Vehicle sharing and workforce scheduling to perform service tasks at customer sites

Author: Pol Arias Melia
Supervisors: Prof. Jiyin Liu & Dr. Rupal Rana

2018

Contents

List of Figures

List of Tables

Acronyms

ABS	Agent Based Simulation
BA	Based Admissible
B&B	Branch and Bound
BC	Branch and Cut
BCP	Branch and Cut and Price
BKS	Best Known Solution
BSA	Best Solution After
BSB	Best Solution Before
BSS	Best Solution Sharing
CP	Constraint Programming
CPP	Car Pooling Problem
CSP	Car Sharing Portland
CVRP	Capacitated Vehicle Routing Problem
CWS	Clarke's and Wright Saving Algorithm
DARP	Dial-A-Ride Problem
DP	Dynamic Programming
DVRP	Maximum Distance Vehicle Routing Problem
EV	Electical Vehicle
FBA	First Based Admissible
ILS	Iterated Local Search
LNS	Large Neighbourhood Search
LP	Linear Programming

m-CTP	Multi-vehicle Covering Tour Problem
MDVRP	multi-Depot Vehicle Routing Problem
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
MIRP	Maritime Inventory Routing Problem
MPVRP	multi-Product Vehicle Routing Problem
OBS	Our Best Solution
OBSNS	Our Best Solution Not Sharing
PDVRP	Pick up and Delivery vehicle Routing Problem
RCWS	Randomized Clarke and Wright
SA	Simulated Annealing
SAV	Shared Autuonmous Vehicles
SP	Set Partitioning
ST	Service Territories
TS	Tabu Search
TSP	Travelling Salesman Problem
UK	United Kingdom
VNS	Variable Neighbourhood Search
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Time Windows
VSWSP	Vehicle Sharing and Workforce Scheduling Problem

Acknowledgements

Professor Jiyin Liu, I could make a very long list of all the positive things you have taught me during this PhD rollercoaster experience, but to keep it simple, thank you for choosing me to work with you and Dr. Rupal Rana for these three last years. Few people know how stubborn I am and how much patience someone working with me needs. Rupal, thank you for always saying the truth and being honest about everything I was working on. Tracey and Aly, you have been the coolest administrators I have ever had. You took care of me every time I needed help and allowed me to rant as much as one could possibly do. Good times.

To my closest friends in Loughborough. Dr. Thanos 'the grumpy' Goulas, what a three years these last ones have been. Thank you for always being there and supporting me. Arturo, you have redefined the concept of patience, and I thank you for that. Vasilis, you need to learn Spanish, then we can go to Mexico. Bader, I abandoned this office, but I did not abandon you, let us grab a juice when we meet. Sharon, you are a badass, one of the strongest women I know and the worst at replying to texts and being punctual. Zoe, through good and bad times we have always been there, so keep it up and finish this. To the ones I left back home, you are too many and I do not have enough space, so, Jordi, Tomas, Cesc, Pau, Victor, thanks for making me feel as though we never parted ways when we see each other. Andrea, the other badass woman I had the pleasure to cross paths with, I am super proud. Ivo, Litus, and Alsida, now you have to respect me more. Sinead, it has been already five years, and it seems like we never left Edinburgh -please continue being the kindest person I know. Finally, thanks Dani, I do not think I could have made it until here without your continuous support. You have been a mentor, a friend and a coach, but you still output 300W more, I will get there.

To my family, I love you and miss you every day. You always support me on whatever I want to do, you listen to me, and always give me good advice. This would have not been possible without you.

Abstract

Most of the research done in the Vehicle Routing Problem (VRP) assumes that each driver is assigned to one and only one vehicle. However, in recent years, research in the VRP has increased its scope to further accommodate more restrictions and real-life features. In this line, vehicle sharing has grown in importance inside large companies with the aim of reducing vehicle emissions. The aim of this thesis is to study different situations where sharing vehicles brings an improvement. Our main study focuses on developing a framework that is capable of assigning different workers to a common vehicle, allowing them to share their journey. We introduce a mathematical programming model that combines the vehicle routing and the scheduling problem with time constraints that allows workers to share vehicles to perform their activities. To deal with bigger instances of the problem an algorithm capable of solving large scenarios needs to be implemented. A multi-phase algorithm is introduced, Phase 1 allows us to solve the non-sharing scheduling/routing problem whose aim is to find the best schedule for workers. Phase 2 will merge the allocated workers into common vehicles when possible, while Phase 3 is the improvement procedure of the algorithm. The algorithm is tested in three different settings; using workers as drivers, hiring dedicated drivers, and allowing workers to walk between jobs when possible. Results show that sharing vehicles is practicable under specific conditions, and it is able to reduce both the number of vehicles and the total distance, without affecting the performance of workers schedule.

Chapter 1

Introduction

In recent years, vehicle sharing services have become an attractive option for governmental bodies, new transportation companies, and customers with specific needs (i.e carpooling, one day rent) that can be fulfilled by sharing one vehicle. From a government perspective, there is the need to reduce both carbon emissions and congestions which affect most of the cities in the world (Tom (2018)), an example of this problem can be seen in the United Kingdom (UK), as seven of their cities are listed among the top 110 worldwide most congested cities.

One of the main reasons is the continuous increase of vehicles in the commercial activities that are contributing to increased emissions. Thus, companies aim to efficiently program their routes and schedules to improve the efficiency from both the vehicles and the workers time perspective. To deal with this problem certain number of countries in Europe have implemented new policies through energy taxes to encourage companies to use their resources more efficiently. In the case of UK (Gov (2016)), where companies can be exempt from paying certain taxes through the application of schemes demonstrating that the company is operating under a more efficient and friendly environmental framework. Other highly populated cities have introduced pro-carsharing policies. On 19th of September 2010, the city of New York (Liu et al. (2015)) approved car-shared vehicles to use up to 20% of existing parking spaces in off-street parking facilities and up to 40% of parking spaces within public car parks.

The potential environmental benefits associated with vehicle sharing economies can be regarded as a fundamental component from an organizational and sustainable point of view, especially with the development of new economies and new urbanization areas. Particularly in highly developed urban communities, various drivers support the instigation of new policies ranging from new environmental regulations to the emerging demand towards more sustainable transport solutions, from consumer, corporate and government institutions. As stated in Hart et al. (1997), "sustainable development will constitute one of the biggest opportunities in the history of commerce.".

Due to its relative novelty, research on the theory behind the context of sharing economies and mobility systems has been scarce. Despite the growing demand of such sharing services and the development of new technologies for a more sustainable transportation network, there is a lack of a formal description of the problem at hand and its implications. One of the aims of this thesis is to introduce an overview and a formal definition of such services from an industrial perspective.

The need for optimizing road transportation occurs in both the public and private sectors, constituting a major challenge for most developed regions. Under these circumstances research on the VRP emerged. The objective of the VRP is to generate the routing of a set of vehicles so that a performance measure such as total distance or time is minimized under certain constraints. The most common restriction is the limit in vehicle capacity while satisfying the customers demand. The studies on VRP not only focus in this specific problem but has extended to new variants of the problem, such as Green VRP, Heterogeneous VRP, Time Windows VRP, and the VRP with vehicle sharing presented in this thesis.

Sharing within private institutions arises when workers are allowed to share their companies vehicles. Commonly, it can be found in big metropolitan cities (Agatz et al. (2011)) as workers tend to share their personal vehicle to commute to work. In the work presented in this thesis, we examine the problem where a set of workers complete a number of assigned tasks at customer premises located in different places. The company aims to use as few vehicles as possible to complete these tasks, by the means of workers sharing vehicles. Workers depart from a central depot (can be understood as the headquarters) and need to complete their work schedule under specific constraints. Rather than each worker using their own vehicle, we encourage sharing of vehicles if the locations of the tasks that need to be served allow this.

Sharing for such services can appear in two forms, workers as drivers, or using dedicated drivers to transport workers between tasks. The present thesis aims to contribute to the current research on the application of both of these options under a formal description and an in-depth analysis of its applicability. Furthermore, we allow workers to walk between tasks to recreate newly develop strategies used by companies aiming to reduce vehicle usage.

1.1 Objectives

Field service companies commonly operate under the assumption that each technician (referred to as worker thereafter) is assigned to a specific vehicle for his exclusive use to serve a number of customers, hence, workers either own the vehicles or are provided with one by the company. The main aim of this thesis is to develop a methodology to tackle the vehicle sharing and workforce scheduling problem for field service companies. As this problem has not been studied in the literature we define and formulate it and propose different solution approaches. Therefore the following objectives have been defined:

- **To develop a mathematical model for the problem.** To the best of our knowledge no formal description of this problem has been previously introduced. We will develop a mathematical model which aims to formulate the vehicle sharing and workforce scheduling problem using workers as drivers. One of the advantages of using exact methods to solve the problem is that it allows us to confirm that the solution found is the optimal one. On the other hand, the time needed to solve such complex problems increases dramatically when the instance size increases. To validate the model, we will run some small instances to give us insight into possible solutions and test that the model is correct.

- **To develop a three-phase heuristic solution framework.** Once the instances increase in number of nodes, there is the need to implement more efficient algorithms which will allow us to solve bigger problem sizes. We will develop a heuristic solution framework which consists of three phases, similar to the cluster first route second based algorithms as the complexity of synchronizing workers with vehicles makes the problem arduous to solve. Phase 1 will solve a variation of the capacitated vehicle routing problem with side time constraints. To do so, we will use Juan et al. (2011) algorithm to obtain the best routing possible between tasks for each worker assigned to a specific vehicle. In phase 2, we will redefine the concept of savings list given by Clarke and Wright (1964) then try to merge each one of the already assigned workers and build a list of the best mergings possible between all workers. Each worker will be merged until there are no more possible. Finally, phase 3 will try to improve the solution given by the previous phase, by destroying part of the solution and repairing it as found in the Large Neighbourhood Search. To validate our algorithm, we will compare the results obtained between sharing and not sharing vehicles. For small instances of the problem, we will solve the non-sharing instances using an exact method which will allow us to compare to the optimal solution of non-sharing vehicles.

- **To test the solution method by comparing with the best non-sharing solution.** We will test if having a worse assignment for workers leads to a better sharing solution, i.e., reducing the number of vehicles used even further. To study this case, we will introduce two more algorithms to create the non-sharing solution so we can compare how they affect the final sharing solution. Therefore, we introduce a cluster based algorithm using a similar idea to the k-means algorithm, which assigns jobs to workers considering the total distance driven. The second proposed method is the classical Clarke and Wright's Savings (Clarke and Wright (1964)) algorithm, which will offer us a medium quality solution. Additionally , we will implement an Iterated Local Search and randomized shuffling method to compare the performance of each one. The comparison of the three approaches will give us an idea of how good our solution is as we do not have an optimal solutions to compare our results.

- **To investigate the effects of different parameter settings.** Several parameters can be considered based on realistic settings to study their effects. In this thesis we will study the effect of having different geographically distributed instances using clustered and non clustered data sets and show how the duration of the jobs has a major impact on the shareability for each instance.

- **To study the cases of including dedicated drivers and allowing short-distance walking.** Traditionally sharing vehicles appears in cases such as carpooling where a person provides their vehicle to be shared with a number of co-workers. Following this idea, we will introduce our first approach to sharing vehicles within the working schedule, as workers will be driving the shared vehicle. There are other frameworks allowing vehicle sharing, the idea of using dedicated drivers has been used in some other problems (i.e bus drivers), following this concept we will modify the proposed heuristic to make it capable of solving the vehicle sharing and workforce scheduling problem using dedicated drivers. Furthermore, we will adapt the mathematical model from using only workers as drivers to using dedicated drivers as well. Finally, a common strategy to reduce problem instances and simplify the problem while adapting it to real life decisions is the inclusion of using short-distance walking as a method to cluster nodes. An in-depth study will also be presented comparing the results from previous approaches.

1.2 Structure of this Thesis

This thesis consists in 7 chapters. In this first chapter we have introduced the necessary background information for the reader to understand the importance and the necessity to use newly available technologies to improve and develop new approaches for a more sustainable future. We have briefly introduced the problem we are aiming to solve and shown the main objectives of this thesis. The remainder of this thesis is structured as follows.

Chapter 2 introduces the literature review related to our problem. Firstly, we will introduce the VRP and the two most common variants (Capacitated Vehicle Routing Problem (CVRP) and Vehicle Routing Time Windows (VRPTW)) to showcase the main methodologies used to solve these types of problems which are closely related to ours. Also we will review literature on different types of vehicle sharing problems which currently exist, and present a novel and simplified classification for these problems.

In Chapter 3 the description for the vehicle sharing and workforce scheduling problem and its main features are provided. To define the problem we present a Mixed Integer Linear Programming (MILP) formulation. To solve small instances of the problem we will use the Gurobi Optimizer. We will also show that the VRPTW is a subset of our problem and that any feasible solution for the former is a feasible solution for the latter one.

In Chapter 4 we develop the heuristic approach to solve big instances of the problem. To do so, we use a multi-phase algorithm which in case of not finding a sharing solution will provide the company with a good non-sharing solution. This algorithm will be compared to some benchmarks from Christofides et al. (1979) showing that we find really good solutions for the non-sharing phase. Also, the data sets used will be introduced and all the corresponding parameters will be explained. A discussion of the results and the comparison between sharing and non-sharing will be presented at the end of the chapter.

Chapter 5 develops the idea of using dedicated drivers as a method for sharing vehicles. This concept has been previously studied in different problems. In this chapter, we will show how to modify the previously introduced mathematical model to be adapted for the case with dedicated drivers. Also, the heuristic approach is modified and a complete study is shown on how using drivers might affect the sharing capabilities.

Chapter 6 will present how using short-distance walking interacts with sharing vehicles. To include this feature, we have developed a heuristic pre-process which will cluster nodes within a reasonable walking distance. We will present a study to show how this procedure influences both the non-sharing and sharing solutions.

Finally, Chapter 7 ends with the summary and conclusions of this thesis, its contributions and some future research lines.

Chapter 2

Literature Review

In this chapter we aim to introduce some of the main concepts and problems studied in the literature which are similar to ours, and identify where our research fits in current literature. We will start by defining what the VRP is, its origins and how it has evolved through the years. We then present the concept of Rich Vehicle Routing Problem, which focuses on solving more realistic features in the VRPs problems. To the best of our knowledge there are no classification that aims to differentiate properties and the meaning of sharing in such problems and hence we propose a new classification for the vehicle sharing problems and a possible definition for each one of the classified groups.

2.1 Vehicle Routing Problem

The Vehicle Routing Problem is considered one of the most important combinatorial optimization problems. Combinatorial optimization focuses on problems of finding a solution from a discrete finite set of feasible solutions, which either maximizes or minimizes an objective function. To understand the relevance of the VRP it is important to introduce the TSP.

The TSP is considered one of the first and most studied combinatorial optimization problems. The problem has gained much attention in the research world due to the simplicity to express it but the difficulty to solve it. The description of the problem can be summarized as: there are m nodes (which are often called cities) to be visited by a salesman and there is a cost of c_{ij} associated with direct travelling between each pair of nodes (i, j). The objective is to minimize the total cost of a tour starting from a base node (home city), visiting every other node exactly once and returning to the base node. The problem is NP-hard, and the largest instance solved to optimality has been 85,509 cities. Applegate et al. (2006) introduce what until now is the most powerful solver for the large scale TSPs, and the algorithm is called Concorde. A more in-depth explanation about this problem can be seen in the early works of Lawer et al. (1985), Reinelt (1994) and Cook (2012). Fig. 2-1 shows a graphic representation of an example TSP and its solution.

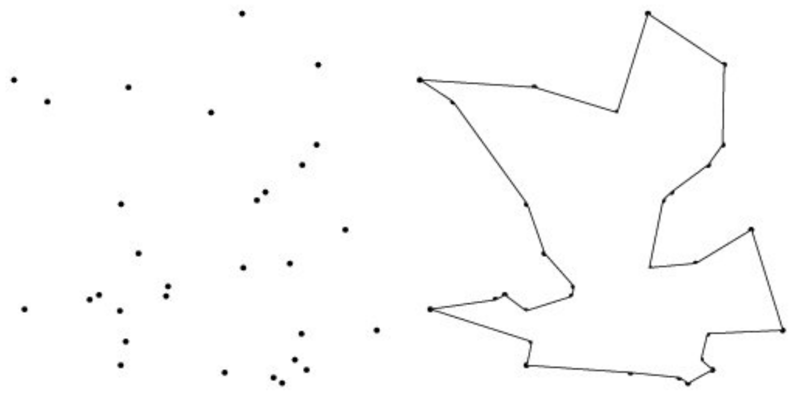

Figure 2-1: Example of a TSP instance

The VRP can be seen as an extended and more complex TSP with similar features. It is defined in the form of a graph $G = (V, A)$ where $V = \{0...., n\}$ is the set of vertices representing the different nodes (cities) with vertex 0 being the depot, and A is the set of arcs each linking a pair of nodes. For every arc (i, j) where $i, j = 0, 1, ..., n$ and $i \neq j$ there is associated a distance cost, and these costs can be presented in a matrix form $C = (c_{ij})$. Depending on the variation of the problem this cost can be taken as a travel cost or as a travel time. The objective of the VRP is then to minimize the cost of a set of vehicles with conditions:

- Each city in vector V has to be visited exactly once by one vehicle;
- All the routes start and end at the depot;
- A vehicle cannot stop twice at the same non-depot node.
- Some additional cosntraints, commonly known as side constraints, are satisfied.

The most common side constraints are:

- Capacity constraints: a non-negative weight (known as demand) d_i is given to each node other than the depot and the sum of weights in any vehicle route cannot exceed the vehicle capacity. Capacity-constrained VRPs will be referred to as CVRPs first defined by Dantzig and Ramser (1959);
- Total time restrictions: the length of any route may not exceed a maximum bound L; this length is made up of travel times c_{ij} between nodes i and j , and of stopping times t_i at each city i on the route. Time or distance constrained VRPs will be referred to as Maximum Distance Vehicle Routing Problem (DVRP);
- Time windows: city i must be visited within the time interval $[a_i, b_i]$ and waiting is allowed at each city. They may introduce the precedence constraints between pairs of cities; city i may have to be visited before city j.

Many different formulations have been presented in the literature during the last few decades, the two most important ones are based on two indexes (often called two index flow) and three index formulations seen in Toth and Vigo (2002a). There exist important differences between the two formulations. In the two index formulation the fleet is only modelled implicitly, i.e., even though we know an edge $\{i, j\}$ is used in the solution, it is not specified which vehicle will travel through it. Thus, this kind of formulations cannot model specific vehicle constraints such as different capacities, associated depots (multi depot), or different costs for different vehicles. However, the main advantage is that they provide non-redundant representations, meaning that there does not exist symmetric solutions as a result of numbering the vehicles. On the other hand, three index formulations have one more dimension of complexity and can lead to symmetries. Fischetti et al. (1997) present some symmetry breaking constraints that can be added to the model but often by only using enumerative or MIP-based approaches, so the problem remains intractable. However, for the purpose of our work the three-index formulation is the one that suits our goals better as it can differentiate between vehicles and it is easier to add specific constraints too. Therefore, to formulate the problem presented in this thesis we will use a variation of the three index formulation presented below.

As its name suggests, the three-index formulation has a binary variable of the form x_{ijk} that models the movement of the vehicles over the arcs; in other words, if

$x_{ijk} = 1$ vehicle $k \in 1..K$ moves through arc $(i,j) \in A$. Moreover, another type of two-index binary variables y_{ik} is introduced indicating whether or not vehicle k visits the customer $i \in V$.

