .	4
2.2	Examples of a directed and indirect graph (Ruohonen, 2013)	6
2.3	Pareto Frontier for a minimization problem with two objectives (Caramia and Dell’Olmo, 2008)	8
2.4	Cluster-First Route-Second process example (Prins et al., 2014).	10
2.5	1-Point Crossover example for a binary coding (Umbarkar and Sheth, 2015) . . .	11
2.6	Simulated Annealing Behavior during the iteration process (Caparrini, 2017) . .	11
4.1	Flow chart of the Tabu Search Algorithm	22
4.3	Relocate Neighborhood	24
4.4	Exchange Neighborhood	25
4.5	Delete Neighborhood	25
4.6	Genetic Algorithm Type Neighborhood	26
4.7	Tabu Search Script	29
5.1	(A) Full-Timer (B) Full-Timer + Part-Timer	32
5.2	Current Allocation System	32
5.3	Graphics related with the settled objectives and vehicles used for YF allocation .	35
5.4	Graphics related with the settled objectives for YF allocation (continuation) . . .	36
5.5	Pareto charts for three different days	40
5.6	Objectives Functions Results comparison	41
5.7	Computational Time results	43
B.1	Non-dominated set for a run of the third scenario for 22/10/17	49

List of Tables

2.1	DARP Classification	5
4.1	Algorithm modules and scripts	29
5.1	Numbers of vehicles per capacity	31
5.2	Request <i>per</i> Year	33
5.3	Requests per day in the sample. All days are from 2017.	33
5.4	Objective Function Values for YellowFish allocation method	34
5.5	Average number of non-dominated solutions when optimizing f_4 with waiting time constraints	37
5.6	07/12 (29 Requests): Best Objective Function Values for the Algorithm	38
5.7	07/11 (60 Requests): Best Objective Function Values for the Algorithm	38
5.8	29/10 (105 Requests) -Best Objective Function Values for the Algorithm	38
5.9	22/10 (212 Requests): Best Objective Function Values for the Algorithm	38
5.10	01/09 (419 Requests): Best Objective Function Values for the Algorithm	39
5.11	Computational times comparison between two laptops	44
A.1	Non-dominates set size for the three scenarios	47

Abbreviations

ADT	Abstract Data Type
B&C	Branch-and-cut
B&P	Branch-and-price
B&P&C	Branch-and-price-and-cut
CVP	Chinese Postman Problem
DA	Deterministic Annealing
DARP	Dial-a-Ride Problem
GA	Genetic Algorithm
GA Type	Genetic Algorithm Type local search operator
IDE	Integrated Development Environments
LNS	Large Neighborhood Search
NP-hard	Non-deterministic polynomial-time hard
PDP	Pickup and Delivery Problem
RPP	Rural Postman Problem
SA	Simulated Annealing
TS	Tabu Search
TSP	Traveling Salesman Problem
VSN	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VSCode	Visual Studio Code
VS	Visual Studio
YF	YellowFish Transfers

Chapter 1

Introduction

The present work is performed in the context of the dissertation project of the Integrated Master in Industrial Engineering and Management of the Faculty of Engineering of Oporto University.

1.1 Problem Overview

The problem under research arises in YellowFish Transfers company. YellowFish is an airport private transfers company in Algarve, Portugal and currently allocates its requests manually in a daily basis. This allocation process implies, regarding the booking set for a specific day and the available drivers, the allocation of every request to one vehicle. Although the process tries to be efficient reducing the operational costs, with days that can reach more than 500 requests, the efficiency and the total time of the task can be compromised once it is a completely manual procedure. In order to improve the current solution it is firstly needed to model the problem. In this particular case, the problem is modeled as a single customer Dial-a-Ride Problem (DARP). A DARP is a vehicle routing problem involving humans as the load to be transported. Once this framework has a real application in the tourism sector and a company has always several concerns regarding its services, the model tackles not one objective but three. Regarding the solution approach, among the several methods in academia, a Tabu Search meta-heuristic is applied. Besides addressing a multi-objective DARP the present Tabu Search samples the neighborhood and uses five different local search operators (not operating at the same time). The algorithm is developed and implemented in Python 3.6.5 language with Visual Studio Code as IDE.

1.2 Motivation

Having a project that gathers a multiple objectives, seasonality (which allows the analyses of the algorithm performance within a wide range of instances size), a Tabu search and being "single customer" it is a unique opportunity to create value for academia and for the real world, saving money for the company and potentially for its clients but also saving green-house gas emissions due to the less kilometers spent by the vehicles.

Besides the decrease in the time that is spent on the allocation task, once the problem is a multi-objective, its implementation can not only creates value in one domain but in three (operations costs in the core service, employees and customers) resulting in a three times more satisfied company. Once it has a practical application, this project is helping to optimize a process that is now done 100% manually, helping the company to reduce the time that spends on the allocation task and hopefully the operation costs. This fact is also important because nowadays the company closes the booking process 48 hours before the operation day and the proposed method can contribute to reduce this time potentially to 24 hours.

1.3 Structure of the Document

Besides the chapter that it is being read at this precise moment, there are more five chapters. The Chapter 2, Literature Review, does a contextualization of the problem and the algorithm in academia, showing the common features of the formulation as well as the methods used to achieve to feasible solutions for the model and their pros and cons. One section ahead the reader, finds Chapter 3, Single Customer Dial-a-Ride Problem with Heterogeneous Fleet and Drivers, that mathematically formulates the problem. This chapter presents the variables, parameters, constraints, objective functions and the way they relate with each other. In Chapter 4, Algorithm, the construction and features of the Tabu Search algorithm are explained, detailing all the methods used in the present work. The Chapter 5, YellowFish Case Study, connects the theory with the real world application. First a description of the company along with their current model of allocating the vehicles to the requests is made. Furthermore, the results of the computational experiments with the dataset provided by the company are shown and analyzed. Finally Chapter 6 summarizes the work realized and highlights the most important findings and results. Some prospects and future work are also a matter of attention in these last pages of dissertation.

Chapter 2

Literature Review

In this chapter all the background necessary to understand the following chapters is given. Therefore, concepts related with the Vehicle Routing Problem (VRP), the Dial-a-Ride Problem (DARP), graph theory, and solving methods for DARP (exact methods, heuristics, meta-heuristics and some other less studied approaches) are exposed.

2.1 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) was firstly discussed in 1959 when Dantzig and Ramser, with the objective of solving the problem of distributing gasoline to service stations set the mathematical programming formulation and algorithmic approach. VRP is one classical type of a routing problem. Besides VRP, some of the most well known problems, related with the present work, are the Traveling Salesman Problem (TSP), the Rural Postman Problem (RPP) and the Chinese Postman Problem (CVP) (Laporte and Osman, 1995).

The classical VRP aims to find a set of routes at a minimal cost (finding the shortest paths, minimizing the number of vehicles, etc.), beginning and ending the route at the depot, so that the known demand of all nodes is fulfilled. Each node is visited only once, by only one vehicle, and each vehicle has a limited capacity (Tonci Caric, 2008).

The Pickup and Delivery Problem (PDP), see Figure 2.1, is an extension of the VRP, which considers the transportation of goods or persons between pickup and delivery locations without transshipment at intermediate location (Wang, 2014). As shown in the Figure 2.1 part (A), a PDP is characterized by at least one depot (place from where all the vehicles start and end a route) and several requests with a pick-up (+) and drop off (-) locations. A solution for a PDP is a set of routes in order to satisfy all the request. Note that the pickup and drop-off of a specific request has to be in the same route (Figure 2.1 part (b)). Nevertheless, it is important to keep in mind that for a same problem different goals can imply completely different sets of routes as a solution. In the case where the load are people, the problem is nominated as Dial-a-Ride Problem (DARP), and instead of load, terms as customers or clients are commonly used (Savelsbergh and Sol, 1995).

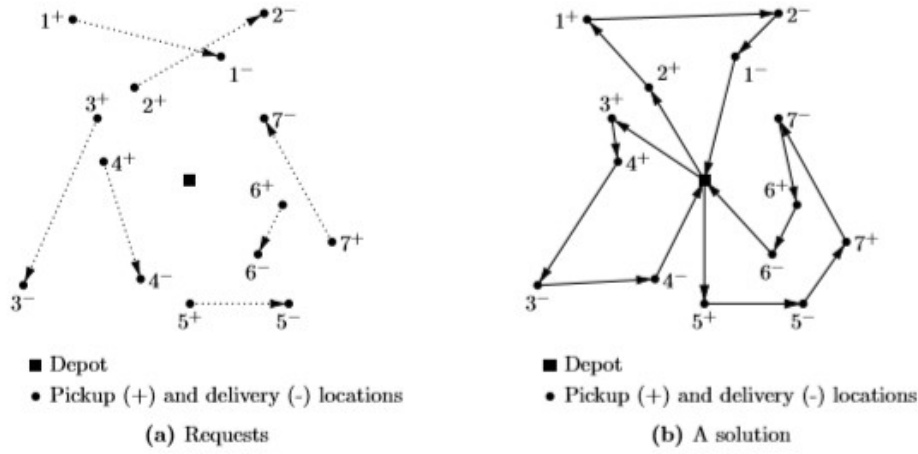


Figure 2.1: Pickup and Delivery problem and possible solution example (Wang, 2014).

According to Laporte and Osman (1995), "*Vehicle routing is truly one of the great success stories of operations research*", being the Dial-a-Ride an extension of a VRP it can be also included in the statement. In fact, since its first relevant publications in the 80's, focusing on the single vehicle problem (Chassaing et al., 2016), research into the DARP has been growing steadily (Ho et al., 2018).

2.2 Dial-a-Ride Characteristics

According to Colomi and Righini (2001), the name 'dial-a-ride' comes from the phone call by which a customer is supposed to ask for service. A DARP as well as the others problems referred in the previous section (PDP, VRP, TSP, RPP, CVP) is a NP-hard problem (non-deterministic polynomial-time hard) meaning that the number of configurations increases exponentially to the problem size (Urban, 2006). Therefore, exact solving methods are only suitable for small sized situations. The most common application of DARP arises in door-to-door transportation services for elderly or disabled people (Cordeau and Laporte, 2007) as well as other health care areas. Works like the ones presented by Desrosiers et al. (1991), Toth and Vigo (1997), Colomi and Righini (2001), Cordeau and Laporte (2003), Diana and Dessouky (2004), Melachrinoudis et al. (2007) and Paquette et al. (2013) are examples of those kind of application. Nevertheless, in the past few years, applications in public transportation (Marković et al. (2015), Parragh et al. (2015)) and in airport transportation (Reinhardt et al. (2013)) areas are also being studied and applied.

Besides the different applications, this type of combinatorial problem can be classified according with two different aspects:

- Timing of the system decisions;
- Certainty of the request information.

Regarding the first aspect, it is possible to have *static* or *dynamic* systems. In a *static* problem, all the information is known and all the decisions are settled before the start of the operations. On the other hand, in a *dynamic* system, modifications and new information can appear after the beginning of the operations. In (Ho et al., 2018) three common triggers of change are outlined:

- Appearance of new users;
- Updated information about the existing users;
- Unexpected disturbances (delays, vehicle breakdowns, etc.).

Despite of some early studies about dynamic DARP (Wilson et al. (1976), Psaraftis (1988)), according to Parragh et al. (2012), Ho et al. (2018), Berbeglia et al. (2012) and Cordeau and Laporte (2003) the majority of the work is concentrated in the static version of the problem.

As for the certainty of the request information (second aspect of classification), a DARP can be considered as *deterministic* or *stochastic*. When the information used to the decision process is certain, the DARP is considered *deterministic*. For problems where decisions are made upon imperfect information (based on forecasts with the associated uncertainties) the problem is classified as *stochastic*. A summary of the DARP classification is presented in Table 5.2

Table 2.1: DARP Classification

		Certain Informations?	
		No	Yes
Changes after the beginning of the operations?	No	Static and stochastic	Static and deterministic
	Yes	Dynamic and stochastic	Dynamic and deterministic

2.2.1 DARP Formulation

There are two main reference DARP formulations among literature. One is based in three-index decision variables (Cordeau, 2006) and other is based in two-index decision variables (Ropke et al., 2007). For that reason, they are known as three-index or two index formulations. Regardless the formulation type they are based in graph theory, more precisely in direct graphs. As it is possible to see in the Figure 2.2a graph is a pair of sets (V, E) , where V is the set of vertices and E is the set of edges, formed by pairs of vertices. E is a multiset or, in other words, its elements can appear more than once so that every element has a multiplicity (Ruohonen, 2013). The graph is nominated as direct in the case that the elements of E are ordered pairs (Figure 2.2b), directions are relevant (the arc from a vertex u to vertex v is written as (u, v) and the other pair (v, u) is the opposite direction arc (Ruohonen, 2013)).

2.2.2 DARP Common Features

Due to the application oriented character of dial-a-ride problems, a large variety of objective functions and constraints may be observed in the literature (Braekers et al., 2014). One of the simplest

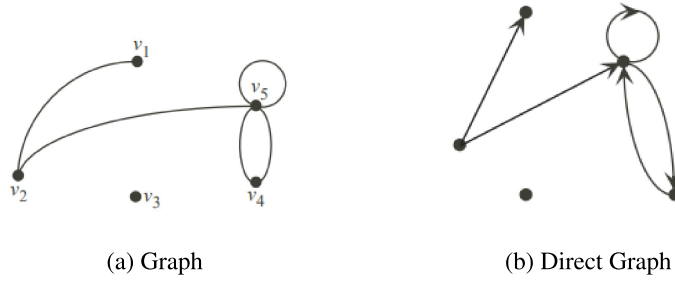


Figure 2.2: Examples of a directed and indirect graph (Ruohonen, 2013)

cases of the DARP is when all users are served by a single vehicle (Cordeau and Laporte, 2007). To this simple model several features can be added and/or modified such as:

- **Rejection:** To reject a client request may be or not allowed. While in the majority of the cases, probably because they are applied to health care services, rejections are not allowed, there are opposite examples like the one presented by Parragh et al. (2015). In this case, Parragh et al. (2015) try to maximize the total profit of a demand-responsive transportation company operating during the nights in Oporto, leading to the necessity of evaluating the profitability of a request and allowing its rejection when not profitable.
- **Time Window:** In some problems, users specify the earliest/latest times that they can be picked-up/dropped-off (e.g., Chassaing et al. (2016));
- **Depot:** A DARP can be single (e.g., Berbeglia et al. (2012)) or multi-depot (e.g., Melachrinoudis and Min (2011)). A DARP is single depot if all the vehicles/routes start and end at the same place, and is multi-depot if there is more than one location for a vehicle to start and end its route;
- **Vehicle Capacity:** A fleet is homogeneous if all the vehicles have the same capacity (e.g., Rahmani et al. (2016)). On the other hand, when a fleet has vehicles with different capacities is named heterogeneous fleet (e.g., Aldaihani and Dessouky (2003)).