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k \in K} x_{ijk} \tag{2.1}$$

subject to:

$$\sum_{k \in K} y_{ik} = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.2}$$

$$\sum_{k \in K} y_{0k} = K \tag{2.3}$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} = y_{ik} \qquad \forall i \in V, \forall k \in K \tag{2.4}$$

$$\sum_{i \in V} d_i y_{ik} \leq C \qquad \forall k \in K \tag{2.5}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ijk} \leq y_{hk} \qquad \forall S \subseteq V \setminus \{0\}, h \in S, k \in K \tag{2.6}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in V, k \in K \tag{2.7}$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in V, k \in K \tag{2.8}$$

$$\tag{2.9}$$

The objective function (2.1) is the travelling cost of the vehicles, which is to be minimized. Constraints (2.2) ensure that each node is visited exactly one time. Constraint (2.3) forces all the vehicles to be used. This constraint can be relaxed so the problem can also minimize the number of vehicles used. Constraints (2.4) are what are commonly known as the flow conservation constraints. This means that if a vehicle is assigned to a node, it has to go there and leave that node. Constraints (2.5) ensure that the demand of each node is satisfied and the capacity of the vehicles is not surpassed. This set of constraints can be modified so each vehicle has different capacities, which would introduce the Heterogenous VRP (Taillard (1999)). Constraints (2.6) are the coherence constraint between the flow (x_{ijk}) and assignment (y_{ik}) variables. If x_{ijk} is 1 then y_{ik} has to be 1. Finally, Constraints (2.7) and (2.8) indicate that both type of variables have to be binary.

Despite its easy-to-state features, this problem is known to be computationally arduous to solve. In this context, based on complexity theory, the VRP and its variants are shown to be NP-hard (Lenstra and Kan (1981)). In practice, if a problem is NP-hard, this means that we cannot guarantee to find an optimal solution efficiently. Thus, as stated in Garey and Johnson (1979), it is unlikely to have a polynomial time algorithm that solves the problem optimally.

Nowadays, due to the increase in computational power and the possibility to produce a good heuristic solution with more efficient algorithms, the research focus on VRP has moved to what it is called Rich VRP (Caceres-Cruz et al. (2015)). Rich VRPs consider new realistic and innovative VRP variants. These new problems can deal with a broad range of new constraints such as uncertainty behaviours, dynamic schedules, heterogeneous fleets or time windows. Finally, some Rich VRPs are integrated with other problems such as inventory, scheduling, or location problems (Lin et al. (2014)). In our case, our problem mainly deals with time windows constraints, capacity constraints and vehicle sharing. The basic VRP can be considered as a special case of our problem. As the basic VRP is NP-hard, our problem is also NP-hard. The probelm features make the research for new algorithms and heuristics a key interest (this can be seen in Solomon (1987)).

The VRP was first defined in Dantzig and Ramser (1959) as a generalization of the TSP and since then the VRP has attracted a huge amount of attention in the research community. There are two main variants of the problem that have been studied the most; CVRP and VRPTW, this is due to their easy-to-describe and difficult-to-solve feature and their wide applications. In this section, we have identified the main variants related to our problem and we will present the relevant articles published up to date. Moreover, the literature review is separated into two main blocks taking into account how the problem is solved, exact methods or approximate algorithms.

2.1.1 CVRP and VRPTW

The most studied variants are the basic VRP, the CVRP and the VRPTW, because of their high complexity level and their wide applicability in real situations. These problems share close similarities to our problem and have gone through an extensive evolution from the beginning of the study on them. In this section we will discuss how the study has evolved through the years both in variants leading to the now known as Rich VRP and its solving methods.

The CVRP was introduced in Dantzig and Ramser (1959), each vehicle visiting a node has to fulfil a demand d_i assigned to that node and each vehicle has a maximum capacity C_i that cannot be surpassed. To the best of our knowledge, the first formulations to solve the problem using exact methods and up to now, the most successful, have been the two index, two commodity flow or the set partitioning formulation. We next review some of the different variants and solving methods for CVRP problems.

Christofides et al. (1981) presents two main approaches for solving the CVRP; using branch and cut and dynamic programming. They use cutting techniques by applying Lagrangian Relaxation (Geoffrion (1974)) to their problem and calculating lower bounds to reduce the search space. The second method they use includes the concept of q-routes which are special cases of dynamic programming relaxation that produce bounds to combinatorial problems. They use what they call a (q,i)-path which starts from the depot and visits a set of cities until city i with a demand of q, using this idea, they can solve the problem by applying dynamic programming. They were able to solve problems up to 25 cities.

Fisher and Jaikumar (1981) present a new heuristic for the VRP based on Bender's (Geoffrion (1972)) decomposition technique. Basically, they reformulate the problem so two main problems can be contained in the VRP. They study a generalized assignment problem (GAP), which assigns customers to vehicles by relaxing some of the VRP constraints and the TSP which determines the best vehicle route for each vehicle. The procedure of the algorithm follows an iterative process solving the GAP master problem and later the TSP for the best route. While the authors do not prove optimality, they manage to solve problems up to 199 customers, and have the advantage to improve the efficiency of the heuristic by implementing better algorithms for both separate problems. These improvements have been reported in Martello and Toth (1990) and Desrochers et al. (1992) for the VRPTW.

Laporte et al. (1985) introduce two integer programming formulations depending on the complexity of the problem. They use both constraint relaxation techniques and a new subtour elimination constraint to solve the problem more efficiently. The first step of the algorithm defines a subproblem which relaxes the integer constraints to find quicker results, and puts it to a queue, and the best value known from the subproblem is stored. If the queue is empty, the algorithm stops as it has reached a pseudo optimal solution. Otherwise solves the problem by using linear programming. If the new found solution in better than the previously stored, then it will save the subproblem and will repeat the entire process. Otherwise, the algorithm checks for possible subtour elimination constraints, generates them and will try to solve again by using linear programming. If there is no subtour elimination constraints it verifies if the solution is integer. If it is integer, the solution will be stored as the best found so far. If no integer solution is found the algorithm branches on the fractional values and creates a new subproblem, which will be inserted in the above mentioned queue.

Augerat et al. (1998) present an implementation of an Linear Programming (LP) using a cutting plane algorithm with relaxation techniques. The algorithm as described in their paper, works as an iterative process, at each iteration they solve a linear program with degree constraints (the number of times a vehicle goes from node i to j) and some capacity inequalities. If the optimal solution of the LP is found and corresponds to a k-route then the problems is solved to optimality. Otherwise, the authors tighten the problem by adding a set of new constraints which are violated by the optimal solution found. They repeat all the steps until an optimal solution is found. By using this procedure, they are able to find optimal solutions for problems up to 135 customers.

Ralphs (2003) describes another branch and cut algorithm based on the two-index formulation using cutting planes for the capacity constraints. Their aim is to improve the separation problem of the capacity constraints which is shown to be NP-Hard by Augerat et al. (1995).

Lysgaard et al. (2004) introduce a branch and cut algorithm also based on the two index formulation which is strengthened by valid inequalities. Some of the inequalities used to improve the lower bound of the LP solution include rounding capacity, generalized capacity, framed capacity, strengthened comb, multistar, partial multistar, extended hypotour inequalities, and Gomory mixed integer cuts. This algorithm led to the finding of optimal solutions for three problems never solved in Augerat et al. (1998).

Baldacci et al. (2004) introduce a two-commodity flow formulation of the CVRP which is based on the two-commodity flow formulation presented by Finke et al. (1984) for the TSP. Their formulation uses flow variables y_{ij} and y_{ji} that represents the combined capacity that edge $(i, j) \in E$ carries. Hence, if a vehicle goes from i

to j then the flow variable y_{ij} represents the load of that vehicle. On the contrary, the variable y_{ji} represents the empty space on the vehicle (i.e. $y_{ji} = Q - y_{ij}$). This approach resembles how we deal with capacity constraints for each vehicle in our model, but in our case referring to the maximum number of people that can travel in the vehicle. With this formulation and solving the relaxed problem they find new and better lower bounds that are included in a branch and cut algorithm which is able to solve problems up to 135 customers.

Fukasawa et al. (2006) propose a formulation using the q-route (see Christofides et al. (1981)) which considers a cycle that covers the depot and a subset of customers, where the total demand is q. Moreover, they introduce a branch and cut and price algorithm to solve the CVRP. They solve the LP problem including only degree constraints and restricting the set of q-routes available. By including valid inequalities (as seen in Lysgaard et al. (2004)) they find a better lower bound for the LP which is integrated in an enumeration scheme to solve the problem to optimality.

Several exact algorithms have also been presented for the VRPTW. The VRPTW normally is found in problems where vehicles must arrive to the assigned nodes within a specific time window while the CVRP is only focused on the capacity aspect of the vehicle. Well documented reviews of exact methodologies for this problem can be seen in Kallehauge (2008) and Baldacci et al. (2012). The most successful exact method is based on the Set Partitioning (SP) model, where the set of routes contain any route satisfying the time windows constraints.

To solve such problems, the algorithms used for the VRPTW are based on column generation, Branch and Cut (BC) and Branch and Cut and Price (BCP) algorithms. The key component of these algorithms is the method for solving the pricing problem. This algorithm consists of finding a number of VRPTW routes of negative reduced cost with respect to the duals of the SP constraint. This problem is solved using different dynamic programming strategies to find either non-elementary or elementary routes (no customer is visited more than once).

Due to the complexity of the problems, the need of implementing more efficient techniques lead to an increase in the number of works using heuristic or metaheuristic based methods. The CWS constructive algorithm (Clarke and Wright (1964)) is probably the most cited and widely used heuristic to solve the CVRP and one of the algorithms used in our work. In future chapters, an in-depth explanation of such heuristic will be introduced, for a discussion of variants for the CWS the reader is referred to Toth and Vigo (2002b).

In Osman (1993), both Simulated Annealing (SA) and Tabu Search (TS) are implemented and the aim is to compare their performance using the same instances of the problem for the CVRP. Moreover, they improve the TS by implementing a First Based Admissible (FBA) and a Based Admissible (BA) strategy, both of which outperform the SA method in computational time and in the quality of the solution. Additionally, using these metaheuristics, better solutions are found for fourteen out of the seventeen classical problems.

Gendreau et al. (1994) introduce another TS heuristic. The algorithm consists of two main procedures; the construction heuristic for the initial solution and the TS which works as a solution improvement method. First, it uses a generalized insertion method, also called GENIUS (Gendreau et al. (1992)). GENIUS creates an initial tour choosing three arbitrary vertices, and looks for neighbourhood vertices which are suitable to be added in the tour. Then it arbitrarily selects a vertex v of the neighbourhood and adds it to the least cost position of the tour and updates the neighbourhood by taking into account that v is now in the tour. It repeats this process until all vertices are part of the tour. For the improvement heuristic they implement a TS, which once a solution is given randomly selects a number of vertices to be moved to another tour of a different vehicle. By iteratively using this heuristic and penalising some of the heuristic moves in the objective function they manage to outperform most of the existing heuristics.

While algorithms for CVRP are relevant to our problem, adding time constraints to deal with the VRPTW requires alternative approaches which have also been the subject of intensive research efforts for heuristic methods. One of the first heuristics implemented as a construction heuristic was given by Solomon (1987).This is considered to be an insertion heuristic which works as a route construction approach which selects nodes sequentially given a specific order, until a feasible solution has been found. The node selection process considers both capacity of the vehicle and time windows restrictions. A similar idea has been developed in this thesis to construct the vehicle sharing solution as time windows do not directly appear, the synchronization between vehicles and workers can be considered as a time window type constraint.

One of the techniques to optimise our initial solution is based on the work in

Shaw (1998). The author introduces the Large Neighbourhood Search (LNS) based algorithm on rescheduling selected customer visits using Constraint Programming (CP) techniques. The search method removes random workers from the schedule and then reinserts them at an optimal cost. To create more interchange opportunities, customers visits are selected so they are related. To find the optimal reschedule a branch and bound algorithm with CP is used.

While our approach does not solve each destroyed neighbourhood using an exact method we have developed a similar approach repairing the destroyed part heuristically.

Iterated Local Search (ILS) introduced by Lourenço et al. (2003) is another metaheuristic widely used for VRPs. An application of such method is shown in Hashimoto et al. (2008) for the VRPTW. Given an initial solution, and iterative improvement local search heuristic is applied. Their local search uses 2-opt, cross-exchange and or-opt neighbourhood structures. They use an improved Dynamic Programming (DP) algorithm to find the optimal reinsertion by saving information from past DP recursions, in order to reduce the search effort during the evaluation of neighbourhoods. Due to its simplicity and efficiency ILS has also been chosen for this thesis as a comparison methodology.

Finally, in Vidal et al. (2013) the authors study different problems of VRP and the solution algorithms commonly used. They start by introducing constructive heuristics and describe the most efficiently used algorithms such as sweep algorithm and CWS . Also, they introduce the main metaheuristics and local search algorithms used to optimise a given solution. Furthermore, they differentiate between VRP problems by classifying them using their features, such as assignment constraints, sequence choices and evaluation of sequences. They conclude that most of the recent works use a combination of different solving methods such as tabu search with local search, iterated local search, neighbourhood search, etc.

In the next section we will discuss the literature in Rich VRP that have similar or interesting attributes related to our problem.

2.1.2 Rich VRP

A Rich VRPs is defined in Caceres-Cruz et al. (2015) as: "a model that reflects most of the relevant attributes of a real-life vehicle routing distribution system". These attributes might include several of the following: dynamism, stochasticity, heterogeneity, multiperiodicity, integration with other related activities (e.g., vehicle packing, inventory management, etc.), diversity of users and policies, legal and contractual issues, environmental issues, etc. Some examples of such attributes can be seen in Pillac et al. (2012) where they tackle the Dynamic VRP, in which the information is known through time and the routing needs to be updated once more information is known. In Alinaghian and Shokouhi (2018) the authors present a multi-Depot Vehicle Routing Problem (MDVRP) and multi-Product Vehicle Routing Problem (MPVRP), the cargo space of each vehicle is separated into compartments which can contain certain products. Thus, as a model, a Rich VRP is an accurate representation of a real-life distribution system and, therefore, the solutions obtained for the Rich VRP should be able to be directly applied to the real-life scenario. In this section we aim to explain some works that consider some specific features found in our problem but that lack the sharing aspect.

One of the main difficulties of our problem is how to deal with synchronization constraints among workers and vehicles so they are allowed to share their vehicles. Bredström and Rönnqvist (2008) present an extension of the VRPTW by including synchronization constraints. This problem is mainly seen in the healthcare industry as doctors and nurses might have to visit patients at their homes. Each staff member has their own route and schedule, but some patients (i.e., people with wheelchairs, or heavy lifts involved) might need more than one staff member. Hence, the schedule needs to be synchronized so both workers are at the same place at the same time. The authors introduce a Mixed Integer Programming (MIP) formulation and use an iterative heuristic procedure that uses a relaxation technique with Branch and Bound (B&B) and dummy variables to improve the solution. They found solutions for medium instances, for up to 80 nodes and 16 staff members, some of which are proven to be optimal. Drexl (2012) presents a survey of the VRP with multiple synchronization constraints. In the paper, the author tries to identify these possible synchronization restrictions and classify them in groups. Mainly they are separated in five different groups; task synchronization, operations synchronization, movement synchronization, load synchronization and, resource synchronization. Moreover, they introduce an extended literature review about the exact and heuristic methods used to solve this kind of problems. They conclude that the complexity in these problems makes the use of standard solution techniques for the VRP, such as column generation, and local search complicated to apply. Hence, the majority of the papers in the survey use heuristic or metaheuristic based algorithms to solve the problem. It is important to highlight that no paper in the survey tackles vehicle sharing, but it shows the evolution of the techniques and their efficiency.

Another aspect of our work is to explicitly tackle both worker and vehicle scheduling. Freling et al. (2003) present a MIP formulation for the integration of crew and vehicle scheduling with a single depot. To solve the problem they use column generation applied to a set partitioning model, which is obtained by applying the Lagrangian Relaxation technique, that leads to decomposition of the problem, making the problem easy to handle. This is one of the first works that fully integrates the scheduling of vehicles and crew in the same problem. By using exact methods, they are able to solve problems up to 84 nodes.

Some of the restrictions in our model can be found in this work, as they explicitly differentiate between the assignment problem of workers to jobs, and the routing of the vehicle, without sharing. Another formulation is introduced by Haase et al. (2001) in the form of a set partitioning model of the vehicle and crew scheduling problem by considering the single depot case and a homogeneous fleet of vehicles. To solve the problem, they use a column generation approach embedded in a branch-and-bound procedure. Moreover, they identify problem specific properties such as cutting planes that can be added to reduce the search space. The authors also introduce accelerating techniques to increase the speed of the algorithm; omission of redundant constraints in the master problem, node aggregation within the subproblems, dynamic generation of the bus count constraints, substitution of the covering constraints. While the problem does not share vehicles or re-visit nodes it shows the advances and limitations of using exact methods for solving such problems.

Schneider et al. (2014) present the territory-based VRP with time windows. This problem consists of separating customers into clusters and building routes over the clustered space. The authors introduce two different approaches to group the different vertices into so-called Service Territories (ST); using the geographic distance between vertices and taking into account the time windows so all the vertices can be visited in the area by one vehicle. They select an initial set of seed customers and using an iterative process to add customers to those seeds creating the ST. The algorithm to solve the routing problem is based on the Solomon heuristics, in which all customers in a ST are assigned to a vehicle. They then introduce an improving phase where customers can be added to specific clusters if the time windows are viable. The main concept from this paper of grouping customers into clusters is used in this thesis to enable workers to walk between tasks; this will be seen in the later chapters.

In the next section we will introduce an overview of different types of vehicle sharing appeared in the literature.

2.2 Vehicle sharing and its definitions

There are several perspectives and definitions that are important to introduce as some of them have the same meaning with different names and some of them refer to as different problems, creating confusion to the readers. Research on this topic is relatively new starting in the late 1990s, throughout the reviewed works there seems to be two common characteristics that encouraged the research on this topic; the increase in congestion, and the technological advances allowing on-line systems to perform matching tasks. Hence, we will introduce a new classification of such problems aiming to unify them in a common framework.

For the reader to understand what different problems exist, let us introduce a new approach of classifying sharing problems, sharing resource and sharing both resource and path. Reading among the literature we obtained the characteristics which allowed us to group similar problems. In our case, we focus on sharing a vehicle (called resource), or sharing a vehicle and the trip (resource and path). The problems that commonly appear and the names given to such problems are summarized below:

- **Resource**. In this group we can find problems commonly named as bike sharing, vehicle sharing, pick and go, and free floating car sharing. An example of these can be seen as having a station with a number of vehicles (bikes or cars) and users can take them to drive to another station (one way system) or return it to the same station (two way system). In this case the inventory problem arises as there needs to be a minimum number of vehicles to match the demand of each station.

- **Resource and Path**. In this group we find names such as car pooling, taxi sharing, ridematching, ride sharing and, trip sharing. This problems are based on the Dial-A-Ride problem introduced by Cordeau and Laporte (2003), which will be explained in a later section. The main idea is that there are a number of clients which need to be picked up and dropped to different nodes by a vehicle with a certain capacity.

To the best of our knowledge there is no classification that tries to unify and explain the concepts of sharing and their main features. It is also important to observe that some works use the word vehicle sharing meaning ride sharing, hence, there is a need to introduce a standard classification to identify each one of the problems. We aim to provide a summary of what we consider to be the most relevant papers in this area from an optimisation/empirical study point of view and give an explanation of the problem features.

2.3 Resource utilisation

To the best of our knowledge Meijkamp (1998) is one of the first authors to publish a well organized paper on car sharing systems. The aim of their paper is to show how the implementation of more eco-efficient services would affect customer's behaviour. To do so, they use an empirical customer behaviour study on commercial car sharing services in the Netherlands. The results show that by applying sharing policies such as car sharing, customers use their vehicles more efficiently. Moreover, they change their behaviour by reducing their mileage and using other types of transports (bikes or public transport).

Barth and Todd (1999) propose an event based simulation model to tackle the car sharing problem applied to a resort in Southern California. By using a simulation model they are able to test different configurations of the model such as the number of vehicles, trips, etc. By testing different parameters the results show that when trying to minimize the number of vehicles used, they only need between 3 and 6 vehicles per 100 trips, but this incurs in a higher number of vehicles being relocated between stations. On the other hand, when trying to minimize the relocation movements, there should be approximately between 18-24 vehicles per 100 trips.

An empirical study presented by Katzev (2003) includes three different studies using the members that are found within their first year of joining Car Sharing Portland (CSP) as participants. The first study explains the customers behavioural patterns for joining the CSP, and two conclusions are drawn: the customers require the vehicle occasionally and they expect financial savings. The second study aims to investigate the most important predictors of trip usage. They conclude that both distance to the nearest station and length of membership were the most important factors. Finally, the third study looks at the possible difference in mileage of members. The results show that by joining the CSP the total mileage of members did not decrease, but that 26% sold their personal vehicles and 53% were able to avoid an intended purchase.

Uesugi et al. (2007) is one of the first works to deal with one-way car sharing systems using a simulation based approach. They use three different assignment procedures to balance vehicles between stations, taking into account that the origin and destination of the customers are already known. The first method is that each user drives their own vehicle from the beginning until the end of the journey. The second method is called divided assignment in which a group of people (n users) share the same destinaton, but instead of using the same vehicle, they can move between 2 and n vehicles to the their final destination. Finally, in the combined assignment a group of n users use the same vehicle from the origin to destination. To test the effectiveness of this model the authors used simulation as a validation tool. The results showed that by separating trips, it would minimize the balance problem for one way car sharing systems. Although it showed some promising results, the authors state that incentives should be considered so users behave under the proposed model.

Balancing the number of vehicles for each of the stations for car sharing is a complex problem to solve. Until now, only user based approaches have been presented. In the user based approach, the relocation of vehicles is done by the own users to a near stations or directly to the needed station. Weikl and Bogenberger (2013) introduce and categorize different balancing strategies which they group into two different approaches: user based relocation and operator based strategies. User based relocation appears when the clients have an incentive to move vehicles between stations, while operator based relocations are done by workers of the company. Moreover, they show that by mixing both methods the efficiency of the overall system may be improved.

Jorge et al. (2014) present a new mathematical model to optimize relocation operations maximizing profitability of the car-sharing service. Moreover, a simulation model is used to study different real-time relocation policies. Both of these approaches were applied to networks of stations in Lisbon Portugal. They use the optimization model to calculate the optimal relocation solution of vehicles, and use it as an upper bound for the simulation approach. Then a series of simulation runs are used to investigate different relocation strategies and compared to the results of the optimal solution given by the mathematical model. Results show that they can achieve savings using the data from Lisbon, but comparing the simulation based approach to the optimal solution given by the mathematical model the results are quite far. It is important to state that the optimal solution is given knowing all the information while the simulation uses a dynamic procedure.

Another mathematical model is introduced by Ghosh et al. (2015) tackling specifically the bike sharing problem. They state that the most common difficulties in this type of problems is that there is either congestion (more than needed) or starvation (fewer than needed) of vehicles at specific stations. To solve this, they assume that the redeployment of bikes is done by the operator. Their program is solved by a decomposition approach using Lagrangian relaxation but they can only solve a problem for up to 68 stations. To improve their results they focus on an abstraction approach using clustering algorithms (k-means) to group stations, reducing the problem size. Their methods are applied to data sets created from real data and show that they can absorb lost demand that otherwise would be missed.

Recent technological advances has allowed the shift from conventional to electric vehicles, thus, research lines have also shifted towards systems using electrical vehicles as transport method. By using this new type of cars, new constraints arise due to the limitation in mileage they can achieve, and the need to recharge their batteries. In Boyaci et al. (2015) present a mixed integer linear programming approach for the one-way vehicle sharing system, considering vehicle relocation and electric vehicle charging requirements. For larger instance of the problem, they introduce an aggregate model which creates imaginary hubs which accumulates the vehicles before distributing them again, this reduces the number of constraints arising from the problem and they are able to solve the problem by branch and bound. Along this line of research, in Spieser et al. (2016) the authros introduce a mixture of the balancing (of vehicle to stations) and routing problem for Mobility-On-Demand systems. They present a fluid-based optimisation approach, which is commonly used to describe the fluid level in a reservoir subject to randomly determined periods of filling and emptying, which can be translated to a transportation system. They apply their approach to data from Singapore, and show the tradeoffs between fleet size, rebalancing effort, and queueing effects in terms of passenger and vehicle flows.

Most of the research is focused on either one-way or round (or two)-way systems separately. Let us assume there are two stations A and B. In one way systems the client's journey is from A to B, while in round way systems the client goes from A to B and then back to A. Jorge et al. (2015) introduce a mathematical model that combines both systems to enhance round-way systems by allowing one-way trips. To do so, they choose the Logan International Airport (Chicago) as it is a great hub creating many one-way trips. They show that mixing both sharing systems allows further improvement on the net profit and more importantly reduce the number of parking stations needed in the airport for such services as more vehicles are being used instead of being parked.

Brinkmann et al. (2015) present an inventory routing problem for bike sharing systems. This problem has been studied in the last decade since bike sharing has increased its popularity in big cities throughout Europe. The problem focuses on the variation of demands that cause the stations suffer from running out or being full of unused bikes, And so customer demands may not be fulfilled. To fix and rebalance the system, a fleet of vehicles transport bikes between stations during the day. The objective is to maintain suitable level of bikes for all the stations in accordance to their demand. They introduce a MIP model that uses temporal indexes so a station can be refuelled though specific interval. But the problem is too complex to solve so they introduce a heuristic procedure that focuses on two decomposition methods embedded in a Variable Neighbourhood Search (VNS). Because they use temporal intervals they solve sequentially each interval adding the information of the previously solved one. Moreover, they create subsets of bike stations to reduce the problem size and assign them to the vehicle, and use a VNS that either inserts a new station to a subset, or exchanges two stations from two different sets.

Another study for relocating vehicles in car sharing systems is presented in Nourinejad et al. (2015). Staff members of car sharing companies are responsible to balance the number of vehicles between stations. Therefore, a staff member will drive one vehicle from one station to another, so there are enough vehicles to match the demand. The authors state, that recent models do not consider the balancing of the staff needed to reallocate vehicles between stations. To do so, they introduce a mathematical model that allows workers to move between stations by using public transport, walking or bike. The model can only solve problems up to 40 users and therefore decomposition methods are used to solve bigger instances. To validate their methodology they create randomly generated data sets, inspired in the Car2Go daily operations in Toronto.

Finally, a qualitative study has been presented by Shaheen et al. (2015) which shows the future perspectives for car sharing systems from the operators point of view. The results of the surveys show that operators think that parking availability is limiting their growth while working without reservations gives them less certainty. Moreover, there is a need to further improve the connectivity with public transport and the usage of Electical Vehicle (EV)s with charging stations.

2.4 Resource and Path utilisation

The Dial-A-Ride Problem (DARP) is a generalization of two different VRPs: the Pick up and Delivery vehicle Routing Problem (PDVRP) and the VRPTW. The main difference for this problem, compared to other VRPs, is that it considers the human perspective. DARP is defined in Cordeau and Laporte (2003) as "designing vehicle routes and schedules for n users who specify pick-up and drop-off requests between origins and destinations." A three index mathematical formulation is introduced in Cordeau and Laporte (2007):

A directed graph $G = (V, A)$. The vertex set V is partitioned into $\{\{0, 2n + 1\}, P, D\}$ where 0 and $2n + 1$ are two copies of the depot,$P = \{1, ..., n\}$ is the set of pick up vertices and $D = \{n + 1, ..., 2n\}$ is the set of delivery vertices. A request is an ordered pair $(i, n + i)$,where $i \in P$ and $n + i \in D$. To each vertex $i \in V$ is associated a load q_i , with $q_0 = q_{2n+1} = 0$, $q_i \geq 0$ for $i = 1, ..., n$ and $q_i = -q_{i-n}$ for $i = n + 1, ..., 2n$, and a service duration $d_i \geq 0$ with $d_0 = d_{2n+1} = 0$. The arc set is defined as $A = \{(i, j) : i = 0,\ j \in P$, or $i, j \in P \cup D$, $i \neq j$ and $i \neq n + j$,or $i \in D, j = 2_{n+1}$. The capacity of vehicle k is Q_k defining the maximum number of passengers on board, and the maximal duration of route $k \in K$ is denoted by T_k. The cost of traversing arc (i, j) with vehicle k is equal to c^k_{ij} , and the travel time of arc (i, j) is denoted by t_{ij} . The maximal ride time is denoted by L and the time window of vertex i is $[e_i, \delta_i]$. The model uses binary three-index variables x^k_{ij} equal to 1 if and only if arc (i, j) is traversed by vehicle $k \in K$. In addition, let u^k_i be the time at which vehicle k starts servicing vertex i, w_i^k the load of vehicle k upon leaving vertex i, and r_i^k the ride time of user i (corresponding to request $(i, n+i)$ on vehicle k). The model is then as follows.

$$\min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij}^k x_{ij}^k \tag{2.10}$$

subject to

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k = 1 \qquad (i \in P) \tag{2.11}$$

$$\sum_{i \in V} x_{0i}^k = \sum_{i \in V} x_{i2_{n+1}}^k = 1 \qquad (k \in K) \tag{2.12}$$

$$\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{n+ij}^k = 0 \qquad (i \in P, k \in K) \tag{2.13}$$

$$\sum_{j \in V} x_{ji}^k - \sum_{j \in V} x_{ij}^k = 0 \qquad (i \in P \cup D, k \in K) \tag{2.14}$$

$$u_j^k \geq (u_i^k + d_i + t_{ij}) x_{ij}^k \qquad (i, j \in V, k \in K) \tag{2.15}$$

$$w_j^k \geq (w_i^k + q_i) x_{ij}^k \qquad (i, j \in V, k \in K) \tag{2.16}$$

$$r_i^k \geq u_{n+i}^k - (u_i^k + d_i) \qquad (i \in V, k \in K) \tag{2.17}$$

$$u_{2n+1}^k - u_0^k \leq T_k \qquad (k \in K) \tag{2.18}$$

$$e_i \leq u_i^k \leq \delta_i \qquad (i \in V, k \in K) \tag{2.19}$$

$$t_{in+1} \leq r_i^k \leq L \qquad (i \in P, k \in K) \tag{2.20}$$

$$max\{0, q_i\} \leq w_i^k \leq min\{Q_k, Q_k + q_i\} \qquad (i \in V, k \in K) \tag{2.21}$$

$$x_{ij}^k \in 0, 1 \qquad (i, j \in V, k \in K) \tag{2.22}$$

Constraints 2.11 and 2.13 ensure that each request is served once by the same vehicle, while constraints 2.12 and 2.14 guarantee that each vehicle starts and ends its route at the depot. Constraints 2.15 to 2.17 define starts of service times, vehicle loads and user ride times, respectively, while constraints 2.18 to 2.21 ensure that these will be feasible.

As it can be seen the DARP also uses a three index formulation, but does not allow to share vehicles between clients. Therefore, not allowing for a vehicle to pick up another worker that has the same path as the one in the vehicle. It is also important to realize that this is a static model of the problem which will be used in this thesis. Hosni et al. (2014) discuss how the static version of the problem is more complex to solve than the dynamic one, which is why dynamic problems have attracted more interest. As presented before, this problem can have different names depending on the problem we have to deal with. A good overview of features for the ride sharing problem can be seen in Agatz et al. (2012). From an empirical point of view some studies have been proposed in Furuhata et al. (2013) where the authors introduce a review of the features considered in the ride sharing problem from a practical point of view. The authors state that while there are more tools to facilitate a ride sharing system due to the technological evolution, such as GPS, web, and mobile technologies for real-time communication, the use of this service has decreased around 10% in the past 30 years. Hence, the paper tries to explore what are the consequences of this and how this situation can be reverted. Stiglic et al. (2016) conduct a study to quantify the impact of driver and rider flexibility for sharing systems, more precisely, a single-driver, single-rider sharing systems. They conclude that the participant flexibility greatly effects the matching process, having a higher impact in low participation scenarios. Moreover, it is stated that both drivers and riders need to be flexible in terms of arrival times (between 10-15 min as seen in their results), also, from a driver perspective, it is important to be flexible on the detour they are willing to make.

Baldacci et al. (2004) are one of the first to explicitly tackle resource and path type of problems by solving and formulating the Car Pooling Problem (CPP). As they state, this is a re-interpretation of the DARP problem. The CPP is, for each day, to collect the offers of the employees willing to share their own cars and the requests of the employees that wish to be picked up. Each car driver specifies the number of places available on their vehicle and the maximum time he/she is willing to spend driving from home to the workplace. For each rider there is a penalty cost if he/she has not been picked up. Moreover, each employee specifies the earliest time acceptable for leaving home and the latest time acceptable to arrive at work. Hence, the objective function is to minimize this penalty associated with each employee not picked up. To solve the problem they use an exact method by applying a Lagrangean column generation algorithm.

More formulations of similar problems have been presented in other works which highly relate to the DARP. A taxi ride sharing system is formulated in Lin et al. (2012). This again, is a re-interpretation of the DARP, as clients must be picked up and dropped at different locations. One of the contributions is the inclusion of passenger satisfaction in the objective function in the form of the time a client needs to wait to be picked up. To solve the problem they introduce a SA algorithm but solving only up to 29 clients. Bigger instances of the taxi ride sharing problem are solved in Ma et al. (2013), where they propose a taxi sharing algorithm for a real data set of trips, in China. They use a hybrid approach with simulation and a schedule optimization algorithm to solve the problem. The problem is defined as spatio temporal index of taxis, which basically partitions the map in a grid which simplifies the scheduling problem of the taxis, and then apply an insertion heuristic that solves the matching problem for taxis and clients. Using their real data and its parameters, the authors run a series of experiments to validate their model. They found that ride sharing is viable and it can absorb 25% more taxi users, while decreasing the distance. A new type of formulation which indicates if a client goes through an edge separately from the vehicle variable can be seen in Hosni et al. (2014) where the new formulation for the static taxi sharing problem is also compared with the formulation presented in Cordeau (2006). They also introduce another formulation to tackle the dynamic problem which minimizes the addition in route distance of adding another request. Finally, Lee and Savelsbergh (2015) introduce a new formulation for using dedicated drivers for the ride sharing problem. This new formulation is a variation from the taxi ride sharing (we can assume that taxi drivers are dedicated drivers), but with the addition of maximum working hours for drivers. They use a neighbourhood search that improves their initial solution by doing some variations in the form of local search.

Another problem that deals with sharing system using resource and path, is the so called matching problem, which tries to assign passengers to vehicles. Herbawi and Weber (2012) present a mathematical formulation for the ride-matching problem with time windows. One of their contributions is that a driver can pick up another rider after picking one rider. Hence, a set of drivers will pick up and deliver riders between an origin and a destination. Each rider states the earliest time to depart from the origin and the latest time to arrive to their destination. Therefore, drivers might be able to pick up an additional rider if the time windows allow it. To solve the problem the authors have implemented a genetic algorithm, which uses one single cross-over and five different mutation operators. To test their methodology, they used data from a travel and activity survey for northeast Illinois conducted by Chicago Metropolitan Agency for Planning (CMAP). Again, in Huang et al. (2013) the authors develop a fast matching algorithm for large scale real-time ride sharing. The authors state that the algorithm can be applied to different existing services including taxi services, private vehicle sharing, elevator systems, minibus services, and courier services.

Big part of the literature deals with dynamic problems, therefore, different types of technqiues have been applied that better tackle stochastic features. Due to the nature of the dynamic problem, Agatz et al. (2011) present a simulation approach to study the dynamic ride sharing problem with data from metropolitan Atlanta. They create a rolling horizon algorithm that tries to solve a matching problem every time the information is updated with new trips. They compare their algorithm to a greedy approach that always select the match (driver with rider) with the highest saving. Moreover, they do an extensive study on how different parameters of the problem affect the matchings of riders and drivers; distance, number of drivers, etc. They show that there is a potential in applying this type of sharing techniques even with a small number of pool drivers.

Agent Based Simulation (ABS) is another type of simulation which can be used to model such systems. In Martinez et al. (2015) the authors present an ABS approach as it allows to model both clients and drivers to make decisions which benefit them. They use data from Lisbon (Portugal) to simulate their model and show that fares can be reduced up to 9% of their price. Fagnant and Kockelman (2018) adopt the agent based simulation model from Fagnant and Kockelman (2014) which investigates the usage of Shared Autuonmous Vehicles (SAV) for sharing purposes. Fagnant and Kockelman (2018) extend this previous model by allowing dynamic ride sharing, to deliver a benefit-cost analysis, including optimal fleet sizing to solve the problem.

While by using simulation some of the features of the dynamic ride sharing can be modelled, the usage of optimisation thechniques is needed to find competitive solutions. Therefore, some authors present a combination of simulation and optimisation to solve such problem. Di Febbraro et al. (2013) present a simulation and optimization based approach for the dynamic ride sharing problem. They formulate the matching of drivers and riders as a mathematical program, while a simulation iterative process focuses on the rolling horizon. The rolling horizon works as an "event trigger" procedure. They distinguish between different types of events such as bookings or cancellations that affect the number of requests, at each one of these events the optimization procedure is called. The authors show using different scenarios based on data from the area of Genoa that there is a reduction in the mean delay per user by using the dynamic approach. In D'Orey and Ferreira (2014) the authors solve a TSP with side constraints (i.e., capacity of the vehicle, etc) to tackle the addition of a newly created request for a vehicle with the minimum total distance. To validate their approach, they use a simulation based approach, to assess different parameters of the problem and see how they impact on the final output.

In recent years, given the strucural changes of cities' transport infrastrsucture, different transportation modes, improvement of the network, etc., have thrived. Therefore, new features have been tackled adding complexity to the ride shairng problem. Fahnenschreiber et al. (2016) introduce a dynamic ride sharing problem with multi modal stations. It enables connections between two different public transport stations, while sharing the trip. To solve the problem they use a travel information system that computes the possible connections between stations for a given route. For the routing part they use the algorithm called OSRM from Luxen and Vetter (2011) which uses both contraction hierarchies and Dijkstra's algorithms to find the best route. On the other hand, Teubner and Flath (2015) introduce the concept of multi hop ride sharing. The idea behind this is to create new connections between vertices by adding mid points to the route. A possible application for a 2-hop ride can be seen in the following example. Let us assume that a client wants to travel from A to B, the number of possible connections might be limited and there may not be a possible ride. Therefore, they consider the possibility of travelling from A to X (a mid point) and then from X to B. Finally, the authors study the implications of creating new connections with real data from Carpooling.com. In Alonso-Mora et al. (2017) they state that previous works do not tackle the possibility of multiple passengers in a car sharing system. Thus, they formulate the problem as a matching and then routing problem. Firstly, they dynamically create matches of vehicles to requests and then they solve a TSP for each one of the vehicles. They state that by using this approach it allows them to solve big instances, for cities such as NY.

Finally, the concept of shareability has started to gain importance within the research literautre. Santi et al. (2014) introduce the concept of shareability network which is modelled by the collective benefit of sharing as a function of passenger inconvenience. They focus on the taxi ride sharing with a data sample of New York City. The aim is to provide results showing optimal strategies for taxi sharing. Results show that New York City provides excellent shareability options while barely affecting passenger discomfort. In Tachet et al. (2017), they introduce the concept of shareability by using a mathematical formula using urban characteristics. By applying such formula it is in theory possible to measure the level of feasibility of a specific city to apply sharing systems. They state that this mathematical law can be extrapolated to other cities and they test this with New York, San Francisco, Vienna and Singapore. The results show that they follow the same structure, hence, following the same rule.

2.5 Possible Classification

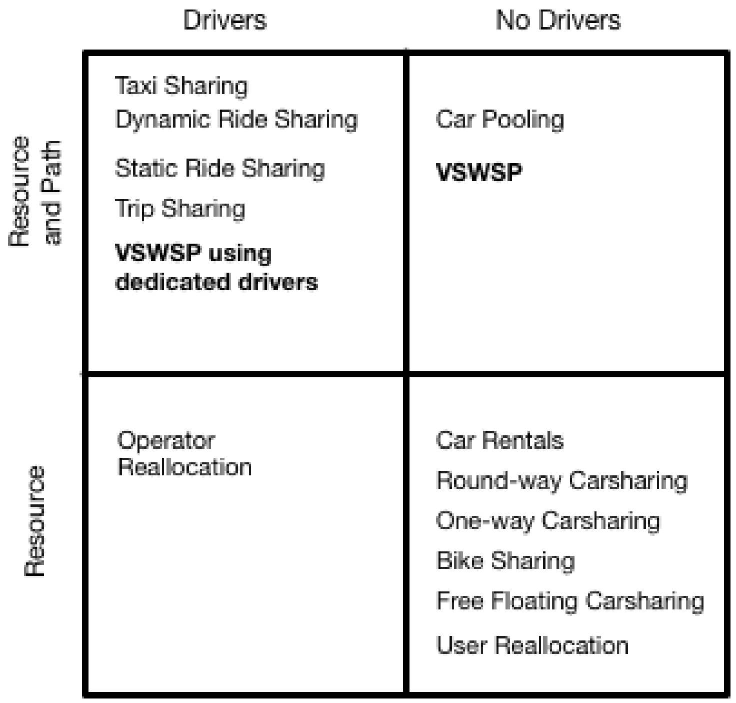

Figure 2-2: New classification of all the different problems

As shown in Fig. 2-2, we summarize our new classification framework that allows us to group all problems into four types, based on two main characteristics. Firstly, we consider in the sharing system whether the users share only a resource or share a resource and the path. Moreover, we look at another characteristic, which repeatedly appears in the literature, and consider if the system uses dedicated drivers or not.