2.2.3 DARP Objective Function

As refereed in the previous sub-section, another crucial and wide aspect of these kind of problems is the objective function. Depending on the objective function the solution for a problem can be completely different. In academia, trying to minimize operations costs or customer satisfaction are frequent tackled as objectives. Functions expressing fleet dimensioning, total traveled distance, drivers working time or total transportation time are frequent objectives for papers addressing operation costs. On the other hand, for client satisfaction assessment, frequent goals pass through minimizing the total riding time, waiting time or time windows deviations. Garaix et al. (2011), Parragh et al. (2015) and Lim et al. (2017) propose some less studied goals such clients occupancy rate, company profit and staff workload.

Furthermore, some works, instead of addressing one single goal try to find a solution that is a good compromise between several goals. Problems with such characteristics are known as multi-objective. To solve them, usually methodologies involve either *a posteriori* information (after modeling and optimizing the problem a decision maker defines preferences based on optimization), *a priori* information (preferences are settled before optimization) or both. In Caramia and Dell’Olmo (2008) the reader can find several methodologies for addressing multi-objective problems. Nevertheless, Ho et al. (2018) outline three ways of dealing with the multi-objective problems among DARP literature.

1. Weighted sum of the goals. Articles such as the ones presented by Sexton and Bodin (1985), Diana and Dessouky (2004), Jorgensen et al. (2007) and Melachrinoudis et al. (2007) are examples of such approach. For problems where the relative importance of each objective is unknown or unquantifiable (Ho et al., 2018), the application of this approach can be compromised once the results are highly dependent on the weights values (*a priori* decision);
2. Order the objectives depending on its importance. One objective is clearly dominant to the others being the one to be optimized. An example of this application is presented by Schilde et al. (2011);
3. Find the Pareto frontier of the problem. A Pareto Frontier is a set of non-dominated solutions. A non-dominated solution is a solution where there is no way of improving any objective without degrading at least one other objective (Figure 2.3) (Deb et al., 2002). In this case, goals are often conflicting against each other meaning that it is not feasible to satisfy all the goals at a time. Once the output is a set of solutions, in the end of the process it requires a human decision about the solution to be implemented (*a posteriori* decision). In papers such as Paquette et al. (2013), Jaeggi et al. (2008) the reader can find this kind of approach to Multi-Objective DARPs.

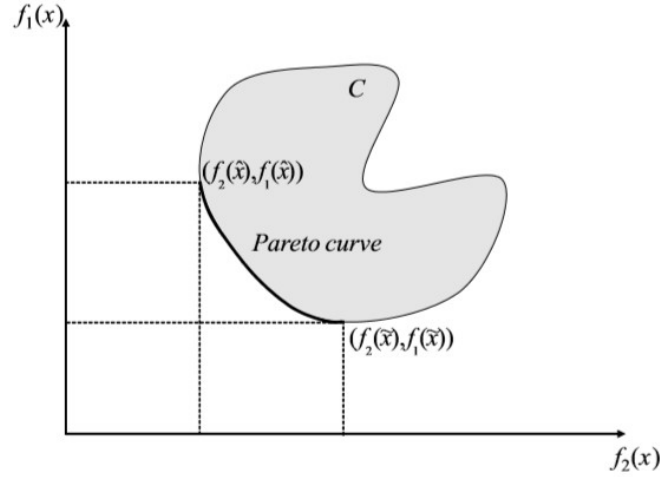


Figure 2.3: Pareto Frontier for a minimization problem with two objectives (Caramia and Dell’Olmo, 2008)

2.3 Solution Approaches

As well as the many variations in DARP formulation, there are also a wide range of methods used to find a solution. It is possible to divide the solutions approaches in two major categories: the exact methods, and the heuristics and meta-heuristics.

2.3.1 Exact Approaches

The majority of the papers that can be found in the exact approach category are based on branch-and-bound (B&B) concepts and have a relatively small size. An instance size of 96 requests in a day in Braekers et al. (2014) and Ropke et al. (2007) is the maximum size found for this kind of applications. The three main methods in the exact approach category are:

- **Branch-and-cut (B&C):** Branch-and-cut algorithms combine a branch-and-bound search procedure with the concept of cutting planes Braekers et al. (2014). To do this, the problem has to suffer a LP-Relaxation (Linear Programming Relaxation) first. Articles such as Braekers et al. (2014) and Cordeau (2006) apply B&C too. Among all the exact methods B&C is the one with more works published (Ho et al., 2018);
- **Branch-and-Price (B&P):** In Parragh et al. (2015) and Garaix et al. (2011) a B&P algorithm is applied to solve a DARP problem. Authors such as Molenbruch et al. (2017) refer this type of approach as Column Generation once there is a focus on column generation rather than generating cuts for LP relaxations in a B&B procedure. B&P algorithms require the reformulation of the problem into a restricted master problem and a pricing subproblem (Ho et al., 2018).

- Branch-and-Price-and-Cut (B&P&C): As the name suggests this approach merge concepts of the last two. In such cases column generation is integrated into a branch-and-cut algorithm, based on the observation that most variables in the solution are nonbasic ($variables = 0$) (Molenbruch et al., 2017). More information and results of this application can be found in works such as Qu and Bard (2015) and Gschwind and Irnich (2014).

2.3.2 Heuristics and Meta-Heuristics Approaches

Once a DARP is a NP-hard problem, for large instances, optimal solutions are not expected to be found in polynomial time Molenbruch et al. (2017). For this reason, approximation methods/heuristics are recommended for large instances. Although the application of heuristics does not ensure that the optimal solution is found, the heuristic algorithms obtain very good results compared with the hand-made schedules, both in terms of service quality (all the service requirements are met) and overall cost (Toth and Vigo, 1997). Heuristics are problem-dependent meaning that they can be quite specific. Some classical heuristics for DARP, as referred by Molenbruch et al. (2017), have the inability to escape from local optima (which has led to a declination of the numbers of works published during the past decade). Therefore, in order to prevent local optima, improvement procedures are normally embedded in a heuristic such as Tabu Search, Simulated Annealing, or Genetic Algorithm (Melachrinoudis et al., 2007). These algorithms, Tabu Search, Simulated Annealing and Genetic Algorithms, referred by Melachrinoudis et al. (2007) are meta-heuristics. The meta-heuristic concept was introduced for the first time by Glover when, in 1986, the first academic work about a Tabu Search algorithm was published. At that time he stated "*Tabu search may be viewed as a meta-heuristic superimposed on another heuristic*". Meta-heuristic optimization techniques are problem-independent and have proved effective in solving complex, real-world optimization problems with many local minima, problems to which traditional, gradient-based methods are ill suited (Jaeggi et al., 2008).

2.3.2.1 Classical Heuristics

Within the classical heuristics there are construction insertion heuristics and cluster-first route-second heuristic. Although meta-heuristics are more effective than construction insertion heuristics, typically they get within roughly 10-15% of optimal in relatively little time (Johnson and McGeoch, 1997), meaning that they can be useful when there is a need of quickly finding feasible solutions (Ho et al., 2018). Because of the last characteristic, there are some applications such as the ones proposed by Braekers et al. (2014) and Aldaihani and Dessouky (2003) which use a constructive heuristic as an initial solution for a meta-heuristic. These heuristics are constructive because the solution is built step-by-step using a set of defined rules. The first example like the one described before was the greedy insertion heuristic, proposed by Jaw et al. (1986) which sorts requests according to the pickup time. In each iteration, in the first route for which a feasible insertion is found, the first-sorted user is inserted at the best feasible position. It is also relevant to highlight the work proposed by Diana and Dessouky (2004) which has proposed a parallel insertion

heuristic. They have reported results on instances of sizes 500 and 1000 (Cordeau and Laporte, 2007) which is, in the largest instances, more than 10 times bigger than the maximum instances registered for an exact method approach. More applications of parallel insertion heuristics can be found in the works of Toth and Vigo (1997) (they were actually the first ones to use it), Luo and Schonfeld (2007) and Calvo and Colomi (2007).

Cluster-first route-second heuristic was proposed by Ioachim et al. (1995) and, as the name implies, solves the problem in two separate phases. The first phase divides the requests in clusters and then for each cluster composes routes (see Figure 2.4). Works such as Jorgensen et al. (2007) join the concept of cluster-first route-second concept with the Genetic Algorithm concept. As it is going to be seen in section 2.3.3., works where heuristics are mixed seem to be increasing.

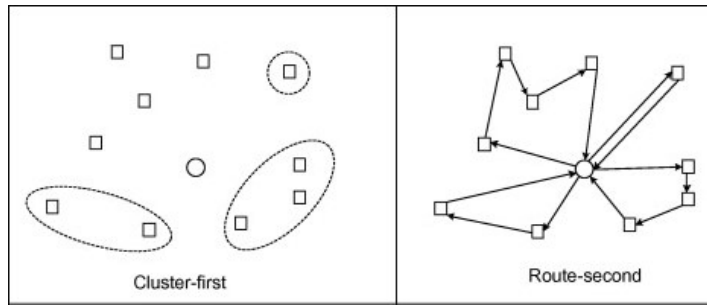


Figure 2.4: Cluster-First Route-Second process example (Prins et al., 2014).

2.3.2.2 Genetic Algorithm

Genetic algorithms (GA) are part of a broader class, the evolutionary algorithms. The term is inspired by Darwin's evolution theory. Unlike other meta-heuristics, that are local search based, GA are population based. From a population (set of solutions) parents are chosen (as it happens in nature "stronger"/fittest individuals have a higher probability of being chosen). Offsprings (new individuals) are created by applying crossover and mutation operators on the parents. Then, some of the existing individuals may be replaced by the new offsprings (Ho et al., 2018). Among the literature there several crossover types (see Umbarkar and Sheth (2015) for more information). Figure 2.5 shows the simplest crossover type, the 1-point crossover. According with Jorgensen et al. (2007), GAs have shown good performance on a number of related routing problem. Nevertheless, the same author states that, regarding the chromosome representation, the extension to the DARP is problematic being the main obstacles the precedence constraints. Note that in the chromosome representation, both the allocation of customers to vehicles and the order of the customers on the routes are encoded (Jorgensen et al., 2007). In fact, among literature, different authors use different coding schemes.

Parent 1:	1 0 1 0 1 0 0 1 0
Parent 2:	1 0 1 1 1 0 1 1 0
Offspring 1:	1 0 1 0 1 0 1 1 0
Offspring 2:	1 0 1 1 1 0 0 1 0

Figure 2.5: 1-Point Crossover example for a binary coding (Umbarkar and Sheth, 2015)

2.3.2.3 Simulated Annealing

Simulated Annealing (SA) is a stochastic local search meta-heuristic inspired by the physical annealing process. "Annealing is referred to as tempering certain alloys of metal, glass, or crystal by heating above its melting point, holding its temperature, and then cooling it very slowly until it solidifies into a perfect crystalline structure" (Du and Swamy, 2016). According with Molenbruch et al. (2017) SA algorithms are usually combined with characteristics from other meta-heuristic frameworks. In terms of application, a SA is a descent algorithm modified by random/stochastic ascent moves in order to escape local minima which are not global minima (Du and Swamy, 2016), meaning that it is possible to accept a worsening solution in a certain iteration. A descent algorithm involves using a feasible solution and making small changes to the solution. The solutions originated from that move will create a neighborhood. Only solutions that improve the objective function are accepted in a descent algorithm.

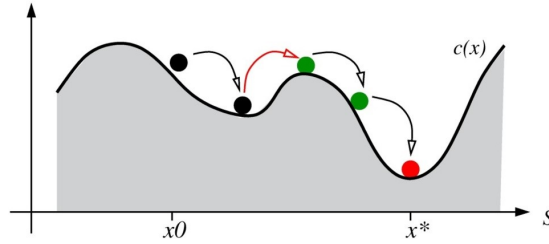


Figure 2.6: Simulated Annealing Behavior during the iteration process (Caparrini, 2017)

In 2014, Braekers et al. suggested a variation of SA known as deterministic annealing (DA). Ho et al. (2018) considers this variation as "*highly effective and efficient*". In fact, for common benchmark instances (with a single-depot and a multi-depot) the framework achieved the best known results. In this variation, besides more complex local search operators, this framework accepts deteriorations which are smaller than a gradually lowered threshold (Molenbruch et al., 2017).

2.3.2.4 Variable Neighborhood Search

Variable Neighborhood Search (VNS) is a meta-heuristic proposed by (Mladenovic and Hansen, 1997). VNS exploits the insight that neighborhoods are defined with respect to a particular operator Molenbruch et al. (2017), meaning that the neighborhood operators change on a systematic

basis allowing to escape from local optima solution. This framework had its first application in DARP in Parragh et al. (2009) and since then authors such as Molenbruch et al. (2017), Muelas et al. (2015) and Schilde et al. (2014) have also applied this framework. Nevertheless, as well as different objective functions also different operators were used, for example in Muelas et al. (2013). Other cases such as Schilde et al. (2011) and Schilde et al. (2014) apply the framework combined with a simulation method.

2.3.2.5 Large Neighborhood Search

Large Neighborhood Search (LNS) method destroys part of the current solution at each iteration. This destruction is achieved by removing a certain percentage of requests from the solution. After, those removed requests are re-inserted in a better position. By doing this, this type of approach implies larger changes than those by the typical neighborhood operators employed in other meta-heuristics Ho et al. (2018). The removal and the insertion are made by one of the removal/insertion heuristics. Molenbruch et al. (2017) highlights random removal, worst removal, sequential removal, route removal and related removal as common "destroy" operators and random insertion, greedy insertion, k -regret insertion, most-constrained-first insertion and space-time-related insertion as common "repair" operators. Ropke and Pisinger (2006), Masmoudi et al. (2016) and Gschwind et al. (2016) apply LNS approach in different domains.

2.3.2.6 Tabu Search Algorithm

Since it was proposed by Glover in 1986, Tabu Search (TS) has been applied multiple times in routing and Dial-a-Ride problems (Ho et al., 2018). One of the reasons may be related with the statement of Cordeau and Laporte (2007) saying that a tabu search "*can easily accommodate a large variety of constraints and objectives, even if these are nonlinear*".

TS explores the solution space by moving at each iteration from the current solution to the best solution in a subset of its neighborhood (Melachrinoudis et al., 2007). As it happens in SA, the new solution may deteriorate from one iteration to the next. Due to declaring as tabu (forbidden) some attributes of recently explored solutions, local optimal and cycling are avoided. The number of iterations that a certain move is forbidden is an attribute called tabu tenure. The tabu status can be overridden when a tabu solution is better than the current best solution found so far (Melachrinoudis et al., 2007), among the literature this is called aspiration criterion. One of the first works applying a TS to DARP system was Cordeau and Laporte (2003) which was used as a main reference for many further more complex DARP systems as the ones proposed by Paquette et al. (2013), Attanasio et al. (2004), Melachrinoudis et al. (2007), Melachrinoudis and Min (2011), Kirchler and Wolfler Calvo (2013), Escobar et al. (2014) and Jaeggi et al. (2008). In Cordeau and Laporte (2003) a relaxation mechanism is applied allowing infeasible solutions. According to Cordeau and Laporte "*the complex modification of a feasible solution into another feasible solution can then be achieved by a series of simpler modification through intermediate infeasible solutions*". Besides the relaxation method, diversification and intensification methods

are applied. A diversification method intends to drive the search towards less explored regions of the solution space whenever a local optimum is reached Cordeau and Laporte (2003). On the other hand, an intensification method intends to explore more thoroughly the portions of the search space that seem promising (Gendreau and Potvin, 2010). In Gendreau et al. (2015) is also highlighted that not always such intensification is needed because the search in the "normal" running is thorough enough.