As it can be seen, using such a simple approach, all the problems can be classified into four types. An interesting case appears when using resource and drivers, in this case a driver is a person (staff member) who is in charge of doing the reallocation of vehicles, also called operator based reallocation. As mentioned in the previous section, the case was introduced by Nourinejad et al. (2015). They highlight the importance of considering the scheduling aspect of staff members in reallocation techniques. Following this classification of sharing problems, a good representation of where our problem is located can also be clearly stated. Our main problem in the thesis can be located in the quadrant of resource and path and no drivers. Workers share the same vehicle and path without any dedicated driver. On the other hand, if we use dedicated drivers as a sharing method, the properties of our problem change while we still share both the vehicle and the path.

2.6 Summary and conclusions of the literature reviewed

To summarize all the works presented in this chapter, we present tables 2.1 to 2.3 where all the problems are clearly separated by their type of sharing methodology and a brief selection of the main characteristics for each one of the problems. In table 2.1, we introduce the papers which are linked to the resource sharing while tables 2.2 and 2.3 present the summary of papers on resource and path sharing. Our main problem is part of what we consider Resource and Path sharing, in which a number of workers share a vehicle while travelling on it at the same time. The literature reviewed tackles problems such as ride and trip sharing, which are similar to the taxi services or the so called DARP. While having some similarities with these types of problems, we introduce concepts and features such as visiting nodes more than one time as workers need to be dropped and picked up. Moreover, to the best of our knowledge there is no previous work that tries to consider vehicle sharing while assigning and scheduling a set of workers at the same time to do jobs at specific locations. Also, some literature introduces concepts such a shareability of transportation systems, which identifies how shareable cities are. While their concept cannot be directly applied to our problem, we show a different definition of how shareability can be measured within the vehicle routing problems.

Table 2.1: Classification of problems from the literature reviewed using resource

Name	Features
Barth and Todd (1999)	Event Simulation based
Boyaci et al. (2015)	MILP Electric Vehicles
Brinkmann et al. (2015)	MIP Bike Sharing VNS
Ghosh et al. (2015)	Bike Sharing MILP Langrangian Decomposition
Jorge et al. (2014)	One way carsharing Discrete Event Simulation MILP
Jorge et al. (2015)	MILP One-way carsharing Round-way carsharing
Katzev (2003)	Empirical Study
Meijkamp (1998)	Empirical Study
Nourinejad et al. (2015)	MIP One-way carsharing Staff balancing Decomposition approach
Shaheen et al. (2015)	Empirical Study
Spieser et al. (2016)	Balancing carsharing Fluid optimisation model
Uesugi et al. (2007)	One-way carsharing Event Simulation based
Weikl and Bogenberger (2013)	One-way carsharing Operator and User based algorithm for reallocation

Table 2.2: Classification of problems from the literature reviewed using resource and path (Part 1)

Name	Features
Agatz et al. (2011)	Dynamic Ride Saring Event Simulation based Matching Problem Formulation
Alonso-Mora et al. (2017)	Dynamic Ride Saring ILP Matching Problem
Baldacci et al. (2004)	Car Pooling Problem Integer Programming Lagrangean Column Generation
Di Febbraro et al. (2013)	Simulation and Optimization based Matching Problem
D'Orey and Ferreira (2014)	Dynamic Taxi Ride Sharing Simulation and Optimization based
Fagnant and Kockelman (2018)	Dynamic Ride Sharing Agent Based Simulation Shared Autonomour Vehicles
Fahnenschreiber et al. (2016)	Dynamic Ride Sharing Multimodal Stations
Herbawi and Weber (2012)	Ridematching Problem Integer Programming Genetic Algorithm
Hosni et al. (2014)	Static and Dynamic Taxi Ride Sharing MIP

Table 2.3: Classification of problems from the literature reviewed using resource and path (Part 2)

Name	Features
Lee and Savelsbergh (2015)	Ride Sharing
	Dedicated Drivers
	Integer Programming
	Neighbourhood Search
Lin et al. (2012)	Taxi Sharing
	Integer Programming
	Simulated Annealing
Ma et al. (2013)	Taxi Sharing
	Simulation and Optimization based algorithm
Martinez et al. (2015)	Ride Sharing
	Agent Based Simulation
Santi et al. (2014)	Taxi ride sharing
	Shareability Network
Stiglic et al. (2016)	Empirical Study of driver and rider flexibility
Tachet et al. (2017)	Shareability law
Teubner and Flath (2015)	Ride Sharing
	Multi Hop

Chapter 3

Problem definition and mathematical model

3.1 Introduction

To the best of our knowledge there is no work in the literature that tackles a problem similar to the one we study and introduce its mathematical formulation. In this section we will introduce some problems which do not consider the same problem as ours but have some features that can be useful and have helped characterise the formulation of our problem. The routing and scheduling problems are not only focused in road transportation, there is also a large quantity of literature that focus on maritime and rail routing and scheduling.

Song and Furman (2013) present a real life Maritime Inventory Routing Problem (MIRP). The main objective is to find the minimum cost of the routing and the assignment of cargo to each ship while considering some of the main characteristics such as, flexible cargo sizes, port draft limits, daily changing production and consumption rates, vessel loading and discharging at multiple ports, possibility of vessels revisiting ports, limited berth availability at ports, and route, cargo size and timing-based transportation costs. The problem by itself has a high complexity level due to all the constraints stated. Hence, the authors propose a new time-space formulation that uses a solution method based on a LNS optimization method and the BC algorithm. This paper introduces some interesting specific features similar to our problem as vessels might revisit ports during different time periods.

Pang and Liu (2014) formulated an integer programming model for a short sea container shipping problem which jointly decide ship routing, berth allocation at the terminals, as well as transshipment of containers to minimize the overall operating cost. Because of allowing transhipment, the route of a container can be different from the route of any ship. This is similar to the vehicle sharing problem where a worker does not need to follow the whole route of one vehicle.

Agra et al. (2015) introduce a stochastic short sea shipping problem where a company is responsible for both the distribution of oil products between islands and the inventory management of those products at consumption storage tanks located at ports. The authors focus on the stochastic nature of the problem as ship routing and scheduling is highly perturbed by the weather conditions and the unpredictable waiting times at ports. Hence, they use the sailing and the waiting times as stochastic parameters. The paper introduce a MIP formulation using two dimensional nodes as ports (hence, they can be revisited). But since the complete model is too large to be solved efficiently, it is decomposed into a master problem and one subproblem for each scenario, based on the idea of the L-shaped algorithm (Birge and Louveaux (2011)). Optimality cuts are added dynamically after the master problem is solved, and it is embedded within a sample average approximation method for the stochastic parameters. This paper specifically introduces the two dimensional nodes as ports can be revisited.

3.1.1 Problem Features

The vehicle sharing and workforce scheduling problem involves the assignment of workers to a specific vehicle in order to fulfill their work schedule. In more detail, a company providing a service (maintenance, engineering, etc) to a set of customers, has a set of tasks to be served each in a known location. A set of workers must be scheduled to undergo these activities while using a pool of available vehicles. In this context the number of tasks generally surpasses the number of workers and two main sub problems must be taken into account; the routing and sharing of vehicles and the scheduling of tasks with workers. We will refer to this problem as Vehicle Sharing and Workforce Scheduling Problem (VSWSP) with time windows.

As in most of the vehicle routing and scheduling problems, there are some features that are shared throughout all of them. There are some specific aspects that need to be introduced in order to fully understand the problem and its limitations.

- Time windows can exist in each node and delimit the time when the worker can perform the task, introduced by Solomon (1987). Time windows can vary through each node and customer. Some customers allow to have wider time windows such as "morning" or "afternoon" while some others have to be visited during a more specific time period. For our problem, this restriction can be used as a soft constraint as companies are allowed to choose their visiting time. Hence, whole day time windows will be considered.
- Synchronization between vehicles and personnel is essential in our problem. Workers are dropped to their assigned nodes but vehicles are allowed to continue their route and pick up the workers once the job is finished. Hence, it is really important that the synchronization between the vehicles and the tasks is as tight as possible so idle time is minimized.
- Static schedule. In Cordeau and Laporte (2007), definitions of static and dynamic problems are introduced. Dynamic problems allow tasks to be assigned as requests, which are revealed during the working day and routes have to be adjusted to the new requests. On the other hand, a static problem assigns tasks known beforehand. In our case, the routes and schedules are constructed knowing all the activities that must be done.

Stated above are the general features of the problem which can generally be found in some of the newest VRP variations. Next, we introduce some unique and new features found in our problem.

- Revisit nodes by vehicles is another key feature of this problem and is also what makes it really arduous to solve. A vehicle might drop a worker to their scheduled node and continue to drop another worker. After a worker finishes the job at a node, a vehicle must pick him up.
- Workers visit nodes without working there. This appears as workers share the vehicle to arrive at their respective jobs. Therefore, a worker might stop to other intermediate locations where the vehicle drops their co-workers.

- Sharing vehicles between workers. The problem focuses on a framework were workers drive the vehicles. Hence, there has to be always a worker in a moving vehicle as there are no designated drivers.
- Equipment carried by workers. In this problem one of the main assumptions is that workers carry the necessary tools to undergo each job in cases or bags.

Finally, the vehicle sharing and workforce scheduling problem can potentially have different number of objective functions. In this work, we will focus on the reduction of the vehicles' total distance travelled and the number of vehicles used.

3.2 Mathematical model

The mathematical model is based on the definition of four sets. Let $N = \{0,..,n\}$ be the set of all nodes, with node 0 representing the depot and others being customer modes to be served by the company. Let V be the set of vehicles and each vehicle $v \in V$ has an associated maximum passenger capacity of Q_v. Let K be the set of workers available to undergo the total number of jobs. Finally, let $D = \{0,1\}$ denote the set of two dimensions or duplicates of each node i: $(i0)$ and $(i1)$. To allow revisits, we have to create a two-dimensional graph to differentiate the two visits so that a vehicle can leave a worker at a customer node in the first visit and then the same or another vehicle picks him up after his job is done in a second visit. Moreover, we define a time window $[b_i, e_i]$ associated with node $i \in N$, where b_i and e_i represent the earliest and latest times for starting the task at node i. Let τ_i be the duration of job $i \in N$. For each visit to node i there is a loading/unloading time represented by β_i for workers getting on/off. Finally, for each arc (i,j), there is an assigned distance cost of t_{ij} that will be used in the objective function.

Therefore the notation list is as follows:

- N = set of all nodes.
- V = set of all vehicles.
- K = set of all workers.
- D = set of dimensions, in our case we can visit 2 times each node.

- b_i = beginning of time window for node i.
- e_i = end of time window for node i.
- τ_i = duration of job i.
- β_i = loading/unloading time for job i.
- t_{ij} = distance between node i and j.

In the formulation we consider three main binary variables; the assignment of jobs to workers, the schedule of each worker through the network, and finally, the routing of each vehicle.

- $w_i^k = \begin{cases} 1 & \text{if job } i \text{ is assigned to worker } k \\ 0 & \text{otherwise} \end{cases}$

- $x_{imjn}^{kv} = \begin{cases} 1 & \text{if worker } k \text{ goes from } im \to jn \text{ using vehicle } v \\ 0 & \text{otherwise} \end{cases}$

- $z_{imjn}^{v} = \begin{cases} 1 & \text{if vehicle } v \text{ goes from } im \to jn \\ 0 & \text{otherwise} \end{cases}$

Figure 3-1 shows a graphical example of a two dimensional representation of the problem as we might need to visit some nodes twice. As seen in the example, the vehicle visits first node 1 from the depot, then moves to node 2, to come back to 1 and finally to the depot. Therefore, we visit twice node 1. Note that, for the first visit to node 1, variable z_{imjn}^{v} is set to be z_{010}^{0} and z_{1020}^{0} . While when visiting node 1 a second time, will change this variable to be z_{2011}^{0} and z_{110}^{0}.

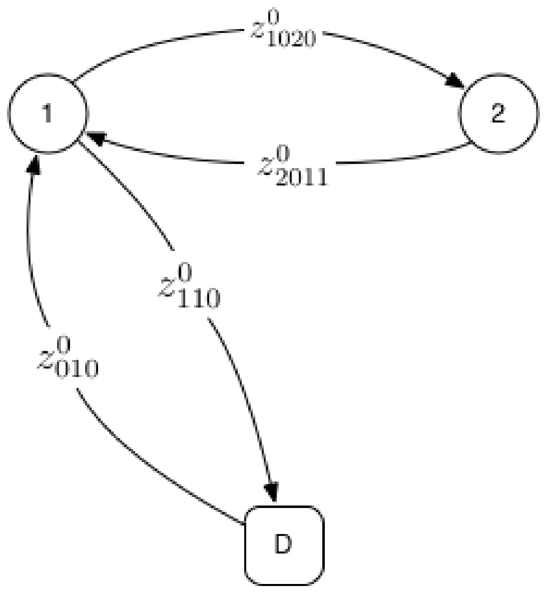

Figure 3-1: Example of a two dimensional representationf of the problem, where a vehicle visits twice node 1.

Besides these variables (assignment, scheduling and routing), the following time variables are also defined and used in the model.

- S_i is the starting time of job i.
- C_i is the completion time for job i.
- a^v_{im} is the arrival time of vehicle v at node im.
- d^v_{im} is the departure time of vehicle v from node im.
- A^k_{im} is the arrival time of worker k at node im.
- D^k_{im} is the departure time of worker k from node im.

Then, the vehicle sharing and workforce scheduling problem can be formulated as the following mixed integer program.

The objective function calculates the total distance travelled by the vehicles. The second and third term in the objective are the distances of the travelled arcs from the depot and those of the travelled arcs back to the depot, respectively, while the first term includes the distances of all other travelled arcs.

$$
\begin{aligned}
min \sum_{i\in N\backslash\{0\}} \sum_{m\in D} \sum_{j\in N\backslash\{0\}} \sum_{n\in D} \sum_{v\in V} t_{ij}\ z^v_{imjn} + \\
\sum_{j\in N\backslash\{0\}} \sum_{n\in D} \sum_{v\in V} t_{0j}\ z^v_{0jn} + \\
\sum_{i\in N\backslash\{0\}} \sum_{m\in D} \sum_{v\in V} t_{i0}\ z^v_{im0}
\end{aligned}
$$

The objective function is to be minimized subject to the constraints below.

Constraint (3.1) requires that each job must be done by one worker.

$$
\sum_{k\in K} w^k_i = 1 \qquad \forall i \in N\backslash\{0\} \tag{3.1}
$$

Constraints (3.2) to (3.4) define the flow conservation constraints for the workers. (3.2) requires that the total flow going to a node has to be the same as the total flow leaving the node. (3.3) and (3.4) ensure that each worker leaves from and returns to the depot.

$$
\begin{aligned}
\sum_{m\in D} \sum_{v\in V} x^{kv}_{im0} + \sum_{m,n\in D} \sum_{v\in V} \sum_{j\in N\backslash\{0\}, j\neq i} x^{kv}_{imjn} - \\
\sum_{m,n\in D} \sum_{v\in V} \sum_{j\in N, j\neq i} x^{kv}_{jnim} - \sum_{m\in D} \sum_{v\in V} x^{kv}_{0im} = 0 \\
\forall k \in K, \forall i \in N\backslash\{0\}
\end{aligned} \tag{3.2}
$$

$$
\sum_{n\in D} \sum_{v\in V} \sum_{j\in N\backslash\{0\}} x^{kv}_{0jn} = 1 \qquad \forall k \in K \tag{3.3}
$$

$$\sum_{m\in D}\sum_{v\in V}\sum_{i\in N\setminus\{0\}} x_{im0}^{kv} = 1 \qquad \forall k \in K \tag{3.4}$$

Constraint (3.5) works as the capacity constraint. The number of workers on a vehicle at any time cannot be more than the maximum capacity of the vehicle (Q_v).

$$\sum_{k\in K} x_{imjn}^{kv} \leq Q_v$$
$$\forall i, j \in N, i \neq j, \forall v \in V, \forall m, n \in D \tag{3.5}$$

Constraint (3.6) is introduced as a coherence constraint. If a worker is assigned to node j then there have to be a vehicle going from some node to j. This constraint allows a worker to have a route, in which, he does not work at all the nodes he goes through.

$$w_{jk} \leq \sum_{i\in N\setminus\{0\}}\sum_{m,n\in D}\sum_{v\in V} x_{imjn}^{kv} + \sum_{n\in D}\sum_{v\in V} x_{0jn}^{kv}$$
$$\forall j \in N\setminus\{0\}, k \in K \tag{3.6}$$

Constraints (3.7) to (3.9) ensure that for each arc $i \rightarrow j$ and worker k there can be at most one assigned vehicle traveling through the arc with the worker.

$$\sum_{v \in V} x_{imjn}^{kv} \leq 1$$

$$\forall i, j \in N \backslash \{0\}, i \neq j, \forall k \in K, \forall m, n \in D \tag{3.7}$$

$$\sum_{v \in V} x_{0jn}^{kv} \leq 1$$

$$\forall j \in N \backslash \{0\}, \forall k \in K, \forall n \in D \tag{3.8}$$

$$\sum_{v \in V} x_{im0}^{kv} \leq 1$$

$$\forall i \in N \backslash \{0\}, \forall k \in K, \forall m \in D \tag{3.9}$$

Constraint (3.10) ensures that the completed time of a job cannot be earlier than its starting time plus the job duration.

$$C_i \geq S_i + \tau_i \qquad \forall i \in N \backslash \{0\} \tag{3.10}$$

The starting time of a job cannot be earlier than the beginning or later than the end of its time window. These are guaranteed by constraints (3.11) and (3.12), respectively.

$$S_i \geq b_i \qquad \forall i \in N \backslash \{0\} \tag{3.11}$$

$$S_i \leq e_i \qquad \forall i \in N \backslash \{0\} \tag{3.12}$$

Constraints (3.13) to (3.15) define the flow conservation for the vehicles. All the vehicles that go to a node *im* have to leave it.

$$z_{im0v} + \sum_{n \in D} \sum_{j \in N \backslash \{0\}, j \neq i} z^v_{imjn} - \sum_{n \in D} \sum_{j \in N \backslash \{0\}, j \neq i} z^v_{jnim} - z^v_{0im} = 0 \quad \forall v \in V, i \in N \backslash \{0\}, m \in D \tag{3.13}$$

$$\sum_{n \in D} \sum_{j \in N \backslash \{0\}} z^v_{0jn} = 1 \quad \forall v \in V \tag{3.14}$$

$$\sum_{m \in D} \sum_{i \in N \backslash \{0\}} z^v_{im0} = 1 \quad \forall v \in V \tag{3.15}$$

Constraint (3.16) ensures that if a worker goes from node im to jn, he has to be assigned to a vehicle v traveling on that arc.

$$x^{kv}_{imjn} \leq z^v_{imjn} \quad \forall v \in V, \forall i, j \in N, i \neq j, \forall m, n \in D, \forall k \in K \tag{3.16}$$

A vehicle needs to have a worker driving it, hence, in constraint (3.17) if a vehicle goes from node im to jn it has to be assigned to at least one worker k.

$$\sum_{k \in K} x^{kv}_{imjn} \geq z^v_{imjn} \quad \forall v \in V, \forall i, j \in N, i \neq j, \forall m, n \in D \tag{3.17}$$

Constraint (3.18) makes sure that the departure time of vehicle v from node im plus the travel time t_{ij} is less than or equal to the arrival time to node jn, if it travels on that arc. M is a number big enough that the constraint holds in case that there is no vehicle travelling from im to jn.

$$d_{imv} + t_{ij} \leq a^v_{jn} + M(1 - z^v_{imjn}) \qquad \forall v \in V, \forall i, j \in N \backslash \{0\}, i \neq j, \forall m, n \in D \quad (3.18)$$

Similar to constraint (3.18) but considering workers, constraint (3.19) forces the departure time from node im plus the travel time t_{ij} to be less than or equal to the arrival time to node jn, if that arc is used by the worker.

$$D_{imk} + t_{ij} \leq A^k_{jn} + M(1 - \sum_{v \in V} x^{kv}_{imjn}) \qquad \forall k \in K, \forall i, j \in N \backslash \{0\}, i \neq j, \forall m, n \in D \quad (3.19)$$

Constraints (3.20) and (3.21) ensure that the differences between the departure and the arrival times for vehicle v and worker k, respectively, at each node have to be greater than or equal to the loading/unloading time.

$$d^v_{im} - a^v_{im} \geq \beta_i \qquad \forall v \in V, i \in N \backslash \{0\}, \forall m \in D \quad (3.20)$$

$$D^k_{im} - A^k_{im} \geq \beta_i \qquad \forall k \in K, i \in N \backslash \{0\}, \forall m \in D \quad (3.21)$$

Constraint (3.22) links the starting time of a job with the arrival time of the worker, such that if a worker k is assigned to a job i, then the start time of the job cannot be earlier than the time of the worker arriving at the job.

$$S_i \geq A_{im}^k + M(w_i^k + \sum_{j \in N\backslash\{0\}} \sum_{n \in D} \sum_{v \in V} x_{jnim}^{kv} - 2)$$
$$\forall i \in N\backslash\{0\}, \forall k \in K, \forall m \in D \quad (3.22)$$

Constraint (3.23) links the completion time of a job with the departure time of the worker, so that if a worker k is assigned to a job i, then the worker's departure from the job cannot be earlier than the completion of the job.

$$C_i \leq D_{im}^k + M(2 - w_i^k - \sum_{j \in N\backslash\{0\}} \sum_{n \in D} \sum_{v \in V} x_{imjn}^{kv})$$
$$\forall i \in N\backslash\{0\}, \forall k \in K, \forall m \in D \quad (3.23)$$

Constraints (3.24) to (3.27) link the departure and arrival times of a worker with the respective variables for vehicles he takes. Thus, the departure time of worker k in node ik, has to be equal to the departure of the vehicle picking him up at the node. This applies also for the arrival time of the worker and the vehicle dropping him to the node.

$$D_{im}^k \geq d_{im}^v + M(\sum_{j \in N} \sum_{n \in D} x_{imjn}^{kv} - 1)$$
$$\forall i \in N\backslash\{0\}, \forall k \in K, \forall m \in D, \forall v \in V \quad (3.24)$$

$$D_{im}^k \leq d_{im}^v - M(\sum_{j \in N} \sum_{n \in D} x_{imjn}^{kv} - 1)$$
$$\forall i \in N\backslash\{0\}, \forall k \in K, \forall m \in D, \forall v \in V \quad (3.25)$$

$$A_{im}^{k} \leq a_{im}^{v} - M(\sum_{j \in N} \sum_{n \in D} x_{jnim}^{kv} - 1) \qquad \forall i \in N \backslash \{0\}, \forall k \in K, \forall m \in D, \forall v \in V \quad (3.26)$$

$$A_{im}^{k} \geq a_{im}^{v} + M(\sum_{j \in N} \sum_{n \in D} x_{jnim}^{kv} - 1) \qquad \forall i \in N \backslash \{0\}, \forall k \in K, \forall m \in D, \forall v \in V \quad (3.27)$$

The next group of constraints (3.28 - 3.35), ensure that there are no cycles between the two dimensions of the same node in any worker schedule or vehicle route.

$$x_{i0i0}^{kv} = 0 \qquad \forall i \in N, \forall k \in K, \forall v \in V \quad (3.28)$$
$$x_{i1i1}^{kv} = 0 \qquad \forall i \in N, \forall k \in K, \forall v \in V \quad (3.29)$$
$$x_{i0i1}^{kv} = 0 \qquad \forall i \in N, \forall k \in K, \forall v \in V \quad (3.30)$$
$$x_{i1i0}^{kv} = 0 \qquad \forall i \in N, \forall k \in K, \forall v \in V \quad (3.31)$$
$$z_{i0i0}^{v} = 0 \qquad \forall i \in N, \forall v \in V \quad (3.32)$$
$$z_{i0i1}^{v} = 0 \qquad \forall i \in N, \forall v \in V \quad (3.33)$$
$$z_{i1i0}^{v} = 0 \qquad \forall i \in N, \forall v \in V \quad (3.34)$$
$$z_{i1i1}^{v} = 0 \qquad \forall i \in N, \forall v \in V \quad (3.35)$$

Constraint (3.36) ensures that if a worker comes to node $j1$, he will leave node $j1$ and not $j0$.

$$\sum_{i,j \in N, i \neq j} \sum_{m \in D} \sum_{v \in V} x_{imj1}^{kv} \leq \sum_{i,j \in N, i \neq j} \sum_{m \in D} \sum_{v \in V} x_{j1im}^{kv} \qquad \forall k \in K \quad (3.36)$$

Finally, constraint (3.37) ensures that if a node is visited only once, then it must be visited in dimension 0.

$$\sum_{v\in V}\sum_{m\in D}\sum_{i\in N\{0\}} z^{v}_{j0im} \geq \sum_{v\in V}\sum_{m\in D}\sum_{i\in N\backslash\{0\}} x^{kv}_{imj1}$$
$$\forall j \in N\backslash\{0\}, k \in K \tag{3.37}$$

The model also includes binary constraints for the assignment, scheduling and routing variables and non negative constraints for the time variables, which are obvious and not listed here.

3.2.1 VRPTW as a subset of the VSWSP

As it can be seen from the formulation above, our model has some similarities to other VRP problems. One direct relationship can be established with the VRPTW, as both share identical basic features. We can then make the following statement:

Theorem 1. *Suppose the parameters for a VRPTW instance and those for a VSWSP instance are all the same except that VRPTW does not allow vehicle sharing. Then, any feasible solution for the VRPTW instance will also be a feasible solution for the VSWSP instance.*

To prove this, let us reduce the dimension of each node to one, i.e., set $D = \{0\}$, hence, simplifying the problem to have only one visit to each node. Let us also change Q_v in eq. (3.5) to 1. Thus, each vehicle can only accommodate one worker. It can be seen that with the changes the model becomes a formulation for the corresponding VRPTW. Notice that these changes tightens the constraints. Therefore, a feasible solution for the VRPTW model will still be feasible if these additional restrictions are removed, i.e., will be a feasible solution for VSWSP. Hence, it is shown that VRPTW a special case of VSWSP.

3.3 Preliminary test

Due to the complexity of the problem, only very small instances can be solved to optimality. In this section we present a small instance with 5 customer nodes, solve it to optimality and compare the solution to the non-sharing solution also solved to optimality. This small example has been created to show that sharing is possible and can be beneficial.

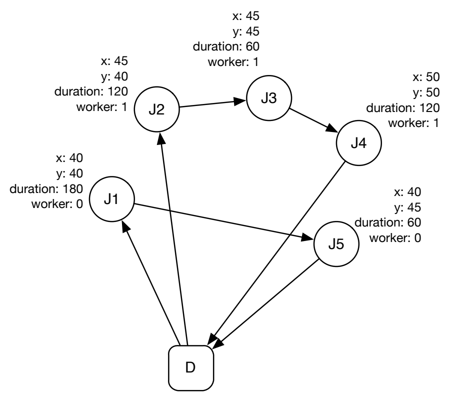

Figure 3-2: Solution for the routing of 5 nodes without sharing using Gurobi as solver

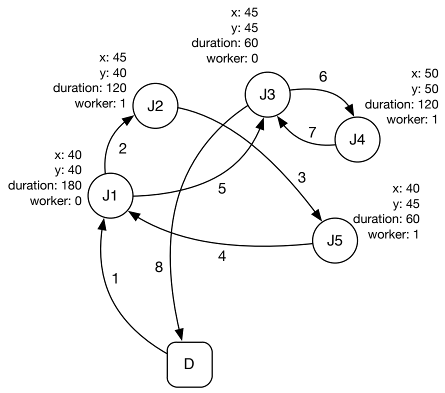

Figure 3-3: Solution for the routing of 5 nodes sharing using Gurobi as solver

The solutions for the versions without and with sharing are shown in Fig. 3-2 and 3-3. The coordinates of the nodes are listed next to the nodes, together with the job durations and the worker assignment for the customer nodes. The coordinates for the depot are $(0, 0)$. Note that the graphs are for illustration and the node positions on the graphs do not match their coordinates. But the distance and travel time parameters used are calculated using the coordinates. We have a total of 5 jobs that need to be assigned. As an assumption for our problem, all jobs have wide time windows, hence we will allow workers to be assigned throughout the whole day (0, 480). In Fig. 3-2 we can clearly see how jobs are assigned to two different workers with their respective vehicle routes. On the other hand, in Fig. 3-3, there are more trips among the customer nodes, but the number of trips to the depot is reduced. Next to the arcs connecting the nodes, are numbers which follow the order of trips. Therefore, first, the route will be from the depot to node 1, second from node 1 to node 2, and so on. Given this and the assignment of workers, it can be seen how worker 0, needs to wait until worker 1 has finished job 5 to pick him up and then drive to node 3.

Finally, for the non-sharing solution the total distance travelled is 262, compared to 157 by sharing vehicles. Furthermore, sharing also saves one vehicle.

3.4 Summary

In this chapter we have described the problem and its main features. Moreover, a novel formulation for the Sharing Vehicle Routing Problem with Time Windows has been introduced to define the problem mathematically and solve it to optimality. We show that VRPTW is a special case of our problem VSWSP, which implies that VSWSP is NP-hard because VRPTW is an NP-Hard problem. Finally, we show that our formulation works by presenting the results for a small example solved to optimality. Clearly, there is some potential as even with a small example we can see the benefits of sharing over non-sharing. Hence, the following chapters will tackle how to deal with bigger instances.

Chapter 4

A metaheuristic approach to the VSWSP

A mathematical model, while being able to solve problems to optimality, has its limitations on tackling big instances. Several tests have been done only solving problems up to 8 nodes which for practical purposes have little use. Due to the high complexity of our problem, we propose a heuristic approach capable of dealing with bigger instances of the problem.

The approach is a three-phase algorithm as shown in Fig.4-1. Phases 1 and 2 are designed for finding an initial feasible solution, while phase 3 is for improving the solution found in the previous phases. In phase 1 jobs are assigned to workers by solving a VRP with service time and maximum route time. In phase 2, we will create an initial feasible solution for, if possible, sharing vehicles. Phase 3 aims at improving the previous solution using different search procedures.

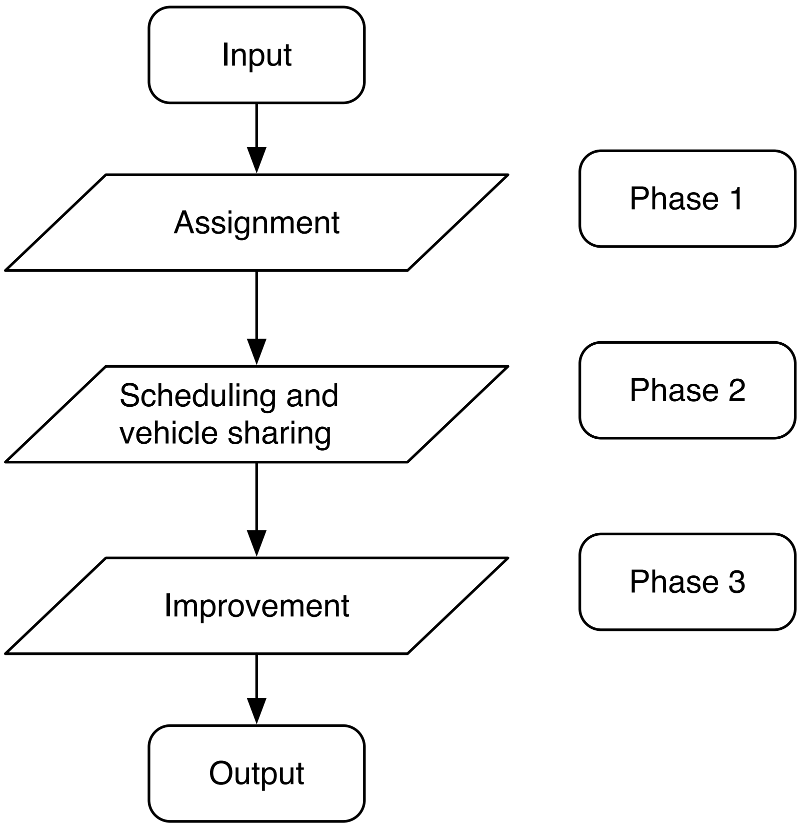

Figure 4-1: Algorithm summary

4.1 Description of the algorithm idea

4.1.1 Phase 1 - Assignment

The assignment process assigns jobs to workers and make sure each worker has sufficient time to finish the jobs assigned to him. It can be viewed as a VRP with maximum time constraints. In the literature this ia a common variation of the classical VRP with a maximum time side constraint. Laporte et al. (1984) tackle a similar variation of the problem known as DVRP which introduces a maximum distance constraint for the VRP. They introduce a mathematical formulation of the problem and two exact algorithms to solve the problem. Our problem is a variation of the problem shown in Laporte et al. (1984), but instead of only considering maximum distance, we also consider the service time at each node. The following mathematical formulation is presented to represent the problem.

$$min \sum_{i,j \in V, i \neq j} \sum_{k \in M} t_{ij} x_{ij}^k \tag{4.1}$$

$$\sum_{j \in V \setminus \{0\}} x_{0j}^k = \sum_{i \in V \setminus \{0\}} x_{i0}^k \qquad \forall k \in M \tag{4.2}$$

$$\sum_{k \in M} \sum_{i \in V} x_{ij}^k = 1 \qquad \forall j \in V \setminus \{0\} \tag{4.3}$$

$$\sum_{i \in V} x_{iu}^k = \sum_{j \in V} x_{uj}^k \qquad \forall u \in V \setminus \{0\}, \forall k \in M \tag{4.4}$$

$$\sum_{i,j \in V, i \neq j} x_{ij}^k (\tau_j + t_{ij}) \leq D_{max} \qquad \forall k \in M \tag{4.5}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i,j \in V, i \neq j, \forall k \in M \tag{4.6}$$

The objective (4.1) of the model is to minimise the total travel time required to serve all customers. Constraints (4.2) force the number of vehicles leaving the depot to be the same as the number of vehicles returning to it. Constraints (4.3) and (4.4) ensure that each customer is visited exactly once, and that if a vehicle visits a customer it must also depart from it. Constraints (4.5) limit the duration of the tour to be within the total time allowed. Finally, (4.6) are binary constraints for the variables used for the problem.

Though this phase-model does not consider vehicle sharing, it is already difficult to solve when problem size is large. Even if it can be solved, usually only one optimal solution is obtained. For the overall VSWSP, different phase-1 solutions might yield different sharing possibilities. Hence, we will use three different algorithms to solve the phase-1 problem, which will allow us to study how the quality of different initial solutions affect the final output. First, an insertion heuristic algorithm which uses a k-means approach to create clusters will be used for the assignment of workers to jobs. The second is a modified Clark and Wright algorithm with added restrictions which tackle the maximum time constraint. Finally, to generate a better initial (and non-sharing) solution, we modified the RCWS algorithm shown in Juan et al. (2011) to address the maximum time constraint. This last approach has been chosen as it gives very high quality solutions for the CVRP and it is easily convertible for our purpose. In later sections computational experiments are shown corroborating this.

To summarize, each worker will be assigned jobs considering the total time limit. The aim of this process is to find the best solution possible without sharing vehicles. Hence, if there is no possibility for sharing, we will go back to this best solution found for the classical variation of the VRP. In cases where sharing is possible, we will compare the solution with vehicle sharing with the solution with each worker using their own vehicle.

4.1.2 Phase 2 - Vehicle Sharing and Scheduling

This next step consists of scheduling the workers while trying to cover their traveling with vehicle routes. Jobs to be done by each worker are fixed based on the assignment results in phase 1 and used as input to phase 2. In phase 2, the order of doing jobs and the routes of vehicles are to be determined. One of the main difficulties of vehicle sharing is that we cannot know in advance which workers have more potential to share a vehicle. To tackle this, we introduce an algorithm with a decision tree structure which constructs a vehicle route by adding nodes one by one. At each node we decide what is the next best step to do. There are mainly three actions that can be done in each node depending on the previous action; Drop off a worker, Pick Up a worker, or Wait for the worker to finish.

- Drop: This action occurs when we decide to drop a worker to a node that he has been previously assigned. Returning to depot and dropping off all workers on board there is also considered as a Drop action.
- Pick Up: This is an action that the vehicle and the worker driving it pick up another worker at a node.
- Wait: This is an action that the vehicle and the workers on it, if any, wait at a node for a worker to finish his job there. A Wait action may be taken in two

situations: (1) If the current job is short enough such that it is better to wait for the worker to finish, the vehicle will wait in the node; (2) In the case that the worker being dropped at the node is the last/only worker in the vehicle at the time, then the empty vehicle is forced to stay in the node until worker finishes the job. Note that the wait action in situation (2) will become unnecessary if each vehicle has a delegated driver who is not a service worker.

As it can be seen in Fig 4-2, depending on which action it has just been taken, the vehicle has several choices for the next action. After a drop, all the possible actions can occur, while in the other cases, the choice is only between two actions. The idea to include this differentiation in the algorithm is that it gives more flexibility and efficiency in the construction phase of the solution. A wait will always occur in the same node, while pick up and drop will be in different nodes (grey colour). To choose the next step, we calculate and compare what is the best possible action that can happen. If arriving to next node (regardless if it is a drop or pick up) takes more time than waiting, we will wait for the worker to finish. Otherwise, we will either do a pick up or a drop off depending on the available nodes. Because it is difficult to know which workers can be merged beforehand, the proposed approach will create a matrix calculating the cost of the merge, where the cost is defined by the total distance travelled.

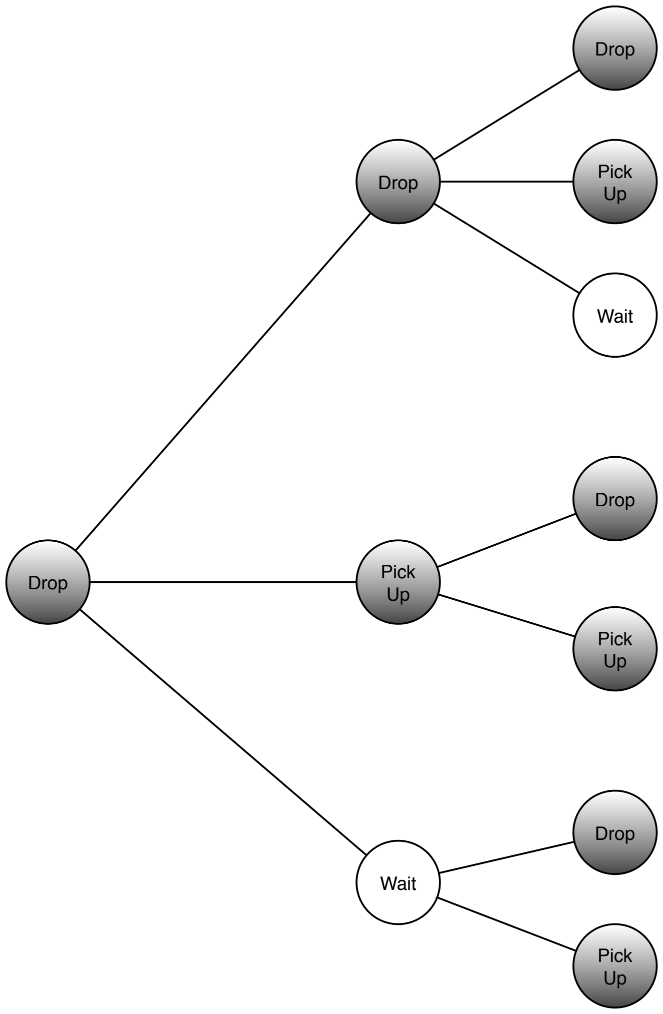

Figure 4-2: Decision tree for the sharing system, in grey are the actions which occur in different nodes, while wait in white will occur in the same node.