Kirchler and Wolfler Calvo (2013) and Escobar et al. (2014) presented a granular Tabu Search which is a Tabu Search with a particular focus on the local search phase. It uses a reduced neighborhood (granular neighborhood) which ideally should not include moves which are unlikely to belong to good solutions Kirchler and Wolfler Calvo (2013). To regulate the size of the neighborhood (Kirchler and Wolfler Calvo, 2013) uses a threshold. This pretends to address the time that is consumed in the evaluation of the neighborhood, which according with Ho et al. (2018) is one of the most time-consuming tasks in a Tabu Search. An alternative to solve the mentioned problem is using a random sampling like Detti et al. (2017) does.

In works like the ones from Glover (1990), Glover (1986), and Du and Pardalos (2013) the reader can find more about TS and get some application tips in Gendreau and Potvin (2010). In Chapter 4, the TS algorithm features are more explored.

2.3.3 Hybrid Algorithms

This kind of approach seems to be a growing trend in the last years. According to Ho et al. (2018) in the last ten years, 17 works were published using an hybrid method. Hybrid methods involve the application of more than one approach, could the method be a meta-heuristic, a mathematical programming or a constraint programming approaches. When there is a combination between two meta-heuristics usually either each meta-heuristic is executed sequentially or one is executed within another (Ho et al., 2018). It is common to find solution that combine a population based meta-heuristic, mostly GA, with a single-solution based solution like local search, TS, VNS and SA. This happens due to the population-based meta-heuristic's ability for exploration and the single-solution based meta-heuristic's ability for exploitation (Ho et al., 2018). Chassaing et al. (2016), Molenbruch et al. (2017), Khelifi et al. (2013) presented works where this kind of approach can be found.

Concerning hybrid methods involving mathematical programming approaches, a popular way to combine a meta-heuristic with a mathematical based approach is to embed a meta-heuristic into a mathematical method or vice versa (Ho et al., 2018). Therefore, these kind of approaches can be also named as *mathheuristics*. While Parragh et al. (2012) and Parragh et al. (2015) use a two hybrid column-generation method, works such as Ritzinger et al. (2016) and Gschwind et al. (2016) combine meta-heuristics with Dynamic Programming (DP). Used for the first time in a DARP by Psaraftis (1980), DP is a well known exact method that solves complex problems by decomposing the problem into smaller subproblems (Ritzinger et al., 2016).

Finally, Berbeglia et al. (2012) presented a framework involving constraint programming and a TS heuristic. According to Berbeglia et al. (2012) constraint programming is a programming

paradigm based on reasoning and search techniques. To more information about this technique check Van Hentenryck (1989).

2.3.4 Other Approaches

It is possible to use some other approaches, less studied in literature, as for example, simulation methods and approximation methods.

Simulation methods can be quite helpful to the understanding of DARP systems. They mostly analyze the impact of parameter changes concerning the system's design and the way of operation (Molenbruch et al., 2017). Therefore, many of its application involve stochastic systems as it happens in Defflorio et al. (2002) or dynamic systems as it happens in Gomes et al. (2014).

Some works like Maalouf et al. (2014) apply an approximation methods on a dynamic DARP while Gupta et al. (2010) uses a α -approximation algorithm to a k -forest problem to solve a static DARP.

2.3.5 Benchmark Problems

Many DARP systems among the literature have real-life application. Nevertheless, with the goal of compare the efficiency of the different algorithms that can be found in literature, two sets of artificial benchmark data have been proposed to perform computational tests (Molenbruch et al., 2017). Both sets are divided in a instances (small vehicle capacities) and b instances (large vehicle capacities).

- The first set was introduced by Cordeau (2006) and had a later extension made by Ropke et al. (2007) and consists of 42 instances, including 16–96 requests (Molenbruch et al., 2017).
- The second set was present by Cordeau and Laporte (2003) containing between 24 and 144 requests (Molenbruch et al., 2017).

In both cases authors have extended the sets to richer problem variants. Authors such as Berbeglia et al. (2012) and Braekers et al. (2014) apply variations for the first benchmark set while works such as Li et al. (2016) apply them on the second one.

See Ho et al. (2018) for a further analysis about papers released between 2007 and 2018 and Cordeau and Laporte (2007) for informations about papers realized until 2007. Also in Molenbruch et al. (2017) it is possible to have an overview about the most relevant papers since 1980 till 2017.

Chapter 3

Single Customer Dial-a-Ride Problem with Heterogeneous Fleet and Drivers

This chapter formulates the mathematical problem of a static and deterministic single customer dial-a-ride problem with pickup and drop-off times, heterogeneous fleet, multiple drivers and multiple depots.

3.1 Differences Between the Proposed Model and a Classical DARP

Despite of having similarities with previous models in literature (multi-depot, heterogeneous fleet), the current problem presents singularities worth of studying. This model does not allow sharing rides in opposition to the classical DARP model, being therefore stated as "single customer". This may lead the reader to think that it works like a taxi, but this problem also differs from a classical taxi situation in the following aspects. All the requests are known in advance and require a previous reservation. A request includes several aspects as known pickup and drop-off places, number of passengers and time of pickup and drop-off. These aspects are crucial for the following formulation. Also there is a set of predefined locations, so the customer is picked-up or dropped in a particular location within the available set.

3.2 Assumptions

In real transportation networks, riding time can vary from day to day (for instance between working days and weekends) and also during the same day (for instance, travel times are higher in rush hours than at night or in the central hours of the day or different drivers drive through different paths and at different velocities) (Colomi and Righini, 2001). Nevertheless, once there is not data allowing the prediction of such uncertain factors the present formulation assumes a fixed traveling time, meaning that both distance and traveling speed are constants. Other assumption is that the unloading/loading time at each vertex is considered negligible. Finally customer cancellations and delays are not considered in the current formulation. Similarly, operating problem (e.g. vehicle

breaks, a driver illness) are not contemplated in the formulation. All these assumptions make this problem a deterministic one. To compensate the fact of considering deterministic although it is not in its real application a slack time in the traveling time is added. In addition, once all the requests are known *a priori*, not being possible new entries or cancellations after the beginning of the operations, this is a *static* dial-a-ride problem.

3.3 Variables

In this section sets, parameters, auxiliary and decision variables are defined in order to be possible to formulate the problem regarding the features observed.

3.3.1 Sets/indexes

A Dial-a-Ride problem can be defined as a graph $G(V, E)$, where:

- $R = \{1, 2, \dots, n\}$ is the set of requests;
- $P^+ = \{1, 2, \dots, n\}$ is the set of pick-up points;
- $P^- = \{n + 1, \dots, 2n\}$ is the set of drop-off points;
- $D = \{2n + 1, \dots, 2n + m\}$ is the set of depot points;
- $V = P^+ \cup P^- \cup D$ is the set of all nodes of the graph; A request i is a pair of a pick-up (i) and a drop-off ($n + i$) points, so if the physical drop-off or pickup locations of two requests are the same they will count as separate nodes in the network;
- E is the set of arcs of the graph;
- $K = \{1, 2, \dots, k\}$ is the set of vehicles;
- J is the set of customer types. Customers can be divided regarding their age (babies, children and adults). A customer is considered a baby when the age ≤ 3 , a children when his age is between 3 and 13 and an adult when its age > 13 . This division is relevant in terms of logistical needs since babies and children need safety seats.
- $M = \{1, 2, \dots, m\}$ is the set of drivers;
- L is the set of drivers' categories. A driver may work as a full-timer or as a part-timer. This will have impact not only on the amount of work hours but also on the driver's salary. Drivers of same category should have similar salaries which means that the difference of salaries within a category salaries be minimized;
- A_k is the list of requests scheduled for vehicle k , with $k \in K$;

3.3.2 Requests' Parameters

Given a request $i \in R$, let:

- T_i is the type of request;

$$T_i = \begin{cases} 0, & \text{if } i \text{ is a pickup} \\ 1, & \text{if } i \text{ is a drop-off} \end{cases}$$

Similarly to other formulations, it is possible to divide customers according to their specifications. Jorgensen et al. (2007) divided them as inbound and outbound. For the inbound requests ($T_i = 1$) the relevant issue is to arrive on time at the drop-off location, for an outbound customer ($T_i = 0$) the pick-up time is specified.

- $q_{i,j}$ is the load (number of passengers) of the request i by category j , with $j \in J$;
- p_i^+ : is the pick-up point, with $p_i^+ \in P^+$;
- p_i^- : is the drop-off point, with $p_i^- \in P^-$;
- e_i is the pick-up time;
- d_i is the drop-off time; A customer either settles a pick-up or a drop-off time, being the other one defined according to a distance/time matrix.
- rev_i is the driver's revenue on a request i ;

3.3.3 Resource parameters

- $t_{g,h}$ is the traveling time between g and h , with $g, h \in V$ and $t_{g,h} \in \mathbb{R}^+$;
- $\delta_{g,h}$ is the distance between g and h , with $g, h \in V$ and $\delta_{g,h} \in \mathbb{R}^+$;
- d_k^+ is the start point of vehicle k , with $k \in K$ and $d_k^+ \in D$;
- d_k^- is the end point of vehicle k , with $k \in K$ and $d_k^- \in D$;
- C_k is the capacity of the vehicle $k \in K$, with $C_k \in \mathbb{N}_0$;
- Max_m is the maximum workload for the driver m , with $m \in M$ and $Max_m \in \mathbb{R}^+$;
- $[st_m, ed_m]$ is the time window of availability of driver $m \in M$, with $st_m \leq ed_m$ and $st_m, ed_m \in \mathbb{R}^+$;
- $\omega_{m,k}$ is defined by

$$\omega_{m,k} = \begin{cases} 1 & \text{if vehicle } k \text{ is allocated to the driver } m, \text{ with } k \in K \text{ and } m \in M \\ 0 & \text{, otherwise} \end{cases}$$

- $\overline{s_m}$ is the average salary of drivers in the same category of driver m , with $m \in M$, $\overline{s_m} \in \mathbb{R}^+$;

Auxiliary variables:

- s_m is the cumulative salary of the driver m , with $m \in M$ and $s_m \in \mathbb{R}^+$;
- Δ_i^k is the number of seats unused of vehicle k while serving the request i , with $i \in R, k \in K$ and $\Delta_i^k \in \mathbb{N}_0$. In the point of view of fuel consumption and also customer satisfaction these values should be as small as possible;

Decision variables:

- v_g^k is the time that vehicle k starts serving the vertex g , with $g \in V, k \in K$ and $v_i \in \mathbb{R}$;
- $x_{g,h}^k$ is given by

$$x_{g,h}^k = \begin{cases} 1, & \text{if vehicle } k \text{ makes the arc between the nodes } g \text{ and } h \\ 0, & \text{otherwise} \end{cases}$$

- $z_{g,h}^k$ is given by

$$z_{g,h}^k = \begin{cases} 1, & \text{if vehicle } k \text{ makes the arc between the nodes } g \text{ and } h \text{ without customers} \\ 0, & \text{otherwise} \end{cases}$$

- $y_{i,j}^k$ is given by

$$y_{i,j}^k = \begin{cases} 1, & \text{if request } i \text{ immediately precedes request } j \text{ for vehicle } k \\ 0, & \text{otherwise} \end{cases}$$

3.4 Objective Function

The problem is formulated as a multi-objective being identified 3 objectives: the minimization of total distance made by the vehicles (f_1), the equality between the wages of the drivers (f_2) and the minimization of the empty seats in a car allocated to a request (f_3). The approach for addressing the multi-objective nature of the problem is found in Chapter 4.

$$\min f_1 = \sum_{k \in K} \sum_{g \in V} \sum_{h \in V} \delta_{g,h} x_{g,h}^k \quad (3.1)$$

$$\min f_2 = \sum_{m \in M} (s_m - \overline{s_m})^2 \quad (3.2)$$

$$\min f_3 = \sum_{i \in R} \sum_{k \in K} \Delta_i^k \quad (3.3)$$

It is relevant to outline that f_1 is the sum of the total kilometers made, with or without customers. Only kilometers made without clients are possible to be minimized, meaning that in terms

of goals, f_1 can also be expressed as f_4 where:

$$\min f_4 = \sum_{k \in K} \sum_{g \in V} \sum_{h \in V} \delta_{g,h} z_{g,h}^k \quad (3.4)$$

3.5 Constraints

The main constraints of the problem are related with requests satisfaction (in terms of time, capacity and precedence), as well as drivers' wages. Constraints (3.5), (3.6) and (3.7) ensure that all requests are satisfied by one vehicle and that the vehicle that picks-up the costumer is the same that drops him. Equations (3.8) and (3.9) ensure that a vehicle starts and ends its daily service at a depot. Regarding the vehicles capacity constraints, those are verified through the constraint (3.10).

$$\sum_{k \in K} x_{p_i^+, p_i^-}^k = 1, \forall i \in R, p_i^+ \in P^+, p_i^- \in P^- \quad (3.5)$$

$$x_{p_i^-, p_j^+}^k = y_{i,j}^k, \forall k \in K, i, j \in R \quad (3.6)$$

$$y_{i,j}^k \times x_{p_i^-, p_j^+}^k = z_{p_i^-, p_j^+}^k, \forall k \in K, i, j \in R \quad (3.7)$$

$$\sum_{h \in P^+} x_{d_k^+, h}^k = 1, \forall k \in K \quad (3.8)$$

$$\sum_{g \in P^-} x_{g, d_k^-}^k = 1, \forall k \in K \quad (3.9)$$

$$x_{p_i^+, p_i^-}^k \sum_{j \in J} q_{i,j} + \Delta_i^k = C_k x_{p_i^+, p_i^-}^k, \forall k \in K, i \in R \quad (3.10)$$

$$y_{i,j}^k (v_{p_i^+}^k + t_{p_i^+, p_i^-} + t_{p_i^-, p_j^+}) \leq v_{p_j^+}^k, \forall k \in K, i, j \in R \quad (3.11)$$

Another important group of constraints in a DARP problem concerns the time specifications. Equation (3.11) ensures that none of the requests of a car overlap in time. Equations (3.12) and (3.13) address the costumer impositions. More precisely, a customer either define at what time wants to be picked-up from a place or dropped-off and the trip time is calculated. Therefore, (3.12) satisfy the requirements of the first type and (3.13) those of the second.

$$v_{p_i^+}^k + t_{p_i^+, p_i^-} \leq d_i * T_i, \forall k \in K, i \in R \quad (3.12)$$

$$v_{p_i^+}^k \geq e_i * (1 - T_i), \forall k \in K, i \in R \quad (3.13)$$

$$\omega_{m,k} (v_{d_k^-}^k - v_{d_k^+}^k) \leq Max_m, \forall m \in M, k \in K \quad (3.14)$$

The last group of constraints covers drivers' issues. Namely, equations (3.14), (3.15) and (3.16) ensure that a driver only works within a previously defined time window.

$$\omega_{m,k} \min(v_g^k) \geq st_m, \forall k \in K, g \in V, m \in M \quad (3.15)$$

$$\omega_{m,k} \max(v_g^k) \leq ed_m, \forall k \in K, g \in V, m \in M \quad (3.16)$$

Equation (3.17) is related to the salaries of the drivers.

$$\omega_{m,k} \times s_m = \sum_{i \in A_k} rev_i, \forall k \in K, m \in M \quad (3.17)$$

Chapter 4

A Tabu Search approach for the multi-objective DARP

Typically, an airport private transfer company solves the allocation problem of Chapter 3 manually, in a daily basis. In order to find a faster and more efficient solution for the model, a Tabu Search meta-heuristic is proposed. In this chapter, the construction and the computational structure of the proposed algorithm are explained.

4.1 Tabu Search

The present TS framework is applied to a multi-objective DARP, indeed among the literature there are not so many works doing that. According to Jaeggi et al. (2008), that applies a TS to a dynamic multi-objective DARP, the majority of the works with a multi-objective function uses GA frameworks, followed by SA approaches and only then TS frameworks. In order to apply Tabu Search meta-heuristic several features have to be settled. As seen in Chapter 2, the main common features are:

- Initial Solution;
- Local Search operators;
- Aspiration Criterion;
- Tabu Tenure;
- Termination Criterion;
- Diversification Method;
- Intensification Method;
- Relaxation Method.

Unlike other models in literature (e.g Cordeau and Laporte (2003)), the proposed one does not have any relaxation method allowing to accept unfeasible solutions. Due to the exclusivity constraint referred in Chapter 3 there are less combinatorial options, not justifying the application of such methods. The flow chart present in Figures 4.1 gives a global overview of the heuristic. Note that, once the total kilometers are the sum of the kilometers made with and without customers, in order to save computational time the algorithm use an extra function (f_4 which only count the kilometers made without clients).

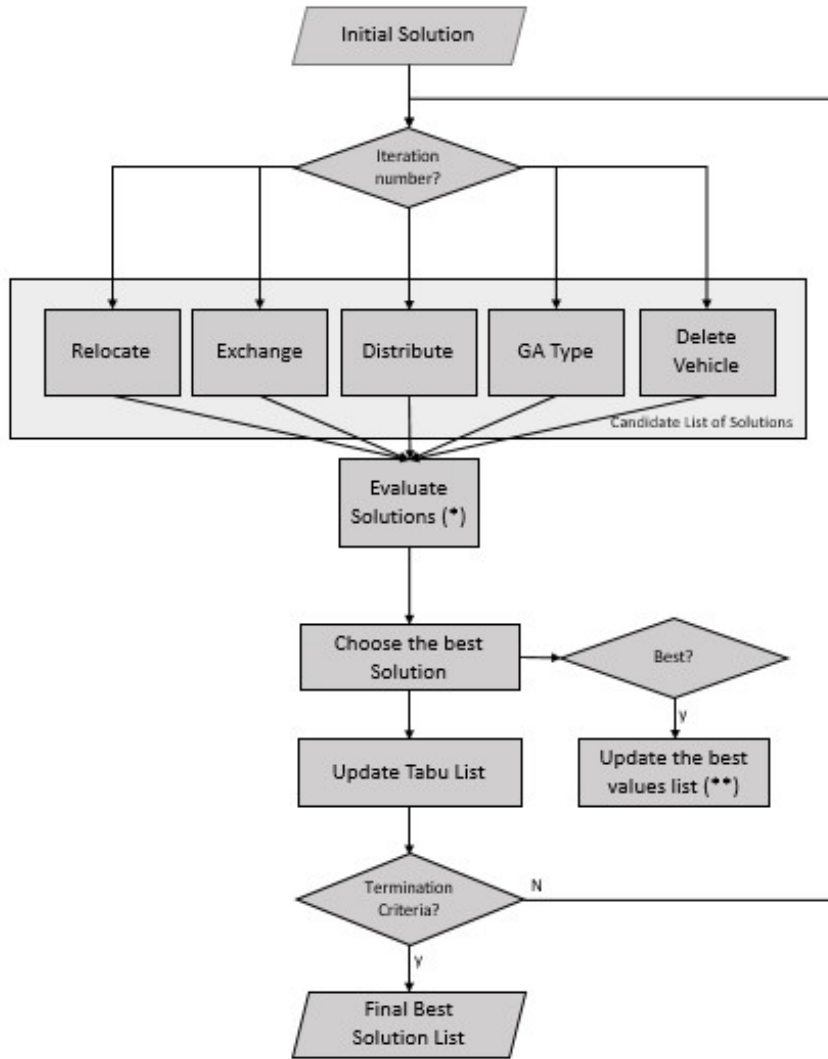
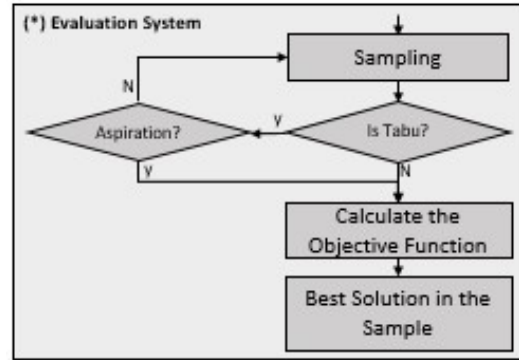
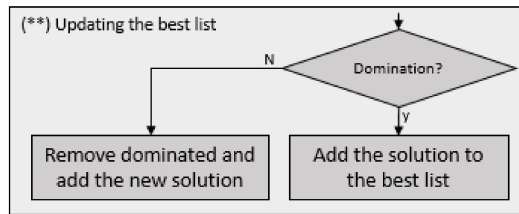


Figure 4.1: Flow chart of the Tabu Search Algorithm

Figures 4.2a and 4.2b explain in a more precise way what happens to update the best solutions list and the evaluation of the candidates process.



(a) Evaluation System



(b) Updating the best solution list system

4.1.1 Initial Solution

According to Sousa et al. (2016), incorporating good strategies in the initial solutions is important in local-based metaheuristics, meaning that the quality of the initial solution has a direct impact on the quality of the solutions found by the heuristic as well as on the number of iterations necessary to achieve those results. Therefore, similarly to authors as (Braekers et al., 2014) and Masmoudi et al. (2016), in the present work, a construction insertion heuristic is used to find an initial solution (see Chapter 2 for more information about these kind of heuristic). First, a list with the daily requests is ordered by the booking time, then, for each request, the heuristic check the vehicles that may bare it regarding time and capacity constraints. The obtained set is then sorted by the minor distance and maximum number of requests. The distances are the kilometers from the last vertex visited by a vehicle until the pickup point of the request being analyzed. This last vertex is either the location of the last drop-off made by the vehicle or the drivers' home if the request that is being evaluated is the first request. The vehicle that is allocated is the closest one meaning that the analyzed request is going to be inserted in that vehicle. In a tie scenario, the chosen vehicle is the one with the biggest number of requests already allocated in order to try to compact the requests in the shortest number of vehicles.

4.1.2 Neighborhood Definition

The search of new feasible solutions is the most expensive computational stage in a TS and in heuristics in general (Gendreau and Potvin, 2010). Similarly to what happens in others TS frameworks (e.g Detti et al. (2017)), only a sample of neighborhood is used. Besides reducing the

computational burden the added randomness can act as an anti-cycling mechanism (Gendreau and Potvin, 2010). This algorithm uses five different operators to calculate the neighborhood. This means that, for the current feasible solution, there are five different ways (relocate, distribute, delete, genetic algorithm type and exchange) to create new feasible solutions (neighbors). The neighborhood operators are inspired in the three inter-route operators, namely, relocate, exchange and 2-opt* (an exchange heuristic proposed by Potvin and Rousseau (1995)) and the elimination route operator used by Braekers et al. (2014). Due to the exclusivity constraint, and once the trips are sorted by time, there is not any application of intra-route operators. Once again using all the local search operators at each iteration could result in inefficiency in computational time. Therefore, similarly to what happens in VNS and LNS approaches, operators change at each iteration. This have further implications regarding diversification and intensification processes which are explained a following section.

4.1.2.1 Relocate

The *Relocate* operator randomly picks a vehicle (A) with trips from the current solution and tries to relocate, one by one, all the trips in other vehicles. For example, for each trip of A it checks other vehicles that may bare that trip regarding time and capacity constraints (Figure 4.3). Then, from the resulting set, a sample of possible vehicles is taken. For each reallocation move is verified if it is tabu (checks if it is a forbidden move or not). In the affirmative case, if the aspiration criterion (this criterion is explained in section 4.1.3) is not met that move is not done. This procedure is repeated for each trip in the vehicle A.



Figure 4.3: Relocate Neighborhood

4.1.2.2 Exchange

This local search operator can be separated in two stages. The first stage is the relocate operator. The second stage consist in a reverse relocate. Like it happen in the 4.4, for a vehicle (B) in the sample the operator checks a request that can be swapped with a request of the vehicle (A). From all the possibilities the operator chooses the best swap. Aspiration and feasibility of the move are also checked. Similar operators can be found in Cordeau and Laporte (2003) and Braekers et al. (2014). Nevertheless, in Braekers et al. (2014), there are two swapping operators while here only one is applied.

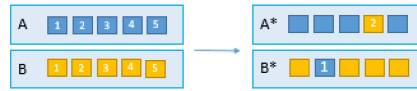


Figure 4.4: Exchange Neighborhood

4.1.2.3 Distribute

Similarly to the previous operators, a random vehicle (A) that has requests is picked from the current solution. A partition point is chosen randomly splitting the requests of (A) in two sub-routes (one with the trips from the first trip until the partition point and other from that point to the last request). If one sub-route is empty (there are not any trips before or after the partition time for the vehicle), the other sub-route is automatically chosen. In the case that both have requests one of them is randomly chosen. For each trip of the selected arc, the operator checks other vehicles that may bare that trip regarding time and capacity constraints. Then, from the resulting set it picks randomly a vehicle. If a vehicle (B) from the available set implies a tabu move another vehicle (C) from the set is picked. This procedure is repeated for each trip in the selected sub-route.

4.1.2.4 Delete

The *Delete* operator starts by the selecting a vehicle (A). For each request of the selected vehicle, it verifies other cars that may bare that request regarding time and capacity constraints once again. Then, from the resulting set it picks randomly one vehicle (B). If moving the request to vehicle B is tabu and the aspiration criterion is not met another vehicle is picked. This procedure is repeated until the vehicle A is empty (Figure 4.5). In the case that is not possible to empty A this neighborhood operator will not produce any results for that iteration. According with Braekers et al. (2014) an incomplete solution results in an increase of the computational time while hardly improves solutions quality.

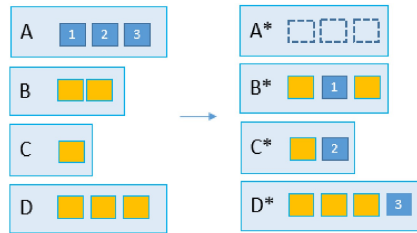


Figure 4.5: Delete Neighborhood

4.1.2.5 Genetic Algorithm Type

The *Genetic Algorithm Type* operator starts by the selecting a vehicle (A). A partition point is defined through an uniform distribution and the trips of the vehicle (A) are split in two sub-routes. Then regarding the problem constraints (time and capacity), a set of possible “partners” for the

vehicle (A) is obtained. Then, from the set of possible "partners" a sample is picked. For each "partner" (B) in the sample, is analyzed if the crossover between the two "partners" is tabu or not and if not in similarity with what happen in a genetic algorithm (section 2.3.2.2) the routes are crossed as it is represented in Figure 4.6.

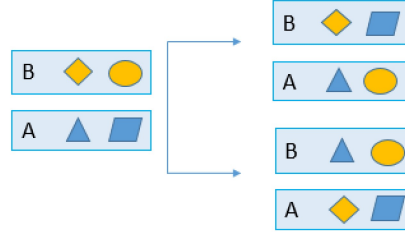


Figure 4.6: Genetic Algorithm Type Neighborhood

4.1.3 Aspiration Criterion

An aspiration criterion measures the benefit of allowing a move that is actually forbidden (tabu). In this case, the method calculates the kilometers made without customers (f_4). If the move implies a smaller number of kilometers without customers than at least one of the solutions in the list of non-dominated solutions for an iteration, the move that would be initially tabu is actually accepted as a feasible neighbor for that iteration. Without the application of this criterion, good solutions could be lost or require more iterations to be found.

4.1.4 Termination Criteria

According to Gendreau and Potvin (2010) the most common termination criteria are:

- After a certain *a priori* number of iterations is reached;
- After a predefined number of iterations without an improvement in the objective function value. The same authors, Gendreau and Potvin (2010), highlights this criterion has one of the most used;
- After the objective function reaches a threshold value. This value is also defined *a priori*;

In this application both first and second criteria are used. A maximum of 250 iteration is settled, nevertheless if no improvement is observed for 40 iterations the algorithm stops before the maximum value of iterations is reached.

4.1.5 Diversification, Intensification and Tabu Tenure

Among literature, there are many ways of implement diversification and intensification process. In DARP a famous way to diversify is the one proposed by Cordeau and Laporte (2003) where a penalty factor is added to the objective function regarding the number of times an attribute has

been added to the solution during the search and the total number of attributes. The more attributes there are, the higher a frequently added attribute should be penalized (Cordeau and Laporte, 2003). Another simple way of doing it is with the tabu tenure. For that reasons in some works the tenure size changes along the process. Longer tenures help in the diversification process while shorter ones contribute for the intensification process. For example, Melachrinoudis et al. (2007) uses a long tabu tenure during the first search process of the algorithm. Then, in a second stage were they intensify the search in a selected region from a set of regions where good quality solutions were found at the past iterations, they reduce the size of the tenure once it enables to better explore those regions (Melachrinoudis and Min, 2011). The tabu tenure settles the number of iterations in which a move is forbidden in order to prevent cycling situations. As referred before, once there is as sampling process in the neighborhood evaluation, which helps in the anti-cycling process, it is possible to apply a shorter tabu tenure to this framework. In the present algorithm the tabu tenure is 4.

In addition to the tabu list size the type of operators can have impact in the diversification and intensification process. Therefore, in the current framework, during the first 12 iterations, from the 85th till 100th iteration and from 150th till 170th, neighborhood operators which imply bigger changes (*Delete* and *Genetic Algorithm type* and *Distribute*) are applied in order to help the diversification process. Furthermore, neighborhood operators involving smaller changes (*Relocate* and *Exchange*) are used with the goal of intensify the search.