4.1.3 Phase 3 - Improvement

In the improvement phase, we apply three different methods to improve the solution given by Phases 1 & 2. The objective is to minimise the total distance of vehicles. Both clustered based and CWS are deterministic, while the RCWS has an iterative process which in each iteration a geometric distribution is used, therefore, not being considered deterministic.

The first method to improve the given initial solution is anILS introduced by Lourenço et al. (2003), which uses a 2-opt swap technique to generate neighbouring solutions in a Local Search (Fig. 4-4). Hence, given an initial solution, for every route created we will apply a swap (see Fig. 4-3) of two nodes in the route and calculate the new solution cost, the nodes do not need to be adjacent. If the cost is lower than the current solution we will update the current solution, and continue this procedure until we do not improve the solution after N iterations.

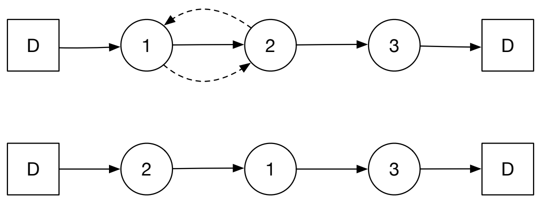

Figure 4-3: 2-opt swap

The second improvement technique is a more greedy approach, but consists of one parameter less than the ILS presented, as we will not use a swapping operator. We will destroy each worker's assignment solution by deassigning jobs to workers, and shuffle all the nodes positions inside the assignment list. Hence, instead of swapping two nodes as in the ILS approach, it will give us more variation in each one of the iterations. Again, we will repeat iteratively until n non improving number of steps. In our approach the stopping condition set using the number of iterations, but this can be easily changed to new conditions like CPU time, etc.

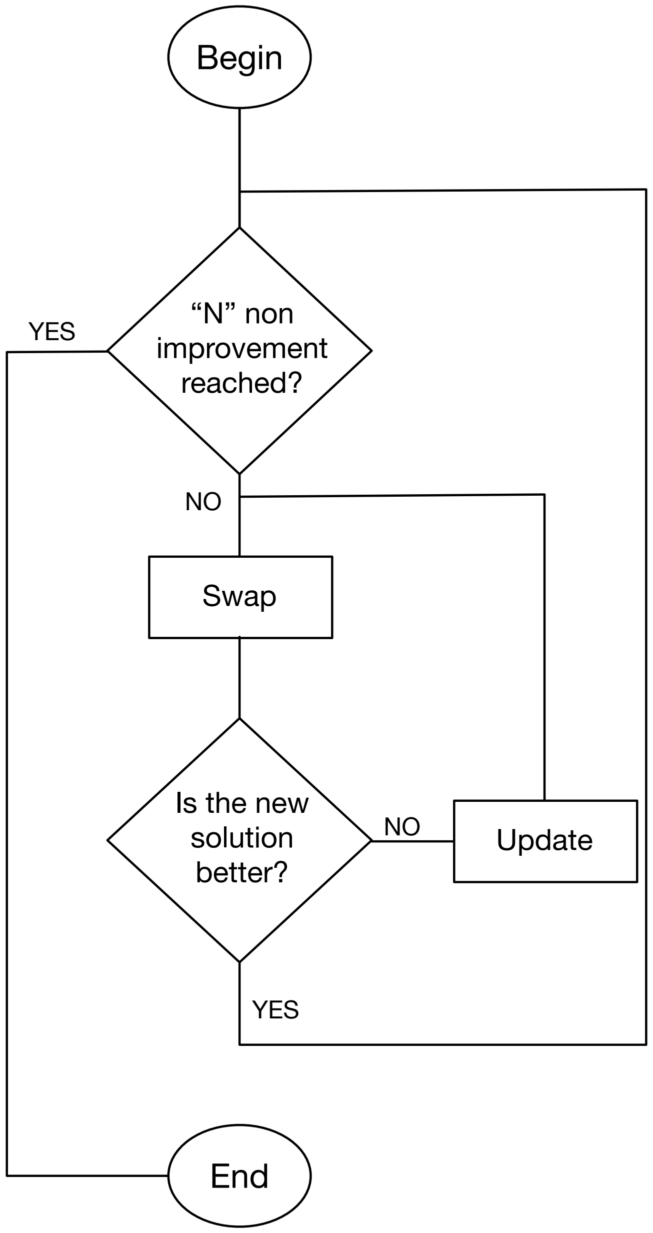

Figure 4-4: ILS for the vehicle sharing problem

Finally, a LNS algorithm is implemented. This type of algorithms follows an iterative process that destroys and repairs part of the solution. One of the main problems for LNS is the definition of a good neighbourhood. In our case, it can be defined in a very straightforward manner; using the matrix of mergings given by phase 2, we will destroy the tours selected for merging and then reconstruct them.

- Destroy: using the information given by phase 2 of possible mergings, we will choose the best merging and destroy the chosen tours. By destroying, we will deallocate all nodes from their respective workers. Hence, we will create a new unassigned list of all the nodes from the workers to merge.

- Repair: to repair the destroyed neighbourhood, we will use the list provided by the destroy step. Contrary to phase 2, in this case, nodes are not pre-assigned to any worker, hence, any worker can be assigned to any job from the list. Workers will be allowed to be scheduled to nodes that otherwise would be already assigned to another worker. This will allow us to have more possible combinations within the workers to merge.

An example of the destroy and repair method can be seen in Fig. 4-5. Two workers are selected for merging, once we have unassigned the jobs as part of the destroy step, all of the jobs can be scheduled to any of the workers. Hence, a better solution can be found which improves the cost of the merging.

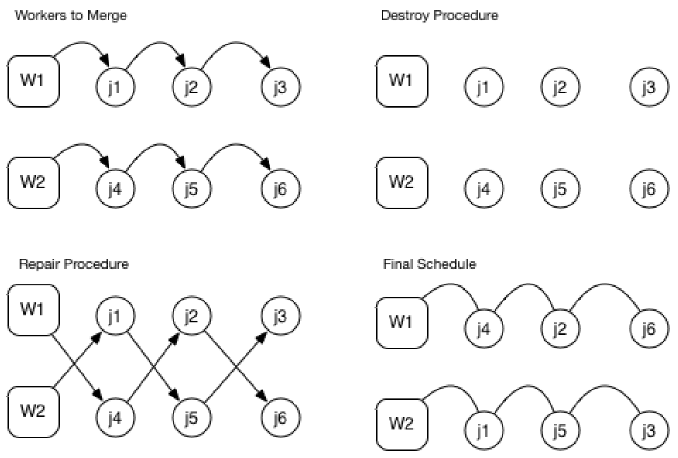

Figure 4-5: Repair and Destroy procedure

4.2 Implementation

In this section we present the implementation of different parts of the solution approach used to solve the VSWSP problem. For most part, the implementation is presented in the form of a pseudocode. Algorithm 1 is the pseudocode for the overall method.

Algorithm 1

```
1: input ← ReadInputs(inputs)
2: nonSharingSol ← GenerateAssignment(input)
3: sharingSol ← GenerateSchedule(nonSharingSol)
4: bestSol ← SharingSol
5: while sharingSol exists and termination criterion is not met do
6:     currentSol ← Improvement(bestSol)
7:     if cost(currentSol) < cost(bestSol) then
8:         bestSol ← currentSol
9:     end if
10: end while
11: return bestSol
```

The algorithm can be summarized as follows. It reads the inputs from a data set, then uses a routing algorithm to generate the assignment of jobs to workers taking into account time restrictions. After workers and nodes are assigned, it tries to merge them while creating their schedules and checking if the merging is feasible. Finally, once an initial solution is obtained, it will proceed to make improvement with the chosen method.

4.2.1 Assignment Pseudocode

For the problem in assignment phase, which can be simplified to a VRP or multiple TSP, we have chosen three different methods; CWS (Clarke and Wright (1964)), RCWS (Juan et al. (2011)) and a clustering algorithm.

The CWS which has been already introduced in Chapter 2 as part of the Literature Review is commonly used as an initial solution generator and it follows the following steps:

1. Assuming each node is visited directly from the depot as a separate route, calculate the savings of merging the routes for each pair of nodes (i, j) as $s(i, j) = d(D, i) + d(D, j) - d(i, j)$, where $d(i, j)$ is the distance between the two nodes, and D being the depot.

2. Rank the savings s(i, j) and list them in descending order of magnitude. This

creates the ”savings list”.

3. Beginning with the largest $s(i, j)$ on the list, check if an edge can be added to merge two routes. The edge (i, j) will be included in the merged route if the following constraints are satisfied:

   (a) Both nodes i and j are still in their individual routes, in which case a new combined route is created including both i and j.

   (b) If only one of the two nodes (i or j) has already been included in a combined route and the node is not interior to that route (a point is interior to a route if it is not adjacent to the depot D in the order of traversal of points), in which case the link (i, j) is added to merge the two routes.

   (c) Or, both i and j have already been included in two different combined routes and neither point is interior to its route, in which case the two routes are merged.

4. If the savings list is not empty, return to Step 3; otherwise, stop and result is taken as the solution to the VRP.

The next algorithm (Algorithm 2) used for the initial solution will be the RCWS introduced by Juan et al. (2011). It is a simple but yet really efficient algorithm that improves or matches best known solution for some of the benchmarks for the CVRP. It is important to note that in both of our problems with or without sharing, we do not use a capacity constraint, instead we use a time limit constraint, which resembles the structure of the classical capacity constraint, but instead of demand, each node has a service time and each vehicle’s capacity is the total route time.

Algorithm 2 RCWS(inputs)

```
1: savingList ← createSavingList(inputs)
2: cwSol ← createCWSolution(inputs, savingList)
3: while termination criterion is not met do
4:     vrpSol ← createRandomizedSolution(input, savingList, rng)
5:     vrpSol ← improveUsingCahce(vrpSol, inputs)
6:     if vrpSol outperforms cwSol then
7:         vrpSol ← improveUsingSplitting(vrpSol, inputs, rng)
8:         bestSols ← updateBestSolutionsList(vrpSol, bestSols)
9:     end if
10: end while
11: return bestSols;
```

As seen in Juan et al. (2011), the program works as follows. Firstly, the algorithm (Algorithm 2) constructs a first solution by using the traditional CWS which will be used as an upper bound to achieve better solutions during the following improvement steps. Then, an iterative process begins, first we will create a solution using a RCWS procedure which uses a geometric distribution on the savings list to choose the best mergings, rather than using a fixed order. This randomized solution is improved (if possible) by the cache procedure. The cache is constructed by saving the best known order to travel between the selected nodes of the route, if a better one is already saved we will return that, otherwise we will save the route order in the randomized solution as the best known. If the solution is better than our upper bound found by the CWS, we will try to improve it by applying a splitting procedure. This step tries to improve the solution by applying another iterative process using the RCWS and cache procedures to a subset of routes. The solutions will be saved to be returned at the end of the process. Following the same modification done in the basic CWS, we will change the capacity constraint so it is adapted to the maximum length of tour using service times.

Finally, the last algorithm (Alg. 3) used to assign job nodes (called cities) to workers is a self developed cluster based one. Firstly, we sort the nodes by distance to the depot (line 1). Then for each worker (called tour), we first assign the furthest available node to him. After that, we start assigning as many nodes to the worker as the maximum time capacity allows. To do so, we create the method getNextCity (see Algorithm 4), which will choose the next node to include in the tour. If the method

Algorithm 3 ClusteringAssignment(inputs)

```
 1: sortedCities ← SortingMethod(inputs)
 2: while there are still cities unrouted do
 3:     firstCity ← GetFirst(sortedCities)
 4:     tour ← Add(firstCity)
 5:     Remove(sortedCities, firstCity)
 6:     while there are still cities unrouted do
 7:         city ← GetNextCity(sortedCities)
 8:         if city ! = null  then
 9:             tour ← Add(city)
10:             Remove(sortedCities, city)
11:         else
12:             break
13:         end if
14:     end while
15:     sol ← AddTour(tour)
16: end while
17: return sol
```

returns null, the tour is finished. On the other hand, if it returns a node we will include it in the tour.

Algorithm 4 GetNextCity(sortedCities)

```
1: nextCity ← null
2: for city in sortedCities do
3:     if K-mean ! = null  then
4:         if time conditions are met  then
5:             nextCity = city
6:         end if
7:     end if
8: end for
9: return nextCity
```

Algorithm 4, if possible, choose the next node to include in the current tour. It iterates over the available nodes, and choose the one that meets the time constraints (in our case availability to work up to 480min) and by applying a similar algorithm to the K-means. Our variation selects points that are within an acceptable distance of the centroid to be included. The original k-means algorithm performs n clusters in which each observation belongs to the cluster with the nearest mean. Our variation selects points that are withina n acceptable distance of the centroid.

4.2.2 Scheduling and Sharing Pseudocode

Phase 2 is the key part of this thesis solution approach. While presenting the following pseudocodes, we will explain in depth how we explore the possibility of vehicle sharing.

Algorithm 5 GenerateSchedule(nonSharingSol)

```
1: unroutedTours ← ExtractTours(nonSharingSol)
2: matrixPossibleMergings ← GeneratePossibleMergings(unroutedTours)
3: while there are still unroutedtours do
4:     toursToSchedule ← selectTours(unroutedTours,matrixPossibleMergings)
5:     while there are still toursToSchedule do
6:         scheduledTours ← Scheduling(toursToSchedule)
7:         if scheduledTours != null then
8:             RemoveTours(unroutedTours,scheduledTours)
9:             sol ← AddTour(scheduledTours)
10:        else
11:            RemoveTour(toursToSchedule)
12:        end if
13:    end while
14: end while
15: return sol
```

Algorithm 5 is the overall pseudocode for creating the schedule using the solution given by the assignment algorithm. Firstly, we will extract the individual worker tours created in Algorithm 3 (line 1). Then, inspired by the idea behind CWS algorithm, we create a cost matrix of merging each pair of tours (line 2). Then starting from the best possible merging according to the matrix, we will merge as many workers as possible to generate a schedule with vehicle sharing (lines 5 to 14). Finally, we will return the final solution.

Algorithm 6 createMatrix

```
1: matrixPossibleMergings $\leftarrow$ null
2: toursToSchedule $\leftarrow$ null
3: for i in unroutedWorkers do
4:     for j > i in unroutedWorkers do
5:         toursToSchedule ADD(unroutedWorkers(i))
6:         toursToSchedule ADD(unroutedWorkers(j))
7:         mergedTours $\leftarrow$ SCHEDULING(toursToSchedule)
8:         if mergedtours != null then
9:             matrix(i,j) $\leftarrow$ COSTOFMERGE(vr)
10:        end if
11:    end for
12: end for
13: return matrix
```

To create the matrix we propose the following procedure (Algorithm 6). We initialize the matrix to have null values in line 1. For each pair of workers i and j where woker j needs to be greater than i, we call the scheduling method to check if the merging is possible (line 7). If the merge is possible, we calculate the cost and add it to the matrix, otherwise, the value of that merging continues being a null.

The scheduling method is shown in Algorithm 7. Any given number of tours to schedule will be the input for this step. CitiesToSchedule will store all the possible cities, which can be merged from toursToSchedule (line 1). DropList is used to store the nodes that can be potentially visited next (line 2) while PickUpList is used to store the nodes that can be potentially visited next to pick up a worker who was previously dropped (line 3). We first initialise these two lists: DropList contains the first nodes that were assigned to the workers being merged (all these workers get on the vehicle when it leaves the depot at the start), while PickUpList is empty. To start the new schedule, we will select one of the cities inside the toursToSchedule and start with a Drop. After the first Drop, we start an iterative process (line 6), in which we decide what is the best next action to take. Depending on the actions we have just taken, we can decide the action to do next. Hence, if the DropList still contains nodes (line 7), we may do a drop or a pick up next.

If the worker in the vehicle is the last worker, then the next pick up city will be itself, as it needs to wait to finish (line 16). Finally, if the DropList is empty, we can no longer do Drops, so the actions we can do will be limited to PickUp (line 19). If we manage to create a feasible merged tour, we will return it; otherwise, we will return the false value.

Algorithm 7 Scheduling(toursToSchedule)

```
1: CitiesToSchedule ← ExtractCities(toursToSchedule)
2: DropList ← ReadDrops(CitiesToSchedule)
3: PickUpList ← null
4: cityD ← GetCityD(DropList)
5: tour ← Drop(cityD)
6: while DropList > 0 || PickUpList > 0 || unfeasible == true do
7:     if DropList > 0 then
8:         cityD ← GetCityD(DropList)
9:         if Drop condition is met then
10:            tour ← Drop(cityD)
11:        else
12:            cityP ← GetCityPU(PickUpList)
13:            tour ← PickUp(cityPU)
14:        end if
15:    end if
16:    if LastWorker == true then
17:        LastCity ← GetCityPU(PickUpList)
18:        tour ← PickUp(LastCity)
19:    else
20:        cityP ← GetCityPU(PickUpList)
21:        tour ← PickUp(cityPU)
22:    end if
23:    if tour == null then
24:        break;
25:    end if
26: end while
27: return tour
```

The process that handles the Drop action (shown as Algorithm 8) consists of updating the variables related to the vehicle and workers. If the prior action was a drop (line 1) we will calculate the departure of the vehicle from the node of the prior drop (line 2). Also, we will compute the arrival time of the vehicle and the workers on it to the new node, and update the pick up and drop lists (lines 3 - 5). We will remove the dropped worker from the vehicle so we can keep this information for the capacity constraint of the vehicle (lines 6 and 12).

Algorithm 8 Drop(CityD,tour,LastState,DropList,PickUpList)

```
 1: if LastState == Drop then
 2:     updateLastNodeDepartureTime(tour);
 3:     updateTimes(CityD,tour);
 4:     UpdateSequences(CityD, tour);
 5:     AddCityPickUpL(CityD,PickUpList);
 6:     RemoveCityDropL(CityD,DropList);
 7:     LastState = Drop;
 8: else
 9:     updateTimes(CityD);
10:     UpdateSequences(CityD, tour);
11:     AddCityPickUpL(CityD,PickUpList);
12:     RemoveCityDropL(CityD,DropList);
13:     LastState = Drop;
14: end if
```

For the pick up process (Algorithm 9), it will follow the same procedure as the drop one. Firstly, we need to identify if we need to stay at the same node the Drop was made waiting for the worker to finish (i.e., the last action was a drop at the same node) so we can accordingly update all the information. If we have to wait (line 1), we will update the times and sequences accordingly to that action (lines 2 - 3). On the other hand, if we need to do a pick up at another node , we need to identify which was the last action the algorithm has done. When updating the times again, it is different depending on whether we come from a Drop or a Pick Up (line 11). In both cases we will update the pick up list and the drop list. To update the drop list (lines 6 and 15), if there is another job assigned to the worker given the assignemtn from Phase 1 we will add it to the list following the longest jobs first rule.

Algorithm 9 PickUp(CityPU,tour,LastState,DropList,PickUpList)

```
1: if LastVisit == CityP then
2:     updateTimes(CityP,tour);
3:     UpdateSequences(CityP, tour);
4:     RemoveCityPickUpL(CityP,PickUpList);
5:     if CityP has next city then
6:         AddNextCityToDropL(CityP, DropList)
7:     end if
8:     LastState = PickUp;
9: end if
10: if tourContains(cityP) then
11:     updateTimes(CityP,tour,LastState);
12:     UpdateSequences(CityP, tour);
13:     RemoveCityPickUpL(CityP, PickUpList);
14:     if CityP has next city then
15:         AddNextCityToDropL(CityP, DropList)
16:     end if
17:     LastState = PickUp;
18: end if
```

4.2.3 Improvement Pseudocode

In Phase 3 the aim is to improve the cost of the solution found. As explained before, three methods have been implemented: an ILS, a shuffling procedure, and a LNS.

Algorithm 10 ILS

```
1: bestSol ← GenerateScheduling(input)
2: bestCost ← CalculateCost(bestSol)
3: while nonImprovement < $Nsteps$ do
4:     sol ← Perturbation(initialSol)
5:     cost ← CalculateCost(sol)
6:     if cost < bestCost then
7:         UpdateSolutions(sol,cost,bestSol,bestCost)
8:     else
9:         nonImprovement ← nonImprovement + 1
10:     end if
11: end while
```

Algorithm 10 shows the pseudocode for the iterative process of swapping two nodes within the same route, from the routes provided by the scheduling phase, which differ from the routes given in Phase 1. Once the perturbation has been produced, we calculate the cost of this new solution. If the new solution is better than the best solution saved, we will update both the best solution and its cost. Otherwise, if it does not improve, we will count it as non-improvement. When the iterative process reaches a certain number of non-improvements, we will stop and return the best solution found so far.