4.1.6 Dealing with a Multi-Objective Function

Although in the formulation three objectives are presented, for the algorithm, only two are critical (total kilometers without customers/total kilometers done and salaries variance between drivers), while the other (total empty seats) behave as control value. According to the information in Chapter 2, dealing with a multi-objective function can involve *a priori* information and/or *a posteriori* information. Once the preferences are not exactly quantifiable (as it should be to use a weight sum approach) two different scenarios regarding the local search process are done. One considering f_2 as the objective to be minimized at each iteration and the opposite version (transforming it in a weighted sum function the weights would be the following: $weight_{f_2, f_4} = (1, 0), (0, 1)$). It is important to keep in mind that although the preferences are not quantifiable, there is a company preference for minimizing f_4 . Therefore, for the situations where the f_2 is the object to be minimized, the framework adapts the idea behind the ϵ -constraints method (proposed by Chankong and Haimes in 1983), where the decision maker chooses one objective out of n to be minimized and the remaining objectives are constrained to be less than or equal to given target values (Caramia and Dell'Olmo, 2008). In the present case a limit of how much the "secondary" objective can deteriorate in favor of the others is set. More precisely the "secondary" objective (total kilometers done) can not get worse then 10% of the initial solution while f_2 is being minimized. Only solutions that meet this requirement can enter in the list with the non-dominated solutions that is updated at each iteration.

4.2 Software Application

The algorithm is developed in Python language with the use of Visual Studio Code (VSCode) software as IDE (Integrated Development Environment).

4.2.1 Python Language

Among several programming languages, the one used for the development of the TS algorithm is Python version 3.6.5. Python has a lower performance when compared with C++ and Java (common languages to develop this type of algorithm). A Python code can be 3-5 times slower than the equivalent code in Java and 5-10 times when compared with the equivalent in C++ (Rossum, 2018). Nevertheless, anecdotal evidence suggests that Python requires a reduced learning time and is ease of reading (Mueller, 2018).

In agreement with (Rossum, 2018), Python shines as a glue language and can also be used to prototype components until their design can be "hardened" in languages such as Java or C++. To run the model the following packages (directories of python modules) and modules (single files of code that are *imported* in the main script) are used:

- Pandas: is a data related package, allowing the use of easy-to-use data structures and data analysis tools as data-frames (Developers, 2018c);
- SciPy: The present package provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization (Developers, 2018d);
- NumPy: It is a fundamental package for scientific computing. Besides other features, includes a powerful N -dimensional array object, random number capabilities as well as useful linear algebra (Developers, 2018b),
- Geopy: This package allows programmers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources (Developers, 2018a). The package has different ways of calculating distances available. For this particular case the geodesic distance (the default one) is used. The geodesic distance assumes an ellipsoidal model of the earth and similarly to what happens in a Euclidean distance, gives the shortest distance between two vertexes on the surface. The default geocoding algorithm uses the method proposed by Karney (2013).
- copy: It is a module related with the copy of information from a variable to another variable;
- datetime, calendar and time: Those are time related modules. They allow calculations with dates as well as, for example, calculate running times;
- random: Is a module that allows the use of sampling and randomize functions;
- csv: Allow the extraction of data from csv files.

4.2.2 Visual Studio Code

The idea behind an IDE is to facilitate the writing, testing and debugging of code. Based on (vasconcellos, 2017) among the most used IDEs are Notepad++, Visual Studio (VS), Vim, Sublime Text, Eclipse, Xcode and Visual Studio Code. Although VS and VSCode have similar names they have some differences (e.g VS only works on Windows while VSCode is cross platform) . To have a close look in them visit Dotnet (2015).

4.2.3 Algorithm Structure in Python

The code of the algorithm is constituted by two main script and three specific modules for this application (Table 4.1).

Table 4.1: Algorithm modules and scripts

Title	Tasks
main_current	main script for the calculation of the present allocation method
main_allocation	main script for the algorithm allocation method applying TS framework
Getdata	module that opens and saves data from a csv files for this particular application
ObjectiveFunction	calculates objective functions and decision variables
Tabu	module concerning all the steps of the TS algorithm

As it possible to see in Figure 4.7 the tabu steps are divided in functions that are called from the main structure of the framework.

```
def TabuSearch (travel_time,initial_solution, fleet, tabu_size,n,car_avl,n_v,driver_avl,drivers, iterations,way,vehicle_driver,locations,distance, no_iter):
    iterations=1
    tabu_list=pd.DataFrame()
    no_improve=0
    best_iteration=[0,0]
    best_list=[initial]
    pair_sol=[initial, initial]
    while it<=iterations:
        if it <=7 or (it>85 and it<=92):
            gattype=Neighborhood_GAtype(pair_sol,fleet,n,way,travel_time,tabu_list,vehicle_driver,drivers,car_avl,locations,distance,n_v,driver_avl,best_list)
            delete=Neighborhood_delete()
            neighbor=[]
            for i in range(len(delete)):
                neighbor.append(delete[i])
            for i in range(len(gattype)): ...
        else:
            exchange=Neighborhood_exchange()
            relocate=Neighborhood_relocate()
            neighbor=[]
            for i in range(len(relocate)): ...
            for i in range(len(exchange)): ...
        evl=Evaluation(no_improve,iterations,it_best, neighbor,best_list,pair_sol,n,car_avl,locations,distance,n_v,driver_avl,drivers,tabu_size,tabu_list)
        best_list=evl.best_list
        best_frame=pd.DataFrame(best_list)
        pair_sol=evl.pair_sol
        no_improve=evl.no_improve
        best_iteration=evl.best_iteration
        tabu_list=evl.tabu
        if no_improve==no_iter:
            break
        iterations+=1
    final_allocation=best_list
    return final_allocation
```

Figure 4.7: Tabu Search Script

Chapter 5

YellowFish Case Study

In this chapter the results obtained from the application of the algorithm presented in Chapter 4 and its scenarios to a set of real data are presented. Therefore, before presenting the results and analysis, there will be a contextualization of the company that has provided the data.

5.1 YellowFish data characteristics

The data used in the project is provided by YellowFish Travel, Ltd. With its headquarters located in Albufeira, Algarve, the company began its activity in January of 2010 as a travel agency. Yellowish Travel provides private chauffeured transfers in Algarve, Lisbon, Alentejo and Southern Spain. As most of its clients are tourist, regularly, Faro Airport is either the pickup or drop-off point. The company has a heterogeneous fleet due to the variability of the number of clients *per* request. Therefore, Table 5.1 separates the current fleet, of a hundred and fourteen vehicles, concerning their capacity.

Table 5.1: Numbers of vehicles per capacity

Capacity	2	3	4	5	6	8	9
Number of Vehicles	1	16	56	1	1	38	1

5.2 Current Allocation Approach

Nowadays, the company closes the reservations for a specific day with 48 hours in advance because all the allocation process is done manually. This allocation process implies, regarding a booking set for a day and the available drivers (who have a vehicle allocated to them), the allocation of each request to one vehicle/driver. On a weekly basis, each driver has to give their work availability and thereafter they are allocated to a specific vehicle for that week. Although a driver can only be allocated to one vehicle, a vehicle can have a maximum of two drivers allocated to it. This happens because there are two different types of drivers. Despite of the method of allocating a driver to a vehicle being out of the scope of this project (being used as a known input value), it

is actually relevant to know the characteristics of these types of drivers. There are two different types of drivers: full-timers and part-timers. Full-timers are expected to have full availability for a week, while part-timers usually have another job and only work in specific days/times slots. Once the full-timers may have days off it is possible to find situations where a vehicle is allocated to a full-timer and a part-timer (Figure 5.1). So, although the full-time driver is not working in a particular day, the vehicle can still be profitable. Despite of the number of drivers allocated to a vehicle, it has always to start and end the day at the full-time driver's house that is allocated to that vehicle.

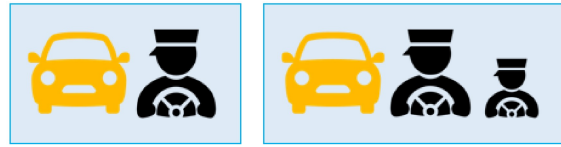


Figure 5.1: (A) Full-Timer (B) Full-Timer + Part-Timer

An employee of the company is responsible to perform the daily assignment of drivers to the services requested for that day with the use of an application similar to an excel file as the one in the Figure 5.2. The columns are the time slots and rows are the available vehicles for that particular day. Green cells represent arrivals while red ones represent departures. For these two colors a light ton means requests until four passengers and dark are services with more than four clients. Dark blue are services that do not involve the airport and Light blue represent schedules maintenance of a vehicle. In terms of the assignment process, the maintenance is treated as if it was a request.

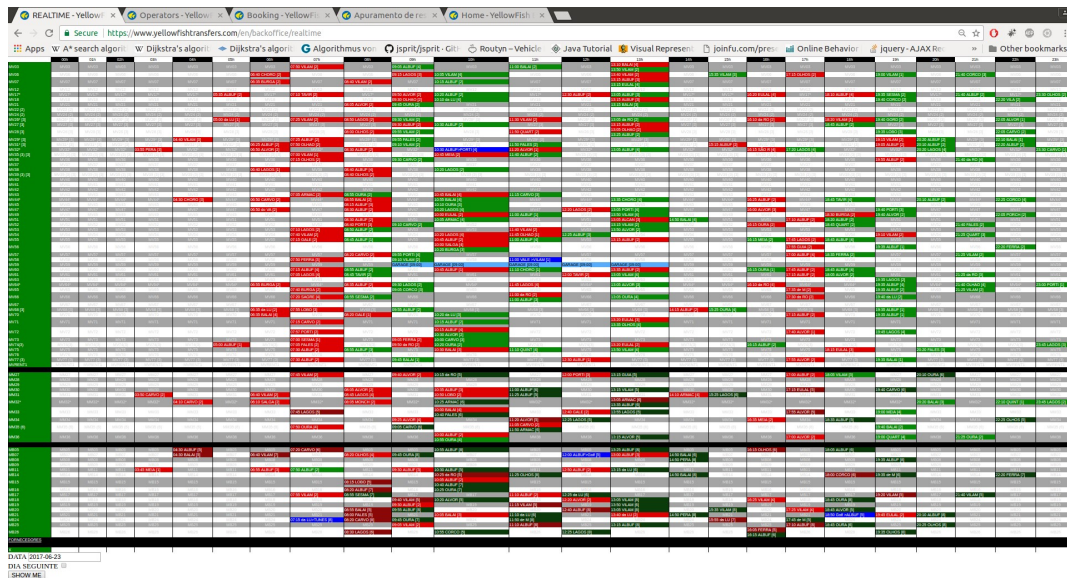


Figure 5.2: Current Allocation System

5.3 Computational Experiments

5.3.1 Dataset

YellowFish has provided datasets about requests registration (and the allocation done by the company) and non-confidential information about their drivers, vehicles and the commission payment associated with each trip locations. These informations are used to run the algorithm and compare the obtained results.

The dataset provided by YellowFish concerning the requests information has 206249 entrances between the 1st of January of 2015 and the 31st of December of 2017. It is important to emphasize that the number of entrances *per* year is not homogeneous. In fact, as it is possible to verify in Table 5.2 it seems to exists an increasing trend in the number of requests.

Table 5.2: Request *per* Year

Year	Number of Requests
2015	54816
2016	69356
2017	82077

Besides the inequality between the request registered *per* year, the same happens between days. Due to the seasonality inherent to the tourism sector, there is an huge difference in the number of requests registered in summer days when compared with winter days. Therefore, a sample of 9 days is taken. The size of the 9 instances of the sample vary between 29 and 419 requests as it is possible to see in table 5.3. Once the picked values have such an wide range in allows to check the performance of the algorithm in different situations.

Table 5.3: Requests per day in the sample. All days are from 2017.

Day	07/12	07/11	29/10	18/10	22/10	13/10	14/10	25/09	01/09
Requests	29	60	105	192	212	262	320	359	419

5.3.1.1 Data Preparation

In to order to be possible to run the algorithm, some data preparation is needed. The main changes regarding the original datasets are:

- Depots are considered the drivers' home and there are some missing values regarding this information. In such cases, the value used as reference is the Faro airport location;
- Some trip commissions (the value that a driver gets if he performs a trip from location x to location y) were not defined, being used 7.0 euros (the requests' commission mode value) in such situations;
- The objective function f_2 involves drivers' cumulative salaries and the cumulative average salary *per* type of worker (full-time and part-time). For a certain day of the month, the

cumulative average salary is given by the total amount of money awarded until that day in the company divided *per* the average of drivers available on that month. When the month changes this value is reseted to zero again. Moreover, the drivers' cumulative salaries for a day are simulated values. They are simulated through a uniform distribution using as boundary values of the cumulative average salary for that day less 20% and plus 20%. A normal distribution is not used because, in some of the months analyzed, it is not possible to ensure the central limit theorem. Therefore it is assumed that the probabilities within the chosen range are the same. The range is wide in order to test the algorithm in a worst case scenario of a big salary inequality.

5.3.2 Analysis of the Current Manual Solution

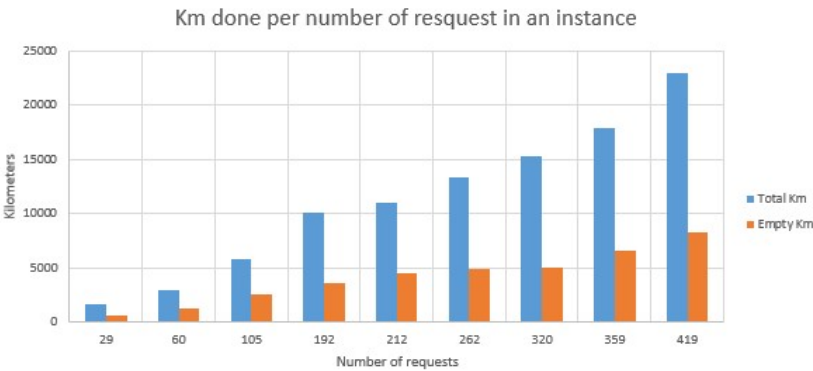
The YellowFish's manual solutions work as reference values for the algorithm results. Therefore a brief analysis on the values is presented in the current subsection. Table 5.4 summarizes the values for the objective functions as well as the ratio between the empty kilometers and the total kilometers, the number of vehicles and the average trips *per* vehicle for an instance in the YellowFish's manual allocation solution.

Table 5.4: Objective Function Values for YellowFish allocation method

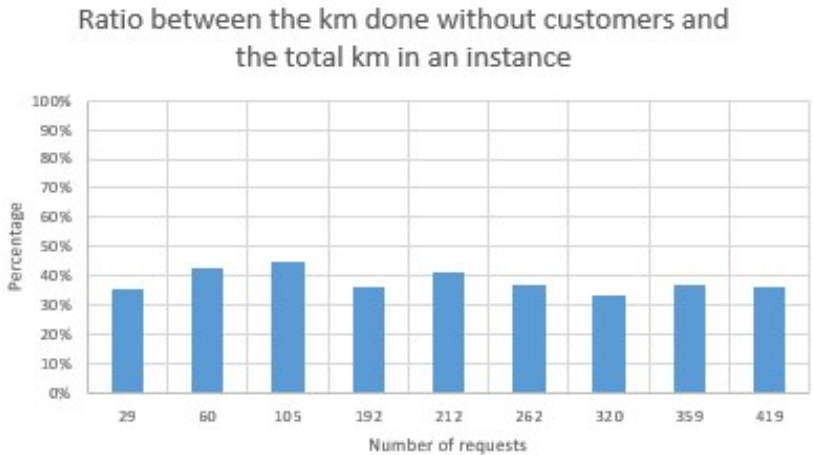
Size	Day	Total Km	Empty Km	Unused Seats	Salaries Deviation	Km Ratio	Vehicles Used	Average Trips
29	07/12	1673	594	71	6250	38%	10	2.29
60	07/11	2984	1263	157	9483	42%	15	4.00
105	29/10	5789	2612	271	47753	45%	29	3.62
192	18/10	10066	3658	480	379428	36%	45	4.27
212	22/10	11062	4540	438	547753	41%	46	4.61
262	13/10	13291	4945	630	202973	37%	52	5.04
320	14/10	19723	7440	614	112555	38%	54	5.83
359	25/09	17918	6550	896	2329383	37%	63	5.70
419	01/09	22976	8239	1246	12246	36%	68	6.16

With the exception of the kilometers ratio and two values of the average trips, all the values increase with the increase of the instances size. As far as the kilometers are concerned, it is possible to see in Figure 5.3a that both total kilometers done and kilometers done without clients have a positive linear trend. Nevertheless, because the increasing rate is not the same, being the empty kilometers smother, there is a decrease in the ratio between the total kilometers without clients and the total kilometers with the increase of the requests' number (Figure 5.3b). A possible factor to justify it is the fact that once there are more clients it is actually easier to make bigger routes with requests that are close to each other.

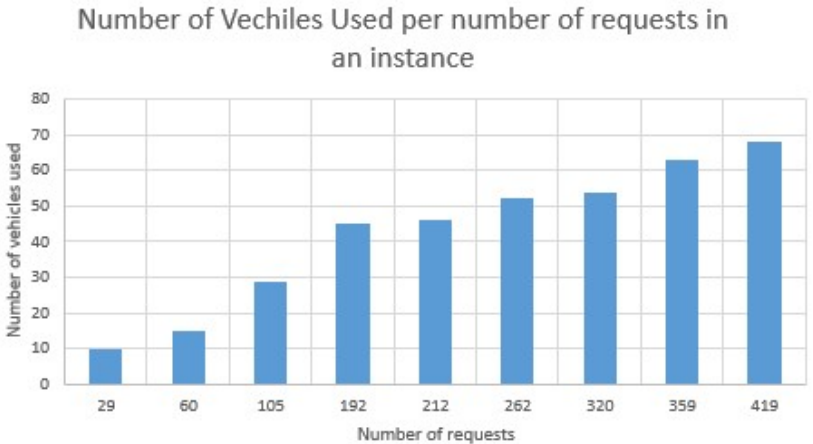
Moreover, regarding the results of a hypothesis test (with $\alpha = 5\%$), there is statistical evidence allowing to approximate the relation between the number of request/number of seats used (Figure 5.4a), the number of requests/number of vehicles (Figure 5.3c) and the number of request/average number of trips in a vehicle to a linear distribution (even if it is possible to make this approximation



(a) Total Kilometers in blue and in orange the kilometers made without customers in the vehicles



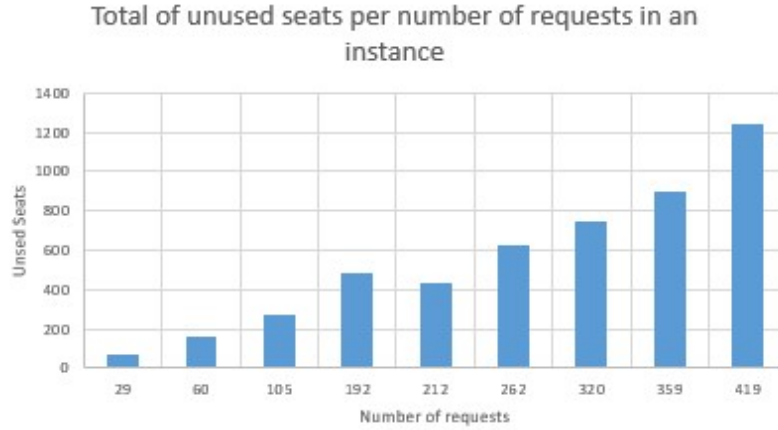
(b) Ratio between total and empty kilometers



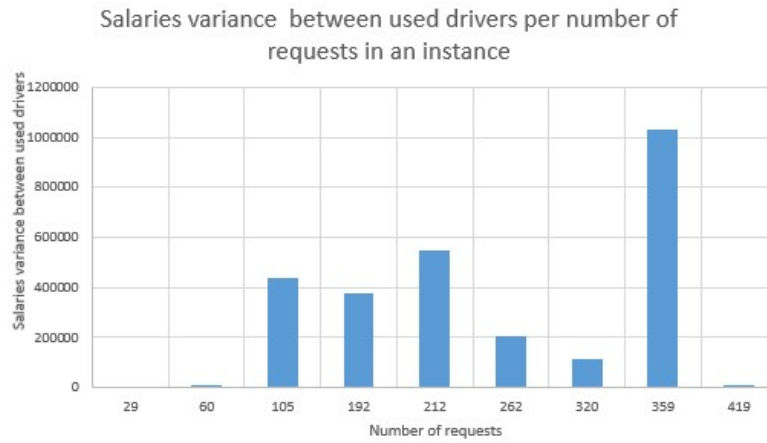
(c) Vehicles used in an instance

Figure 5.3: Graphics related with the settled objectives and vehicles used for YF allocation

only entire values can be used). This approximation, as it is evident in the Figure 5.4b, cannot be done regarding the salaries variance between drivers.



(a) Total number of unused seats



(b) Drivers' salaries variance

Figure 5.4: Graphics related with the settled objectives for YF allocation (continuation)

As explained before, the function works with cumulative values meaning that in the first days of a month and the last ones the order of the values can be quite different. For example between 22/10/17 (212) and 13/10/17 (262) even if they have a similar number of requests the value of the variation is quite different, this because the 13th is an earlier day.

5.3.3 Algorithm Results

The following subsection exposes the algorithm results for the 9 instances sample under three different scenarios. Two runs for each combination day/scenario were performed. All of them with a maximum of 250 iterations and a second stopping criterion of 40 consecutive iterations without improvement (check Chapter 4 for more information about Tabu Search criteria used in

the present framework). In the first scenario, the waiting time of a driver between two requests is assumed not having a limit and the total kilometers done is the objective to be optimized. In the second scenario, the assignment process will try to reduce the idle time of the vehicles (imposing a maximum waiting time between trips of 4 hours) while optimizing the number of kilometers. The third scenario is equal to the second one regarding the waiting time constraints. However, this time, is the salaries variance between the drivers the goal to be minimized. Besides, the results obtained for the smallest instance (07/12/17) in runs that only use some of the local operators (ways in which the algorithm find the neighborhood of a solution) are presented in the present section too. To run the algorithm a 2.60 GHz Intel Core laptop with 8 GB RAM is used. Some runs were also performed in a Intel Core 2.40 GHz laptop with 12 GB RAM in order see the impact that this can have on the computational time of the algorithm. Therefore, to be able to evaluate the performance, computational times regarding the calculation of the initial solution, the tabu procedure (excluding the initial solution) and the local search computational time for each iteration are taken.

For runs with a constant number of iterations (250 iterations) the number of non-dominated solutions varies regarding the day (Table A.1). Until the medium sized instances, in all scenarios, the size of the set of non-dominated solutions in the majority of the runs seems to increase with the number of requests in a day. When comparing the scenarios that have the minimization of the kilometers has a goal, the one which does not have constraints regarding the waiting time have a bigger number of solutions in the set. As it is a less restricted problem, these can lead to more possible combinations and thereafter to more non-dominated solutions. Solutions set sizes for the algorithm with waiting time constraints are around 21% smaller that the set without them (Appendix A). When the homogeneity of salaries is the goal during the local search process, if the constraint about deterioration of f_4 (settled in Chapter 4) is not taken in account the set would be bigger then a set with no waiting time constraints. Nevertheless, when f_4 deterioration is taken in account, the sets become actually really small as shown in the appendix B. For 22/10/17, that in one of the runs, of this scenario, had 85 non-dominated solutions, only 15 are bellow the maximum deterioration value for f_4 (for this day is 4978). For this reason, using this constraint may be not fruitful to have a general view on the behavior of the algorithm.

Table 5.5: Average number of non-dominated solutions when optimizing f_4 with waiting time constraints

Day	07/12	07/11	29/10	18/10	22/10	13/10	14/10	25/09	01/09
Set Size	25	25	39	45	55	30	14	25	12

5.3.3.1 Non-Dominated Solution Values

Tables 5.6 to 5.9 show the values for the best solution for each objective (that is the bold value of the row) and the number of vehicles used, for 5 of the analyzed days. The values concern the waiting time constraint scenario with the objective of minimizing of the total number of kilometers

done during the local search process. We can observe that, for each objective function considered alone, we obtain a diversified set of solutions, with very different values for the other, non considered, objectives. For most of the days, the solution with the minor salary variance is also one of the solutions with less vehicles used. Once the salaries variance is calculated for the drivers used and not for all the drivers available, the use of less drivers involves a smaller variance and a situation where it is easier to homogenize the salaries among the used set. Naturally, the minimization of f_2 will lead to a smaller set of driver/vehicles. Regarding the number of vehicles, the evidences show, for example in 22/10/2017, that there are more than one solution with that value. Even between the same fleet size, a significant difference between kilometers done and salaries can exist.

Table 5.6: 07/12 (29 Requests): Best Objective Function Values for the Algorithm

f_1 (Total Km)	f_4 (Empty Km)	f_3 (Unused Seats)	f_2 (Salaries Variance)	Vehicles Used	Average Trips Trips
1503	423	90	10404	21	1.90
1708	628	94	6357	11	2.64
1541	461	74	7347	18	2.27

Table 5.7: 07/11 (60 Requests): Best Objective Function Values for the Algorithm

f_1 (Total Km)	f_4 (Empty Km)	f_3 (Unused Seats)	f_2 (Salaries Variance)	Vehicles Used	Average Trips Trips
2692	971	186	26360	26	2.36
3075	1354	182	21787	25	2.38
2719	998	194	25827	22	2.37
2815	1094	191	24753	26	2.39

Table 5.8: 29/10 (105 Requests) -Best Objective Function Values for the Algorithm

f_1 (Total Km)	f_4 (Empty Km)	f_3 (Unused Seats)	f_2 (Salaries Variance)	Vehicles Used	Average Trips Trips
4927	1750	267	744319	38	3.05
5325	2148	225	583771	22	4.77
5224	2047	225	595560	22	4.77

Table 5.9: 22/10 (212 Requests): Best Objective Function Values for the Algorithm

f_1 (Total Km)	f_4 (Empty Km)	f_3 (Unused Seats)	f_2 (Salaries Variance)	Vehicles Used	Average Trips Trips
10123	3601	490	902052	47	5.35
11529	5007	548	656188	32	6.46
11184	4662	529	686051	32	6.4
10191	3669	484	900869	47	5.32

While it is obvious by the values exposed that shortening the kilometers lead to an opposite effect in the salaries variance the same does not happen with relation between the kilometers and

Table 5.10: 01/09 (419 Requests): Best Objective Function Values for the Algorithm

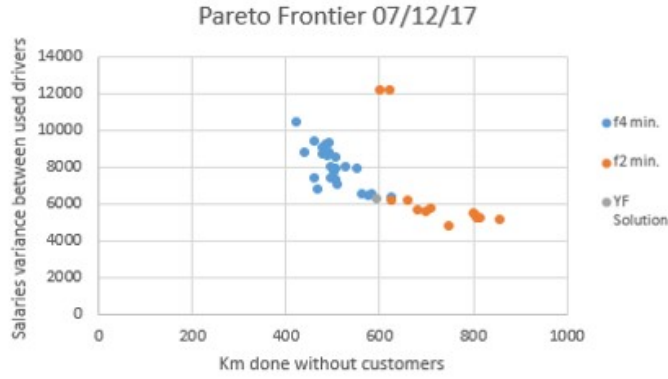
f_1 (Total Km)	f_4 (Empty Km)	f_3 (Unused Seats)	f_2 (Salaries Variance)	Vehicles Used	Average Trips Trips
Km	Km	Seats	Deviation	Used	Trips
24480	8305	1105	85078	68	6.24
25618	9443	1068	81435	64	6.55

the unused seats. Also an increase in the numbers of vehicles used in a day seems to result in a reduction in the kilometers made. Nevertheless, this might not be completely truth. For example, in the table 5.7, the difference between the solution with the minimum number of kilometers and the solution with second minimum number is only 27 km and uses 4 vehicles less, meaning that increasing one of the values does not necessarily imply worse results in the other. With the increase of the used fleet naturally the average number of trips per vehicle in a day decreases.

The Pareto fronts (Figure 5.5a, Figure 5.5b and Figure 5.5c) join values obtained in two different scenarios (the f_4 minimization run and the f_2 minimization run, both with time constraints). The blue points are the values from the f_4 minimization as main goal of the local search process scenario and orange points the on relative to f_2 scenario. The grey point is the YF solution. It is possible to see that the behavior of the algorithm changes regarding the instance size. The blue, grey and orange points for a small instance are close to each other drawing almost a continue line (being close to what is a pareto frontier, check Chapter 2). However, with the increase of the instance size there is a bigger dispersion of the values of the algorithm for both scenarios. Furthermore, when comparing the algorithm non-dominated set with YF solution, for small and medium instances the grey point seems to be (even if is not close to any of the solution clusters in a medium instance) on the frontier line, while the same does not happen for a large instance. From the initial solution until the end of the iteration process, the algorithm, once is minimizing objectives, tend to get closer from the axis. For the largest instance while YF solution is closer to the axis then the ones found by the algorithm. This combined with information of the following section may prove that for large instances it would necessary more iterations in order to converge to better solutions. Therefore, instead of a fixed termination criterion of 250 iterations, the termination criterion should change regarding the size of the instance.