Algorithm 11 randOpt

```
1: bestCost[] ← INITIALIZEBESTCOSTS(sol)
2: while stopping criterion is met do
3:     for vr in scheduleSol do
4:         for w in vr do
5:             SHUFFLECITIES(w)
6:         end for
7:         newVr ← SCHEDULING(vr)
8:         if vr != null then
9:             if COSTOFMERGE(newVr) < bestCost[vr] then
10:                sol ← SETNEWVR(vr,newVr)
11:                bestCost[vr] ← COSTOFMERGE(newVr)
12:            end if
13:        end if
14:    end for
15: end while
16: return sol
```

The next method is a randomized local search (Algorithm 11). Here we will shuffle the jobs assigned to each worker, e.g. $worker_1$ is assigned to do jobs $1, 3, 5$, the scheduling procedure will try to schedule them sequentially, by shuffling them, now $worker_1$ may do the jobs in the order of $3, 1, 5$. Iteratively, we will do a number of shuffles to check different solutions, the number of nodes assigned to each worker is not very high due to the time restriction, hence we can try every possible combination in a short computational time. If the cost of new schedule of the assignment is lower than the previous cost, we will accept the new schedule.

Algorithm 12 LNS

```
1: initialSol ← GenerateScheduling(input)
2: bestCost ← CalculateCost(bestSol)
3: bestSol ← initialSol
4: while stopping criterion is met do
5:     Destroy(initialSol)
6:     sol ← Repair(initialSol)
7:     cost ← CalculateCost(sol)
8:     if cost < bestCost then
9:         UpdateSolutions(sol,cost,bestSol,bestCost)
10:    end if
11: end while
12: return bestSol
```

The final procedure to be used is a LNS (Algorithm 12). As explained before, we will destroy mergings from the initial solution, allowing all the nodes to be re-scheduled again. The repair method consists of a variation of the scheduling algorithm, but changing the assignment of tasks. Previously all jobs had already been assigned to workers. Now, instead of sequentially adding the previously assigned tasks to the drop list after every pick up, we will assign the tasks to the workers while doing the drop or the pick up.

4.3 Results

4.3.1 Creation of datasets

To assess the impact of vehicle sharing during working hours we have created our own data sets as no available benchmarks exist for this type of problem. We create three different types of instances to assess the suitability for sharing vehicles as it might not be possible in all the cases. Hence, we first create a base dataset and then change some parameters to create scenarios ranging from all-day jobs to shorter service times. Also, the geographic distribution of the nodes is considered in two settings: clustered or randomly spread following a Gaussian (Normal) distribution. This will allow us to model a city based scenario (more nodes concentrated in the centre, with other nodes spreading further as they are located away from the centre). As introduced by Gabaix and Ioannides (2004), a lognormal distribution best describes a city's dispersion. For practical reasons we will use a Normal distribution which can always be transformed into a Lognormal distribution. Graphical examples of both distribution settings can be seen in the following images. In the first case we have a non-cluster 100 node scenario (Fig.4-6), and the second cases is a clustered scenario with 10 nodes in each cluster as shown in Fig. 4-7.

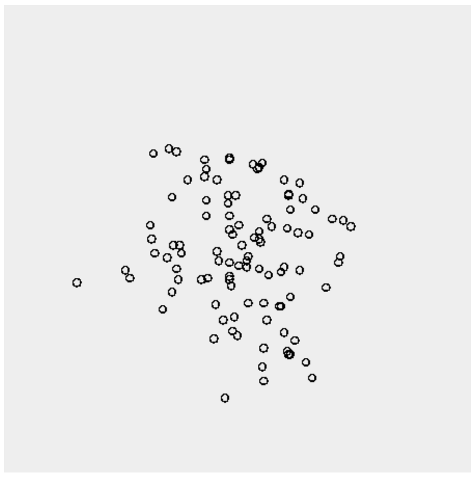
Figure 4-6: Scenario with 100 nodes without clustering

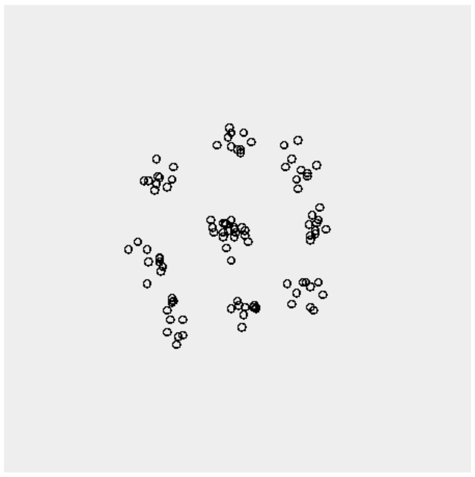
Figure 4-7: Scenario with 100 nodes with clusters of 10 nodes each

The parameters used for each of the instances can be seen in Table 4.1 with the number of nodes ranging from 25 up to 150. Clustered scenarios are defined as $cX\_$ where c stands for "clustered" and X is the number of nodes. Otherwise, non-clustered scenarios are named as $ncX\_$ where nc stands for "non-clustered" and X again is the number of nodes. The second and third columns, present the number of nodes and the number of clusters for each input file. For the clustered scenarios, each cluster is set to be in a specific distance to the depot. This avoids the clusters generated to be too close to the depot. The fourth column shows this distance for the scenarios. Non clustered ones, cities are randomly placed. Finally, the fifth column shows the standard deviation of the Gaussian distribution used for generating the nodes. For the clustered scenarios, a first point is generated as the centre of the cluster and points are created surrounding that point given the standard deviation. For the non clustered scenarios the standard deviation is used for the Gaussian distribution to place points given that the depot is the centre.

Table 4.1: Parameters used for the creation of scenarios

Scenarios	Number of nodes	Number of clusters	Distance between clusters and depot	Standard Deviation
c25_1	25	2	50	7
c25_2	25	4	50	7
c25_3	25	5	50	7
c50_1	50	5	50	7
c50_2	50	8	50	7
c50_3	50	10	50	7
c100_1	100	10	40	7
c100_2	100	15	40	7
c150_1	150	10	40	7
c150_2	150	15	40	7
nc25	25	-	-	45
nc50	50	-	-	45
nc100	100	-	-	45
nc150	150	-	-	45

Besides the parameters shown in the table, there are other features that we will consider for the input data sets.

- Service Times of Jobs: We have identified three different service durations; Short, Medium, and Long jobs. Short jobs have a duration of 60 minutes, Medium jobs 120 minutes and Long jobs 180. To diversify each scenario, we have included a small variation of ± 20 minutes.
- All day jobs: Some jobs in maintenance or cleaning services require the whole day. To replicate this, we have selected some random jobs, and increased their time so that one worker is only able to do one such job. Depending on the distance of the jobs to the depot, the time for an all-day jobs ranges between 300 and 420 minutes. This type of scenarios will be labelled with an L in their title.
- Short jobs: Another set of inputs have been created using shorter times for jobs. It follows the same classification introduced in the first point above, but times are considerably shorter. Short jobs now are between 20 and 30 minutes, medium between 40 and 50 minutes while long ones are between 60 and 70 minutes. This type of scenarios will be labelled with an S in their name.

4.3.2 Experimental Results

Experimental results are provided for all the scenarios we have created. Results for non-sharing and sharing solutions are given for each one of the algorithms used in Phase 1. To compare results with state of the art algorithms for vehicle routing, we use the total distance of the vehicles as objective function. Hence, we summarize the results for CWS, RCWS and the clustering algorithm, with and without sharing. Cluster based and the CWS algorithms are used, as one of the aims of this work is to observe if better sharing (i.e fewer vehicles) benefits from lower quality solutions (i.e. increasing the number of workers). To show the performance of each algorithm and distinguish between their solution qualities, results are shown solving the small instances, for non-sharing (Table 4.2). As it can be seen the RCWS performs constantly better tha both CWS and clustered based by using less vehicles and less distance. While CWS performs better than clustered based.

Table 4.2: Comaprisons between the three initial algorithms used, indicating the number of vehicles and the total distance.

	Clustered based		CWS		RCWS	
	n_vehicles	totalcost	n_vehicles	totalcost	n_vehicles	totalcost
c25_1	11	1575.61	11	1576.93	11	1566
c25_2	11	1328.93	11	1382.81	10	1295.76
c25_3	12	1600.6	11	1534.64	11	1476.34
c50_1	20	2669.47	20	2529.78	20	2536.81
c50_2	20	2137.39	20	2115.55	19	2090.76
c50_3	20	2172.35	21	2168.86	19	2020.12
c100_1	38	4430.33	39	4376.1	38	4329.08
c100_2	34	3264.03	34	3097.94	33	3019.53
c150_1	51	4827.87	52	4768.55	51	4702.32
c150_2	59	7277.5	60	7225.45	58	7014.11
nc25	11	1665.23	10	1511.78	10	1564.02
nc50	21	2605	20	2510.72	19	2479.77
nc100	36	5220.09	37	5177.85	35	5065.87
nc150	57	7005.58	57	6787.09	56	6711.09

Because of the complexity of our problem, the comparison with optimal solutions is not possible. Also, because no benchmarks tackling this problem have been found, we compare the non-sharing solution to the sharing solution in the number of vehicles and the total distance. Hence, we need a good algorithm for the vehicle routing problem with maximum time constraints which gives enough good solutions to compare our approach. The randomized version of CWS in Juan et al. (2011) using cache and splitting techniques, works consistently very well for the CVRP as seen on their work. Therefore, to have valid comparisons we have solved the Christofides et al. (1979) instances that tackle CVRP with service times and maximum route time, to show that our modified algorithm performs well giving non-sharing solutions.

Table 4.3: Solutions for the Christofides et al. (1979) maximum time for routes instances

	BKS	OBS	GAP
CMT06	555.43	555.43	0.00
CMT07	909.68	916.75	0.78
CMT08	865.95	866.60	0.08
CMT09	1162.55	1188.81	2.26
CMT10	1395.85	1435.79	2.86
CMT13	1541.14	1547.53	0.41
CMT14	866.37	866.37	0.00
Average			0.91

As it can be seen in Table 4.3 the algorithm performance is highly competitive as the average GAP (relative gap, %) for Christofides et al. (1979) Best Known Solution (BKS) is less than 1%. Moreover we manage to match the BKS in two instances. This result allows us to have a fair comparison between sharing and not sharing solution for instances up to 200 nodes.

We have tested our approach with three different studies showing how the duration of jobs and geographical distribution of jobs affect the sharing capability of a possible transportation system. To do so, we have defined three set of experiments: on instances with normal duration, long duration and short duration of jobs. The tables of results show the number of vehicles (the number of workers will be the same as vehicles) for the non-sharing initial solution, with its total cost given by the total distance. Secondly, we will try to schedule shared routes for workers in each of the scenarios. The number of vehicles used and Our Best Solution (OBS) for each instance are shown together with the relative gap between the non-sharing and sharing solutions. The number of workers in the sharing solution is the same as the non sharing as we want to improve on that solution. If a sharing solution cannot be found it is shown as "−". For more in-depth results, the reader is directed to the appendices (Appendix A) showing solutions for each one of the methods. All problems are solved using a Toshiba Portege machine with 8G of RAM, Intel Core i5 and 2.30GHz. The CPU time given to solve all instances is 300s which is the time limit used to solve Christofides et al. (1979) benchmarks. Table of results showing all the number of vehicles and the total cost can be seen in Tables A10 and A11 in the Appendix.

Clustering Algorithm

The solutions given by the clustering algorithm for initial solutions can be seen in Tables 4.4, 4.5 and 4.6. In the base scenarios (Table 4.4), almost all the clustered scenarios can have sharing, and because solutions are of worst quality, two of the non-clustered scenarios can also have sharing. Notice that non-clustered scenarios tend to be more difficult to share. Again, for all day jobs (Table 4.5), all the scenarios are possible to have sharing, while in short service times, only one scenario is possible.

Table 4.4: Base Scenario using clustering algorithm

	Non-sharing		Sharing		
	n_vehicles	totalcost	n_vehicles	OBS	GAP
c25_1	11	1575.61	9	1399.23	-11.19
c25_2	11	1328.93	9	1234.51	-7.10
c25_3	12	1600.60	10	1376.44	-14.00
c50_1	20	2669.47	18	2475.67	-7.26
c50_2	20	2137.39	19	2125.28	-0.57
c50_3	20	2172.35	19	1905.45	-12.29
c100_1	38	4430.33	36	4430.88	0.01
c100_2	34	3264.03	32	3228.95	-1.07
c150_1	51	4827.87	-	-	-
c150_2	59	7277.50	57	7277.15	0.00
nc25	11	1665.23	10	1636.61	-1.72
nc50	21	2605.00	20	2558.55	-1.78
nc100	36	5220.09	-	-	-
nc150	57	7005.58	-	-	-

Table 4.5: All day Jobs using clustering algorithm

	Non-sharing		Sharing		
	n_vehicles	totalcost	n_vehicles	OBS	GAP
c25L_1	15	2129.20	9	1398.02	-34.34
c25L_2	12	1592.18	8	1234.50	-22.46
c25L_3	14	1842.70	10	1582.79	-14.10
c50L_1	26	3280.55	17	2316.73	-29.38
c50L_2	25	2640.14	19	2296.57	-13.01
c50L_3	24	2491.99	17	2104.65	-15.54
c100L_1	46	5242.00	35	4595.81	-12.33
c100L_2	40	3708.77	34	3353.06	-9.59
c150L_1	62	5581.59	48	4847.84	-13.15
c150L_2	70	8318.66	57	7324.63	-11.95
nc25L	13	1800.61	10	1750.19	-2.80
nc50L	24	2870.31	19	2516.95	-12.31
nc100L	45	5779.35	36	5342.65	-7.56
nc150L	69	8019.25	57	6992.99	-12.80

Table 4.6: Short jobs using clustering algorithm

	Non-sharing		Sharing		
	n_vehicles	totalcost	n_vehicles	OBS	GAP
c25S_1	4	688.95	-	-	-
c25S_2	4	639.23	-	-	-
c25S_3	5	781.67	-	-	-
c50S_1	8	1269.61	-	-	-
c50S_2	8	1120.89	-	-	-
c50S_3	8	1111.33	-	-	-
c100S_1	15	2246.81	-	-	-
c100S_2	14	1820.2	-	-	-
c150S_1	20	2569.39	-	-	-
c150S_2	23	3416.40	-	-	-
nc25S	6	1281.62	5	1177.61	-8.12
nc50S	9	1636.97	-	-	-
nc100S	16	3086.69	-	-	-
nc150S	23	3817.21	-	-	-

Clarke and Wright

The results obtained using the CWS algorithm for the base scenarios can be seen in Table 4.7. We can observe that in all the clustered scenarios it is possible to share vehicles under this circumstance, and to reduce the total number of vehicles used while maintaining the same number of workers as given by the CWS algorithm. Furthermore, it is worth noting that while reducing the number of vehicles we still manage to reduce the total distance travelled in most of the scenarios. An explanation for this can be given with the fact that by reducing the vehicle number we substract the connection between the depot and the cluster, while only increasing the distance to revisit some intra-cluster nodes. On the other hand, if we consider the non-clustered scenarios, only one out of four is possible to share, reducing one vehicle.

Table 4.7: Base scenarios using Clarke and Wright algorithm

	Non-sharing		Sharing		
	n_vehicles	totalcost	n_vehicles	OBS	GAP
c25_1	11	1576.93	10	1352.13	-14.27
c25_2	11	1382.81	9	1213.88	-12.71
c25_3	11	1534.64	10	1423.36	-6.95
c50_1	20	2529.78	18	2336.44	-7.24
c50_2	20	2115.55	19	2115.06	-0.02
c50_3	21	2168.86	19	2059.36	-5.04
c100_1	39	4376.10	37	4268.05	-2.44
c100_2	34	3097.94	33	3096.39	-0.05
c150_1	52	4768.55	51	4779.03	0.22
c150_2	60	7225.45	58	6984.73	-3.31
nc25	10	1511.78	-	-	-
nc50	20	2510.72	-	-	-
nc100	37	5177.85	36	5173.15	-0.09
nc150	57	6787.09	-	-	-

Table 4.8: All day Jobs using the Clarke and Wright Algorithm

	Non-sharing		Sharing		
	n_vehicles	totalcost	n_vehicles	OBS	GAP
c25L_1	14	1983.44	10	1518.60	-21.83
c25L_2	13	1622.65	7	1073.39	-34.50
c25L_3	14	1760.33	9	1248.63	-27.77
c50L_1	25	3197.39	17	2368.80	-25.26
c50L_2	25	2561.96	17	2078.59	-18.31
c50L_3	25	2450.48	18	2141.44	-12.40
c100L_1	47	5253.75	37	4356.16	-17.12
c100L_2	41	3647.72	32	3220.20	-11.53
c150L_1	63	5447.52	48	4784.02	-11.89
c150L_2	71	8323.27	55	7039.28	-15.44
nc25L	12	1650.09	10	1607.96	-2.34
nc50L	24	2775.68	20	2578.22	-6.88
nc100L	45	5683.17	35	5208.51	-8.21
nc150L	69	7778.81	53	6697.07	-13.49

Table 4.9: Short Jobs using the Clarke and Wright Algorithm

	Non-sharing		Sharing		
	n_vehicles	totalcost	n_vehicles	OBS	GAP
c25S_1	4	638.59	-	-	-
c25S_2	4	601.02	-	-	-
c25S_3	4	717.05	-	-	-
c50S_1	8	1231.62	-	-	-
c50S_2	8	966.10	7	971.73	0.50
c50S_3	8	968.12	-	-	-
c100S_1	15	2155.71	-	-	-
c100S_2	13	1548.10	-	-	-
c150S_1	20	2209.78	19	2216.93	0.28
c150S_2	22	3112.95	-	-	-
nc25S	5	1079.65	-	-	-
nc50S	8	1386.87	-	-	-
nc100S	15	2623.45	-	-	-
nc150S	22	3351.74	-	-	-

Table 4.8 shows the results of applying the CWS algorithm to scenarios with all day jobs. In this case, we obtain even better results while sharing vehicles. In all the instances created we can greatly reduce the number of vehicles, in some cases reducing up to 16 vehicles. Moreover, in all cases we also manage to reduce the total distance of the vehicles, including the non-clustered ones.

Finally, Table 4.9, presents the results found for the cases with short service times for the jobs. We found that by reducing the duration of the jobs, the possibility of sharing is massively reduced, hence, the results presented in this table. Using this approach only two possible scenarios manage to share their vehicles.

Randomized Clarke and Wright

Table 4.10 shows the results for the RCWS applied to the base scenarios. Again, in this case, most of the clustered scenarios are able to share vehicles. But because this version of the CWS gives relatively much better results reducing the number of initial workers and vehicles, some instances cannot reduce any vehicles further by sharing. Again, non-clustered scenarios perform much worse, and none can be shared.

Table 4.10: Base Scenario using Randomized Clarke and Wright

	Non-sharing		Sharing		
	n_vehicles	totalcost	n_vehicles	OBS	GAP
c25_1	11	1566.00	9	1395.54	-10.89
c25_2	10	1295.76	9	1203.84	-7.09
c25_3	11	1476.34	10	1411.80	-4.37
c50_1	20	2536.81	18	2318.85	-8.59
c50_2	19	2090.76	-	-	-
c50_3	19	2020.12	-	-	-
c100_1	38	4329.08	37	4273.32	-1.29
c100_2	33	3019.53	32	2984.76	-1.15
c150_1	51	4702.32	50	4691.82	-0.22
c150_2	58	7014.11	-	-	-
nc25	10	1564.02	-	-	-
nc50	19	2479.77	-	-	-
nc100	35	5065.87	-	-	-
nc150	56	6711.09	-	-	-

The results of the all day jobs scenarios (seen in Table 4.11), show again an improvement with vehicle sharing. In all scenarios we improve both the number of vehicles and the distance of the best solution. It is important to note that while in some solutions the number of vehicles is worse than the ones in the classical CWS, the RCWS finds initial solutions with lower number of workers.