5.3.3.2 Global Analysis

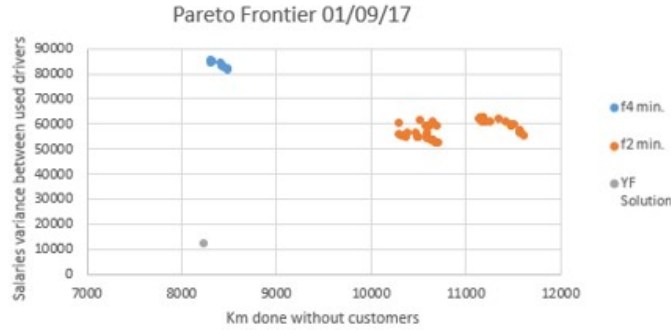
The Figures 5.6a, 5.6b, 5.6c, 5.6d are based on the solution with the less kilometers done in each solution set for the scenarios considering f_4 as the main objective to be minimized during the Tabu Search (orange bars are the values of the algorithm with the time constraint and grey bars are the values of the algorithm without time constraints). Moreover, the solution with the minor salaries variance between drivers, for the scenario where f_2 is the main objective of minimization during the iteration procedure appears in the figures as yellow bars. The values of the YellowFish solution correspond to the blue bars. Although, for each figure, the different scenarios behave in a similar way (the distributions have a similar shape), with the increase of the number of requests *per*



(a) Pareto Frontier for 12/07/17



(b) Pareto Frontier for 22/10/17



(c) Pareto Frontier for 01/09/17

Figure 5.5: Pareto charts for three different days

instance, the actual values change. Both orange and grey bars have a smoother trend concerning the number of kilometers done when compared with the YF solution. On the other hand, the yellow bars have the contrary behavior.

Continuing on the kilometers analysis, the difference between the YF solutions and the ones found by the algorithm (orange and grey bars) gets bigger with the increase of the instance size until the medium sized instances. After that the solutions get closer or in some cases (for the scenario with time constraints), even bigger than the YF solution. Within an instance, all the bars regarding the unused seats are quite close to each other. As expected, once their conflictive goals,



Figure 5.6: Objectives Functions Results comparison

the scenario where f_2 is minimized through the Tabu Search process (yellow bars) underperformed in the minimization of the kilometers done, being by far, higher than the YF solution.

The opposite scenario happens in the analysis of the salaries variance between drivers used in an instance. While orange and grey bars clearly underperformed in this objective, the yellow bars minimize the goal. With the exception of the instance of size 359, the difference between the blue bars and the yellow bar decreases or it is even positive with the increasing of the instance size. Once, a similar phenomenon happened for the orange and grey scenarios regarding the number of kilometers and for the number of vehicles, it allows to assume that the number of maximum iterations should increase in order to find better solutions for large instances. Although, the salaries variance seems to have big gaps between the solutions in an instance, it is relevant to remind that this functions are working with variances (squared values).

The number of unused seats is similar for all the cases (algorithm scenarios and YellowFish solution). Note that, for the scenarios of the algorithm ran, the size of the fleet was not an objective to be minimized.

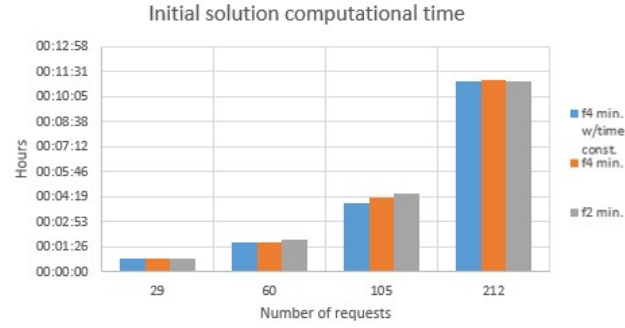
5.3.3.3 Algorithm Performance

Most of the final non-dominated solution sets contain values that were found in the final iterations proving that the algorithm converges for a local optimal solution. Nevertheless, from a total of 60 runs (from different sizes and scenarios) 10% of them has stopped before the maximum number of iterations (this happens when there are 40 consecutive runnings without any improvement, more information available in Chapter 4). All this cases were referent to scenarios with f_4 as minimization goal during the iteration process and none of this cases happened in large instances. For those cases, although the number of kilometers done were higher when compared with runs with 250 iterations, the values were already lower then the manual solution of YF.

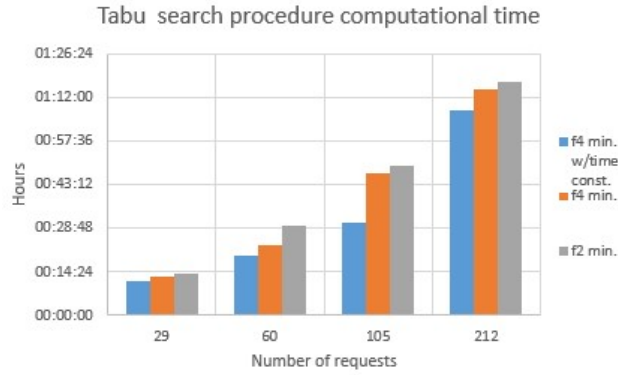
Figures 5.7a, 5.7b and 5.7c show different aspects of the algorithm performance for the average values of four instances under the three different scenarios. Not the entire sample was used, because, as referred in the beginning of the section, it was used two different laptops to run the tests. The figures show the values of the algorithm performance in the 2.60 GHz Intel Core laptop with 8 GB RAM. As expected, the time spent calculating the initial solution as well the time spent during the Tabu Search procedure increases with instances size (for all scenarios). The increasing trend seems to be exponential for the initial solution computing time. Also a exponential trend seems to be verified for Tabu Search process and for the average local search computational time, it is smother than the previous one. However, because it is a small sample a second analysis with a bigger sample would be fruitful to better understand the behavior of the algorithm performance with the increase of the instances sizes.

All of the bars have quite similar values in each one of the figures. This is specially present on figure 5.7a because the procedure of calculating the initial solution it is basically the same for all the scenarios.

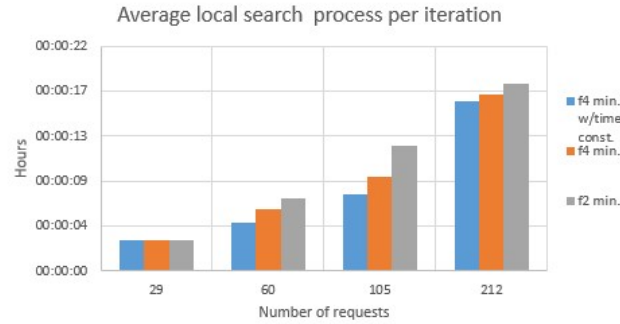
As far as the total computational time for the Tabu Search procedure and and the average for the local search stage are concerned, the grey bars (the main objective to be minimized in the



(a) Initial running time



(b) Tabu running time



(c) Average operator running time

Figure 5.7: Computational Time results

iteration process is f_2) are the most expensive, followed from the orange bars (the main objective to be minimized in the iteration process is f_4 without considering waiting time constraints) and finally the blues bars (the main goal is to minimize f_4 in the iteration process considering waiting time constraints).

Regarding the operators, *GA type* operator, *Delete Type* operator and *textitDistribute Type* operator are computationally more expensive than *Relocate* and *Exchange* operators once they involve more movements. For 07/12/17 (the smallest instance) there was a difference of 2 seconds on the average computational time of the local search stage, between a run using only *GA type*,

delete type (with an average value of 5 seconds) and *distribute type* operator and another using only *relocate type* and *exchange type* operators (with an average value of 3 seconds). Considering the algorithm behavior, this difference will probably tend to increase in the instance size, but further studies should be performed.

Table 5.11: Computational times comparison between two laptops

Laptop	Instance Size	Tabu Seach Proces	Initial Solution	Local Seach Stage Average
8 GB RAM	212	01:16:57	00:10:59	00:00:18
12 GB RAM	262	01:16:15	00:12:19	00:00:18

The values in the Table 5.11 show that despite of having more 50 requests in an instance, a computer with better processor and had similar computational times to the smaller day in the weaker computer.

Chapter 6

Conclusions and Future Work

By modeling as a Dial-a-Ride problem the allocation process of YellowFish Transfers, this project it is included in one of the most studied areas of operational research. Nevertheless, the present project has shown to be relevant both for literature and to real world due to the uniqueness of the problem characteristics and for its real application. Once it uses data from a real company (YellowFish) in order to solve a real problem, during the development process of the algorithm, several changes were made to better fit the company needs. During the first stages of the project, adapting the drivers constraints, the remuneration system and the exclusivity of the customers in a ride to a mathematical formulation was proven to be quite challenging. At the end, a three index formulation, with a multi-objective function, was the adopted formulation. The next stage of the project would pass thought choosing a solution approach and the programming language. A Tabu Search was the solution approach used once it has proven to be effective in this kind of problems over the years but it does not have many applications regarding multi-objective works. Using this framework would conciliated the possibility of achieving interesting results for the company as well as academic relevance. Python 3.6.5 was the programming language adopted.

Given the results exposed in the Chapter 5, the algorithm has proved to be able to decrease the number of kilometers while keeping similar values to unoccupied seats of the vehicle with or without waiting time constraints when compared with YF solutions. As a trade off, for these scenarios the algorithm underperforms when the salaries variance between drivers is taken in account. When this last referred goal is the subject of the iterative process the rolls switch. Nevertheless, in this situation, as the number of requests increases the difference between the YF solution and the one achieved by the algorithm seems to decrease. The used fleet seems also to be directly related with salaries variance and inversely proportional to the kilometers.

The objective function that intends to minimize the number of kilometers has a more straight convergence when compared with the salaries variance between the workers. Therefore, while operators like *GA type*, *delete type* and *distribute type* seem to slow down the process for the kilometers, they can be helpful regarding salaries once they help in diversification process. *Relocate type* and *exchange type* operators seem to provide a straightforward convergence and are useful regarding the intensification process. Explore more combinations on the ways the operators are

used may also have impact on the algorithm result. The same applies to initial solution, which according to literature, has a high impact in the iteration process. Therefore, taking a initial solution that tries to improve one of the objectives while the Tabu Search will try to optimize the others could lead to some interesting results.

Having less vehicles being used in the company would help the company to reduce its costs. Once the algorithm is able to find solutions involving less kilometers (when compared with the manual solution) and with close or even inferior fleet sizes, adding additional constraints to the problem regarding the explicit minimization of number of vehicles used in a day could be fruitful for the company.

The scenario of the algorithm with waiting time restrictions did find, in the majority of times, worse solutions for kilometers than the scenario without that additional constraint. However, as it is a small difference and it leads to a more homogeneous timetable for drivers, this scenario is considered worth to explore by the company and to be used in further extensions of the work. Some other extensions should be taken in account in a close future. Utmost, in one run, tackle more than one objective. Either by transforming the multiple objectives in a weighted function (which indeed already happened but always only with one weight different of zero) or by changing the objective to be minimized after a certain numbers of iterations. As the project it is in an early stage the trials performed only tackled one at the time to have a better understanding of the algorithm behavior. Following this idea, before making further adaptations, the results for more days should be tested.

Instead of assuming the deterministic mode, it could be transformed in a stochastic one and some analysis could be made about the impact of this transformation using, for example, as others authors have previously done, a simulation software. An useful tool to deal with the uncertainty associated with these changes would pass through the integrations with machine learning and data mining techniques.

The algorithm was implemented in Python which, as expected, does not have a very good computational performance. Therefore, as well as creating an user interface the algorithm should be converted to a programming language with a higher performance.

In order to be possible a performance comparison with others methods in literature, one possibility passes through adapt the algorithm to run in benchmark data (presented in Chapter 2).

Moreover, the model can be improved in order to become more robust. For example, the model could be adapted to some specific cases that happen in the YellowFish allocation system. More precisely, the time limitations for part-time drivers, the swap procedure between a part-time driver and a full-time one and the need of using outsourced fleet to satisfy all the requests. About the last issue, it was not necessary in the days analyzed, indeed, it is verified that the fleet was not totally is used in the analyzed days.

In conclusion, although the model has a lot of improvement opportunities, the main objectives of the present dissertation were met. The algorithm is able find better/competitive solutions in less time than the current allocation method.

Appendix A

Appendix 1

Table A.1: Non-dominates set size for the three scenarios

Day	Scenario 1	Scenario 2	Scenario 3
07/12	25	40	40
07/11	25	45	54
29/10	39	53	70
18/10	46	39	64
22/10	55	77	85
13/10	35	61	51
14/10	14	18	40
25/09	25	20	51
01/09	12	18	40

Appendix B

Appendix 2

f4	f2	f3	f1	iteration	Average trips	Vehicles Used
4595	667213	502	11117	0	6.42	33
4576	697891	501	11096	2	6.33	34
4625	675662	501	11147	4	6.36	33
4615	679160	501	11137	4	6.38	33
4664	655106	509	11186	5	6.5	30
4778	625383	505	11300	7	6.59	30
4748	628453	527	11270	9	6.65	30
4840	582900	505	11362	10	6.68	31
4857	581702	505	11379	11	6.69	31
4832	584232	505	11354	13	6.71	31
4857	615034	506	11379	54	6.66	34
5067	558855	485	11589	55	6.61	35
4964	560790	490	11486	56	6.56	35
5049	560546	518	11571	57	6.52	35
5004	564680	490	11526	57	6.49	35
5164	553181	518	11686	58	6.46	35
5069	570048	518	11591	59	6.44	35
5077	555107	518	11599	61	6.41	35
4961	590014	514	11483	62	6.38	36
4999	588548	514	11521	63	6.37	35
4873	590343	514	11395	64	6.35	35
5006	599762	510	11528	66	6.32	37
5013	615090	514	11535	67	6.3	36
5095	610818	514	11617	68	6.28	36
5067	612303	514	11589	69	6.26	36
5030	616296	514	11552	72	6.25	36
4948	616881	482	11470	76	6.24	36
5104	605844	483	11626	77	6.22	36
5095	608274	483	11617	78	6.21	36
5076	615895	480	11598	79	6.19	38
5059	629664	480	11581	81	6.16	39
5104	613187	480	11626	86	6.14	38
5154	582493	510	11676	87	6.13	36
5124	588486	494	11646	91	6.12	37
5459	549934	517	11981	92	6.13	33

5225	574177	506	11747	93	6.14	32
5434	567481	517	11956	93	6.15	33
5564	522741	496	12086	94	6.17	30
5629	505371	512	12151	96	6.19	29
5683	497679	480	12205	97	6.22	28
5683	492133	512	12205	98	6.24	29
5883	476675	520	12405	99	6.27	28
5683	497230	520	12205	99	6.29	29
5773	496132	556	12295	110	6.32	27
5786	492927	529	12308	146	6.34	28
5664	501856	468	12186	147	6.35	32
5686	510514	464	12208	148	6.35	33
5680	519695	464	12202	149	6.35	33
5675	532666	460	12197	150	6.35	33
5692	498425	460	12214	151	6.36	32
5792	496831	472	12314	153	6.38	29
6008	474546	472	12530	154	6.39	29
5960	477300	468	12482	154	6.41	28
5813	477785	464	12335	156	6.43	28
6000	477064	482	12522	162	6.44	29
6020	475579	519	12542	163	6.47	24
6128	467754	523	12650	165	6.5	24
6031	471699	523	12553	165	6.53	24
6144	458707	612	12666	167	6.57	22
5964	482435	572	12486	168	6.61	21
5918	483200	552	12440	171	6.65	20
5834	496958	544	12356	176	6.68	22
5864	489536	541	12386	177	6.71	24
5819	503571	541	12341	178	6.73	24
5946	497554	533	12468	210	6.75	25
5869	546247	484	12391	211	6.76	30
5826	580597	484	12348	213	6.76	31
5873	509889	419	12395	214	6.76	30
5875	519222	415	12397	228	6.77	31
6575	500253	420	13097	229	6.77	31
6565	510398	416	13087	231	6.76	32
6473	524149	416	12995	232	6.76	32
6560	526606	412	13082	233	6.76	33
6538	542245	408	13060	236	6.75	33
6588	505672	412	13110	237	6.75	32
6566	506779	408	13088	242	6.75	32
6598	510119	419	13120	243	6.75	31
6601	505190	419	13123	244	6.75	30
6598	522321	420	13120	245	6.76	31
6671	521543	412	13193	247	6.75	32
6712	505360	420	13234	248	6.75	32
6713	510257	416	13235	249	6.75	33
6704	544098	420	13226	250	6.74	34

Figure B.1: Non-dominated set for a run of the third scenario for 22/10/17

Red cells are solutions in which the number of kilometers done is inferior to the maximum deteriorating value.

References

- Aldaihani, M. and M. M. Dessouky (2003). Hybrid scheduling methods for paratransit operations. *Computers & Industrial Engineering* 45(1), 75–96.
- Apicella, C. L., F. W. Marlowe, J. H. Fowler, and N. A. Christakis (2012). Social networks and cooperation in hunter-gatherers. *Nature* 481(7382), 497.
- Attanasio, A., J.-F. Cordeau, G. Ghiani, and G. Laporte (2004). Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing* 30(3), 377–387.
- Berbeglia, G., J.-F. Cordeau, and G. Laporte (2012). A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. *INFORMS Journal on Computing* 24(3), 343–355.
- Braekers, K., A. Caris, and G. K. Janssens (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.
- Calvo, R. W. and A. Colomi (2007). An effective and fast heuristic for the dial-a-ride problem. *4OR* 5, 61–73.
- Caparrini, F. S. (2017). Local Search Algorithms in NetLogo. <http://www.cs.us.es/~fsancho/?e=132>, Last accessed on 2018-06-20.
- Caramia, M. and P. Dell’Olmo (2008). *Multi-objective management in freight logistics: Increasing capacity, service level and safety with optimization algorithms*. Springer Science & Business Media.
- Chankong, V. and Y. Y. Haimes (1983). Multiobjective decision making: theory and methodology. In *North Holland series in system science and engineering*, Number 8. North-Holland.
- Chassaing, M., G. Fleury, C. Duhamel, and P. Lacomme (2016). Determination of robust solutions for the darp with variations in transportation time. *IFAC-PapersOnLine* 49(12), 943–948.
- Colomi, A. and G. Righini (2001). Modeling and optimizing dynamic dial-a-ride problems. *International Transactions in Operational Research* 8, 155–166.
- Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* 54, 573–586.
- Cordeau, J.-F. and G. Laporte (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 37(6), 579–594.
- Cordeau, J.-F. and G. Laporte (2007). The dial-a-ride problem: models and algorithms. *Annals of Operations Research* 153(1), 29–46.

- Dantzig, G. B. and J. H. Ramser (1959). The truck dispatching problem. *Management science* 6(1), 80–91.
- Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation* 6(2), 182–197.
- Deflorio, F. P., B. Dalla Chiara, A. Murro, and M. A. SpA (2002). Simulation and performance of drts in a realistic environment. In *Proceedings of the 13th mini-Euro conference handling uncertainty in the analysis of traffic and transportation systems and the 9th meeting of the Euro working group on transportation intermodality, sustainability and intelligent transport systems*, pp. 622–628.
- Desrosiers, J., Y. Dumas, F. Soumis, S. Taillefer, and D. Villeneuve (1991). An algorithm for mini-clustering in handicapped transport. *Les cahiers du GERAD*.
- Deti, P., F. Papalini, and G. Z. M. de Lara (2017). A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. *Omega* 70, 1–14.
- Developers, G. (2018a). Welcome to GeoPy’s documentation. <http://geopy.readthedocs.io/en/stable/>, Last accessed on 2018-06-16.
- Developers, N. (2018b). NumPy. <http://www.numpy.org/>, Last accessed on 2018-06-14.
- Developers, P. (2018c). Python Data Analysis Library. <https://pandas.pydata.org/>, Last accessed on 2018-06-15.
- Developers, S. (2018d). SciPy Library. <https://www.scipy.org/scipylib/index.html>, Last accessed on 2018-06-14.
- Diana, M. and M. M. Dessouky (2004). A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological* 38(6), 539–557.
- Dotnet, T. (2015). What is Visual Studio Code and is it different from Visual studio 2015? <http://www.talkingdotnet.com/what-is-visual-studio-code-and-difference-between-visual-studio-2015/>, Last accessed on 2018-06-13.
- Du, D.-Z. and P. M. Pardalos (2013). *Handbook of combinatorial optimization: supplement*, Volume 1. Springer Science & Business Media.
- Du, K.-L. and M. N. S. Swamy (2016). *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature* (1st ed.). Birkhäuser Basel.
- Escobar, J. W., R. Linfati, P. Toth, and M. G. Baldoquin (2014). A hybrid granular tabu search algorithm for the multi-depot vehicle routing problem. *Journal of Heuristics* 20(5), 483–509.
- Garaix, T., C. Artigues, D. Feillet, and D. Josselin (2011). Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. *Computers & OR* 38, 1435–1442.
- Gendreau, M., J. Nossack, and E. Pesch (2015). Mathematical formulations for a 1-full-truckload pickup-and-delivery problem. *European Journal of Operational Research* 242(3), 1008–1016.
- Gendreau, M. and J.-Y. Potvin (2010). *Handbook of metaheuristics*, Volume 2. Springer.

- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research* 13(5), 533–549.
- Glover, F. (1990). Tabu search: A tutorial. *Interfaces* 20(4), 74–94.
- Gomes, R., J. P. de Sousa, and T. G. Dias (2014). A grasp-based approach for demand responsive transportation. *International Journal of Transportation* 2(1), 21–32.
- Gschwind, T., M. Drexler, et al. (2016). Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. Technical report.
- Gschwind, T. and S. Irnich (2014). Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. *Transportation Science* 49(2), 335–354.
- Gupta, A., M. Hajiaghayi, V. Nagarajan, and R. Ravi (2010). Dial a ride from k-forest. *ACM Transactions on Algorithms (TALG)* 6(2), 41.
- Ho, S. C., W. Y. Szeto, Y.-H. Kuo, J. M. Y. Leung, M. Petering, and T. W. H. Tou (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*.
- Ioachim, I., J. Desrosiers, Y. Dumas, M. M. Solomon, and D. Villeneuve (1995). A request clustering algorithm for door-to-door handicapped transportation. *Transportation Science* 29(1), 63–78.
- Jaeggi, D. M., G. T. Parks, T. Kipouros, and P. J. Clarkson (2008). The development of a multi-objective tabu search algorithm for continuous optimisation problems. *European Journal of Operational Research* 185(3), 1192–1212.
- Jaw, J.-J., A. R. Odoni, H. N. Psaraftis, and N. H. Wilson (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research Part B: Methodological* 20(3), 243–257.
- Johnson, D. S. and L. A. McGeoch (1997). The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization* 1, 215–310.
- Jorgensen, R. M., J. Larsen, and K. B. Bergvinsdottir (2007). Solving the dial-a-ride problem using genetic algorithms. *Journal of the Operational Research Society* 58(10), 1321–1331.
- Karney, C. F. (2013). Algorithms for geodesics. *Journal of Geodesy* 87(1), 43–55.
- Khelifi, L., I. Zidi, K. Zidi, and K. Ghedira (2013). A hybrid approach based on multi-objective simulated annealing and tabu search to solve the dynamic dial a ride problem. *IEE*.
- Kirchler, D. and R. Wolfler Calvo (2013). A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B: Methodological* 56, 120–135.
- Laporte, G. and I. H. Osman (1995). Routin problems: A bibliograogy. *Annals of Operations Research* 61, 227–262.
- Li, H., T. Lv, and Y. Lu (2016). The combination truck routing problem: A survey. *Procedia Engineering* 137, 639–648.
- Lim, A., Z. Zhang, and H. Qin (2017). Pickup and delivery service with manpower planning in hong kong public hospitals. *Transportation Science* 51, 688–705.

- Luo, Y. and P. Schonfeld (2007). A rejected-reinsertion heuristic for the static dial-a-ride problem. *Transportation Research Part B: Methodological* 41(7), 736–755.
- Maalouf, M., C. A. MacKenzie, S. Radakrishnan, et al. (2014). A new fuzzy logic approach to capacitated dynamic dial-a-ride problem. *Fuzzy Sets and Systems* 255, 30–40.
- Marković, N., R. Nair, P. Schonfeld, E. Miller-Hooks, and M. Mohebbi (2015). Optimizing dial-a-ride services in maryland: Benefits of computerized routing and scheduling. *Transportation Research Part C: Emerging Technologies* 55, 156–165.
- Masmoudi, M. A., M. Hosny, K. Braekers, and A. Dammak (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.
- Melachrinoudis, E., A. B. Ilhan, and H. Min (2007). A dial-a-ride problem for client transportation in a health-care organization. *Computers & Operations Research* 34(3), 742–759.
- Melachrinoudis, E. and H. Min (2011). A tabu search heuristic for solving the multi-depot, multi-vehicle, double request dial-a-ride problem faced by a healthcare organisation. *International Journal of Operational Research* 10(2), 214.
- Mladenovic, N. and P. Hansen (1997). Variable neighborhood search. *European Journal of Operational Research* 191, 593–595.
- Molenbruch, Y., K. Braekers, and A. Caris (2017). Typology and literature review for dial-a-ride problems. *Annals of Operations Research* 259(1-2), 295–325.
- Molenbruch, Y., K. Braekers, A. Caris, and G. Vanden Berghe (2017). Multi-directional local search for a bi-objective dial-a-ride problem in patient transportation. *Computers & Operations Research* 77, 58–71.
- Muelas, S., A. LaTorre, and J.-M. Pena (2015). A distributed vns algorithm for optimizing dial-a-ride problems in large-scale scenarios. *Transportation Research Part C: Emerging Technologies* 54, 110–130.
- Muelas, S., A. LaTorre, and J. M. P. Sánchez (2013). A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city. *Expert Syst. Appl.* 40, 5516–5531.
- Mueller, J. P. (2018). *Beginning programming with Python for dummies*. John Wiley & Sons.
- Paquette, J., J.-F. Cordeau, G. Laporte, and M. M. B. Pascoal (2013). Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological* 52, 1–16.
- Parragh, S. N., J.-F. Cordeau, K. Doerner, and R. F. Hartl (2012). Models and algorithms for the heterogeneous dial-a-ride problem with driver related constraints. *CIRRELT*.
- Parragh, S. N., K. F. Doerner, R. F. Hartl, and X. Gandibleux (2009). A heuristic two-phase solution approach for the multi-objective dial-a-ride problem. *Networks* 54, 227–242.
- Parragh, S. N., J. Pinho de Sousa, and B. Almada-Lobo (2015). The dial-a-ride problem with split requests and profits. *Transportation Science* 49(2), 311–334.
- Potvin, J.-Y. and J.-M. Rousseau (1995). An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society* 46(12), 1433–1446.

- Prins, C., P. Lacomme, and C. Prodhon (2014). Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies* 40, 179–200.
- Psaraftis, H. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science* 14(2), 130–154.
- Psaraftis, H. (1988). Dynamic vehicle routing problems. bl golden, aa assad, eds. *Vehicle Routing: Methods and Studies* 16, 223–248.
- Qu, Y. and J. F. Bard (2015). A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. *Transportation Science* 49, 254–270.
- Rahmani, N., B. Detienne, R. Sadykov, and F. Vanderbeck (2016). A column generation based heuristic for the dial-a-ride problem. In *International Conference on Information Systems, Logistics and Supply Chain (ILS)*.
- Reinhardt, L. B., T. Clausen, and D. Pisinger (2013). Synchronized dial-a-ride transportation of disabled passengers at airports. *European Journal of Operational Research* 225, 106–117.
- Ritzinger, U., J. Puchinger, and R. F. Hartl (2016). Dynamic programming based metaheuristics for the dial-a-ride problem. *Annals of Operations Research* 236(2), 341–358.
- Ropke, S., J.-F. Cordeau, and G. Laporte (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks* 49(4), 258–272.
- Ropke, S. and D. Pisinger (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* 40, 455–472.
- Rossum, G. V. (2018). Comparing Python to Other Languages. <https://www.python.org/doc/essays/comparisons/>, Last accessed on 2018-06-18.
- Ruohonen, K. (2013). Graph theory. tampereen teknillinen yliopisto. originally titled graafiteoria, lecture notes translated by tamminen, j., lee, k. C. and Piché, R.
- Savelsbergh, M. W. P. and M. Sol (1995). The general pickup and delivery problem. *Transportation Science* 29(1), 17–29.
- Schilde, M., K. F. Doerner, and R. F. Hartl (2011). Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. In *Computers & OR*.
- Schilde, M., K. F. Doerner, and R. F. Hartl (2014). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. In *European Journal of Operational Research*.
- Sexton, T. and L. Bodin (1985). Optimizing single vehicle many-to-many operations with desired delivery times: I. scheduling. *Transportation Science* 19(4), 411–435.
- Sousa, T., H. Morais, R. Castro, and Z. Vale (2016). Evaluation of different initial solution algorithms to be used in the heuristics optimization to solve the energy resource scheduling in smart grids. *Applied Soft Computing* 48, 491–506.
- Tonci Caric, H. G. (2008). *Vehicle Routing Problem*. IntechOpen.
- Toth, P. and D. Vigo (1997). Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science* 31(1), 60–71.

- Umbarkar, A. and P. Sheth (2015). Crossover operators in genetic algorithms: A review. *ICTACT journal on soft computing* 6(1).
- Urban, K.-P. (2006). A guided simulated annealing search for solving the pick-up and delivery problem with time windows and capacity constraints. *International Journal of Logistics Research and Applications* 9(4), 369–381.
- Van Hentenryck, P. (1989). Constraint satisfaction in logic programming.
- vasconcellos, P. H. (2017). Top 5 Python IDEs For Data Science. <https://www.datacamp.com/community/tutorials/data-science-python-ide>, Last accessed on 2018-06-15.
- Wang, X. (2014). *Operational Transportation Planning of Modern Freight Forwarding Companies: Vehicle Routing Under Consideration of Subcontracting and Request Exchange*. Springer.
- Wilson, N. H., R. W. Weissberg, and J. Hauser (1976). Advanced dial-a-ride algorithms research project. Technical report.