Table 4.11: All day jobs using Randomized Clarke and Wright

	Non-sharing		Sharing		
	n_vehicles	totalcost	n_vehicles	OBS	GAP
c25L_1	14	1983.44	10	1518.60	-23.44
c25L_2	12	1546.68	8	1200.59	-22.38
c25L_3	14	1760.32	9	1248.63	-29.07
c50L_1	24	3101.38	16	2295.72	-25.98
c50L_2	25	2553.23	17	2081.53	-18.47
c50L_3	24	2345.11	18	2027.97	-13.52
c100L_1	46	5071.27	34	4186.73	-17.44
c100L_2	40	3588.11	34	3282.80	-8.51
c150L_1	62	5346.90	48	4685.99	-12.36
c150L_2	71	8145.83	54	6737.75	-17.29
nc25L	12	1650.09	10	1607.96	-2.55
nc50L	23	2759.32	20	2566.27	-7.00
nc100L	44	5525.67	36	5136.76	-7.04
nc150L	68	7654.07	54	6708.44	-12.35

Similarly, while considering short jobs scenarios (Table 4.12), vehicle sharing seems to be not possible in most of the cases. Because the RCWS finds higher quality solutions, there is only one case in which sharing is an option.

Table 4.12: Short Jobs using Randomized Clarke and Wright

	Non-sharing		Sharing		
	n_vehicles	totalcost	n_vehicles	OBS	GAP
c25S_1	4	627.70	-	-	-
c25S_2	4	592.36	-	-	-
c25S_3	5	717.05	-	-	-
c50S_1	8	1215.97	-	-	-
c50S_2	8	963.48	7	-	-
c50S_3	8	958.99	7	952.59	-0.67
c100S_1	15	2003.36	-	-	-
c100S_2	13	1521.98	-	-	-
c150S_1	19	2152.96	-	-	-
c150S_2	19	2152.96	-	-	-
nc25S	5	1028.70	-	-	-
nc50S	8	1314.16	-	-	-
nc100S	15	2545.79	-	-	-
nc150S	22	3242.65	-	-	-

4.4 Non-sharing Optimal Solutions vs Sharing

Due to the complexity of the problem, finding optimal solutions sharing vehicles is an arduous task for bigger instances. Therefore, using the formulation introduced in Phase 1 we can obtain some optimal solution for the smallest instances we test. We are only able to solve the base and all day jobs scenarios as when applied to the short durations the problem complexity increases and it needs more than 12h to solve (time when we stopped the solver).

Table 4.13 shows the comparison between the optimal solutions found applying the above mentioned model using Gurobi Optimizer and the best solutions found by our algorithm both sharing and no sharing. We present the optimal solution both in total distance and number of vehicles (Opt Of and Opt n_vehicles), Our Best Solution Not Sharing (OBSNS) and the relative gap between the non-sharing solutions found by the exact method and by our algorithm (GAP), and the number of vehicles and the best solution with sharing allowed (n_vehicles and Best Solution Sharing (BSS)). It is important to highlight that our algorithm has been run for 300s.

Table 4.13: Comparison between optimal solutions without sharing and our best sharing solutions..

	Opt Of	Opt n_vehicles	OBSNS	GAP	n_vehicles	BSS
c25_1	1552	11	1566.00	0.90	9	1399.99
c25_2	1280	10	1295.76	1.23	9	1205.63
c25_3	1465	11	1476.34	0.77	10	1411.80
c25L_1	1967	14	1983.44	0.83	10	1518.60
c25L_2	1531	12	1546.68	1.02	8	1200.59
c25L_3	1746	14	1760.32	0.82	9	1248.63
Average				0.92		

As seen in the table, the average GAP between the non-sharing solutions continue the same trend as with the previously introduced comparison with Christofides et al. (1979) benchmarks, with the average being less than 1%. Optimal solutions are found between 1h and 8h of running time while OBSNS use only 300s. For the purpose of this thesis, the most interesting fact is that by using our best algorithm to find initial solutions, we are able to further improve the optimal non-sharing solutions by sharing vehicles using less running time. Hence, in all the instances which we can share vehicles, we can improve the traditional solution with a short amount of time.

4.4.1 Effect of the Duration of Jobs on Shareability

One of the concepts we want to introduce is shareability. For our problem at hand, we understand shareability as how much a transportation system is able to share vehicles. Because there is no specific background defining this concept we have run a series of experiments which allow us to illustrate this concept. We have defined two variables to explain this phenomenon; number of scenarios which allow sharing and total number of vehicles shared. From previous results we can see that the duration of jobs significantly affects the shareability of scenarios. Hence, we have tested our findings by running 5 simulation runs with 10 randomly created instances of 50 nodes. We continue to see differences between the results of short, medium and long jobs. and the details of each run can be seen in table 4.14.

Table 4.14: Parameters used for job durations for the simulation runs

	Short	Medium	Long
Run1	20	50	80
Run2	50	80	110
Run3	80	110	140
Run4	110	140	170
Run5	140	170	200

Figures 4-8 and 4-9 show the results found both in clustered and non-clustered scenarios. These findings support the idea that duration of jobs highly affects shareability, as in both cases almost all of the scenarios with the longest times allow more sharing. While in the clustered scenarios it follows a more steady curve, the non-clustered variations in Runs 2, 3 and 4 can be explained by the manner in which the instances are created. With a Normal distribution some of the points might be further away from the depot, hence, it would allow the assigned route to be shared.

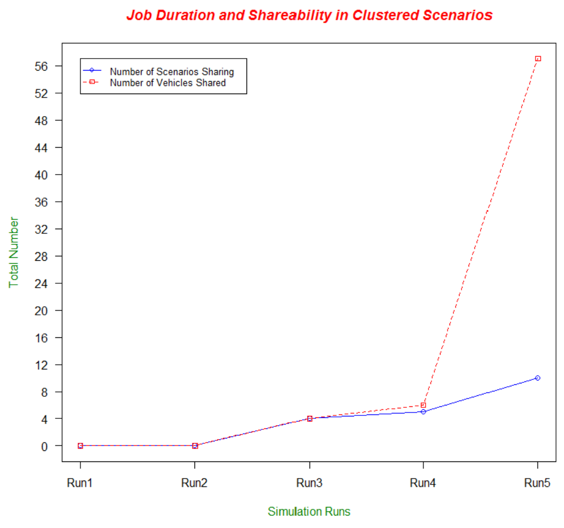

Figure 4-8: Results for the clustered simulations

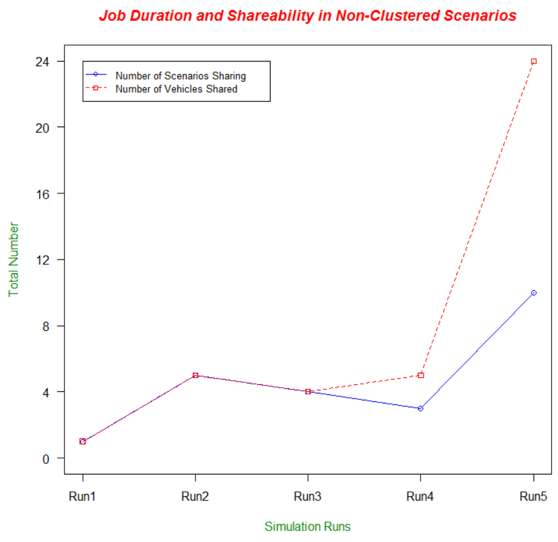

Figure 4-9: Results for the non clustered simulations

4.5 Summary and Conclusions

As seen in the results, numbers of vehicles and workers are the same in the initial solution (as each worker uses one vehicle). After applying the sharing procedure, we maintain the same number of workers, but reduce the number of vehicles. Focusing only on the number of vehicles and workers, it can be seen that we can maintain the same standard of service even if we share vehicles. Workers have their tasks assigned and in most cases there is some slack, that can be used to revisit/drive other workers to different locations.

We have tested different quality assignment solutions for Phase 1 to check if there was any difference in the sharing procedure. An interesting trade off that can be seen, is the fact that in some cases, solutions that give worst assignments for workers, give better reductions of vehicles while sharing. Examples of this are seen in some scenarios such as c100L_sd7_40_2, or nc150L_sd45, which result in fewer vehicles but more workers. This means that even using lower quality initial solutions, it might have a greater impact in reducing the number of vehicles. On the other hand, having lower quality solutions such as in the cluster based algorithm, gives us much worse solutions regardless. Hence, there is a limit to which using a worse solution could benefit the sharing capability. In general, using always the RCWS algorithms gives the best sharing solutions both in distance and number of vehicles, but in very specific instances better solutions can be found by using more workers in the initial solution.

On the other hand, we have decided to also compare the different results of the objective function, in our case total distance of the vehicles. We have chosen this, as it is a straight comparison that can be used comparing sharing and non-sharing solution, and see if there is any trade off between reducing the number of vehicles and an increase/decrease in the total distance. As it can be seen, if we take the best solution found for sharing, we find that in almost all the cases while reducing the number of vehicles we manage to reduce the total distance of the vehicles with all three algorithms, hence we can conclude that regardless of how good the non-sharing initial solution might be, there is always a possibility to further improve this, by merging workers into shared vehicles.

To conclude, we have tested the possibility of vehicle sharing and its impacts on the final solution, using a variety of scenarios with different features. Moreover, we have shown how they differ considering different initial quality solutions, and its effect on the number of vehicles and total distance. Clearly, sharing vehicles has proven to be possible in scenarios in which the service times of the jobs are longer than the travel time between jobs, and the longer the service times the better chances to share, even if there are short jobs.

Furthermore, clustered scenarios seem more prone to sharing, as the assignment of jobs is less compact, i.e. there is more slack time in clustered assignment than non clustered. Finally, we always want to use the best initial assignment of jobs to workers, as in general it always gives better solutions. In some cases, we manage to reduce the number of vehicles if we compare CWS to the RCWS, but the overall performance shows that using a better initial solution gives better sharing solutions, and if we do not manage to share, the initial solution can be used as the schedule of workers without sharing vehicles.

Chapter 5

The usage of dedicated drivers for the Vehicle Sharing: is it a good approach?

In this chapter we discuss the possibility to use dedicated drivers to facilitate the workers sharing of vehicles. Ride sharing using drivers has been previously studied as a new trend to increase occupancy in taxi and similar services such as Uber. The main objective is to optimally match drivers with riders using a variety of different objective functions. Agatz et al. (2012) present a review of different ride sharing problems in the literature. Contrary to our approach, in Agatz et al. (2012) they focus on dynamic problems, thus, there is no pre-established schedule for the riders.

Recent research has focused on what is known as the trip sharing problem (which essentially is the same concept as ride sharing) in the home healthcare system. Fikar and Hirsch (2015) and Fikar et al. (2016) introduce a variation of this problem where home-care nurses and doctors are dispatched to complete a set of tasks, a feature in their problem is that the home-care staff are required to go back to the depot. To solve their problem they introduce a discrete event simulation and a math-heuristic approach which enables them to find solutions for a set of self-created scenarios.

The new features and assumptions of the problem we study in this chapter can be summarized as follows:

- A vehicle can travel between two points without having a worker driving it. To adapt our problem to a driver based sharing method, we need to let the vehicle free to go between two points without being bounded by a worker. By doing that, even if there is no workers inside the vehicle it can be driven to another point.

- We assume that workers know beforehand the tasks they need to perform and all the necessary tools can be stored in a portable toolbox. Nowadays, in big metropolitan cities, companies have already started to adopt the carry-on toolbox for field workers so they can use public transport to move from point to point.

- The number of drivers is unbounded. In this problem we do not consider a maximum number of drivers used as we try to give the best solution we can find by using any number of drivers. The number of drivers will most likely be fewer than the number of vehicles, as the workers can still drive their vehicles if possible.

5.1 Drivers in our mathematical model

In chapter 3 we have introduced a mathematical model representing the sharing problem using the workers themselves as drivers. In this chapter we aim to study what are the consequences of introducing dedicated drivers for the transport of workers. One of the purposes of this thesis is to show how flexible our approache becomes when either switching between problems or adding new restrictions. To include drivers in our model the only change needed is to remove equation 5.1.

$$\sum_{k}^{K} x_{imjnkv} \geq z_{jnimv} \quad \forall v \in V, \forall i, j \in N, i \neq j, \forall m, n \in D \tag{5.1}$$

This equation states that a vehicle needs to have a worker driving it, hence, in the equation if a vehicle goes from node $im \rightarrow jn$ it has to be assigned to at least one worker k. By removing this constraint we let the vehicle free to move between nodes without being bounded by a worker.

5.2 Drivers in our algorithm

Changes to the heuristic approach are as minimal as in the mathematical model. Due to the flexibility of our approach the same idea as explained in the previous chapter can be easily adapted to use drivers. The main difference from the logic previously introduced is that waiting becomes unnecessary after the last worker is dropped. Having a driver who is not a worker means there is always someone available to drive the vehicle.

In algorithm 13 we can observe how these changes are made. As it can be seen we have removed the condition where we check if the last drop was the last worker in the vehicle. Hence, we will either check if there are drops or pickups to do. Obviously, if waiting for the worker to finish is the best option, *cityPU* will be the same node as the last drop, which still allows vehicles to wait for workers.

Algorithm 13 schedulingDriver

```
1: DropList ← READDROPS(CitiesToSchedule)
2: PickUpList ← null
3: tour ← DROP(city)
4: while DropList > 0 || PickUpList > 0 || unfeasible == true do
5:     if DropList > 0 then
6:         if Drop condition is met then
7:             tour ← DROP(cityD)
8:         else
9:             tour ← PICKUP(cityPU)
10:        end if
11:    else
12:        tour ← PICKUP(cityPU)
13:    end if
14:    if tour == null then
15:        break;
16:    end if
17: end while
18: return tour
```

One of the key features and novelties of this approach is that both the mathematical model and the heuristic approach are easily adaptable to allow sharing by using drivers or without using drivers. One of the aims was to have a flexible algorithm while maintaining the essence of inserting the best possible candidate in the scheduling sequence, at the end of the algorithm we can identify which tours are using or not using drivers as not all the routes will need a driver assigned. Therefore, some tours might share a vehicle but because of their structure there is no need to use a driver. Thus, this approach allows us to generate the best possible sharing with or without drivers.

5.3 Heuristic Results

Similar to the previous chapter, we present the set of results for each initial solution generator. The tables presented show the number of vehicles without sharing (which is the same as the number of workers used), the total cost of the solution, compared to the number of drivers and vehicles in the sharing solution, and the total costs obtained using the three different optimisation approaches. The number of drivers is shown as not all workers will be able to share and they will have to drive their own vehicle.

Ideally what we are looking for by using drivers is a significant reduction in the number of vehicles. On the other hand, this may result in an increase on the number of employees as additional drivers will be needed. Hence, we are looking for possible trade-offs in which the number of vehicles reduced is high enough to compensate for the number of drivers. The complete results are shown in Appendix B.

5.3.1 Cluster results

Results from applying our clustered based algorithm to get an initial solution can be seen in Tables 5.1 - 5.3. It is interesting to observe that the number of scenarios which allow sharing has increased compared to the results shown in the previous chapter. For the base scenario, it is not until we arrive to the 50 node scenarios we actually find that using drivers starts making a difference. In general, from that point we achieve a higher reduction in vehicle number compared to the number of drivers. In some very specific cases, due to the topology of the scenario and because it was randomly created, the total distance traveled while sharing might be greater than the one not sharing (nc50 in Table 5.1).

For all day and short tasks (Tables 5.2 & 5.3), the results indicate a similar trend which was previously seen. In the Table for all day tasks (5.2), results can be immediately seen from 25 nodes, as in all the cases the number of initial vehicles needed is highly reduced by using drivers. On the other hand, short time task scenarios still give poor results in terms of sharing.

Table 5.1: Results for the base scenarios using the clustering initial solution and driver sharing

	No Sharing		Sharing		
	n_vehicles	Cost	n_vehicles	n_drivers	OBS
c25_1	11	1575.61	4	4	858.77
c25_2	10	1328.93	5	4	941.58
c25_3	12	1600.60	5	5	1107.02
c50_1	20	2669.47	11	5	1947.46
c50_2	20	2137.39	12	6	1712.57
c50_3	20	2172.35	16	2	1978.65
c100_1	38	4430.33	29	6	4216.86
c100_2	34	3264.03	23	8	3015.62
c150_1	51	4827.87	26	15	3983.89
c150_2	59	7277.50	45	10	6626.14
nc25	11	1665.23	9	2	1638.25
nc50	21	2605.00	14	6	2648.94
nc100	36	5220.09	30	6	5022.13
nc150	57	7005.58	38	14	6735.85

Table 5.2: Results for scenarios with all day duration tasks using the clustering initial solution and driver sharing

	No Sharing		Sharing		
	n_vehicles	Cost	n_vehicles	n_drivers	OBS
c25L_1	15	2129.20	6	4	1070.30
c25L_2	12	1592.18	6	3	1058.37
c25L_3	14	1842.70	8	2	1442.87
c50L_1	26	3280.55	12	5	1970.7
c50L_2	25	2640.14	12	7	1905.95
c50L_3	24	2491.99	14	5	2210.90
c100L_1	46	5242.00	29	11	4127.65
c100L_2	40	3708.77	26	9	3317.43
c150L_1	62	5581.59	34	16	4487.21
c150L_2	70	8318.66	41	17	6433.06
nc25L	13	1800.61	9	2	1819.28
nc50L	24	2870.31	15	5	2582.31
nc100L	45	5779.35	31	10	5288.14
nc150L	69	8019.25	42	16	6812.66

Table 5.3: Results for scenarios with short duration tasks using the clustering initial solution and driver sharing

	No Sharing		Sharing		
	n_vehicles	Cost	n_vehicles	n_drivers	OBS
c25S_1	4	688.95	-	-	-
c25S_2	4	639.23	-	-	-
c25S_3	5	781.67	-	-	-
c50S_1	8	1269.61	-	-	-
c50S_2	8	1120.89	-	-	-
c50S_3	8	1111.33	-	-	-
c100S_1	15	2246.81	-	-	-
c100S_2	14	1820.20	13	1	1819.68
c150S_1	20	2569.39	19	1	2644.31
c150S_2	23	3416.40	-	-	-
nc25S	6	1281.62	5	0	1177.61
nc50S	9	1636.97	-	-	-
nc100S	16	3086.69	-	-	-
nc150S	23	3817.21	-	-	-

5.3.2 Clarke and Wright results

After running the experiments using the CWS algorithm, we can see some initial differences between these and the previous results using the clustering algorithm. Again, in general it tends to work better for base scenarios with 50 or more nodes (Table 5.4). In general, we improve all the results for the total distance, but again we need to focus our attention on the number of drivers compared to the number of vehicles.

Table 5.5 shows the results for all day tasks. The number of vehicles again can be massively reduced by including drivers while also decreasing the total distance. More interesting result can be seen in Table 5.6 for the short tasks as there are more scenarios which allow sharing compared to the clustering approach.

Table 5.4: Results for the base scenarios using the CWS initial solution and driver sharing

	No Sharing		Sharing		
	n_vehicles	Cost	n_vehicles	n_drivers	OBS
c25_1	11	1576.93	4	4	905.61
c25_2	11	1382.81	4	4	824.19
c25_3	11	1534.64	8	3	1248.66
c50_1	20	2529.78	10	7	1727.85
c50_2	20	2115.55	12	6	1658.08
c50_3	21	2168.86	14	5	1953.89
c100_1	39	4376.10	21	11	3550.14
c100_2	34	3097.94	22	9	3016.56
c150_1	52	4768.55	27	15	4057.16
c150_2	60	7225.45	33	20	5661.16
nc25	10	1511.78	9	1	1528.90
nc50	20	2510.72	16	4	2458.37
nc100	37	5177.85	29	6	5113.85
nc150	57	6787.09	35	17	6561.40

Table 5.5: Results for scenarios with all day tasks using the CWS initial solution and driver sharing

	No Sharing		Sharing		
	n_vehicles	Cost	n_vehicles	n_drivers	OBS
c25L_1	14	1983.44	7	3	1215.23
c25L_2	13	1622.65	5	3	921.66
c25L_3	14	1760.33	7	3	1102.03
c50L_1	25	3197.39	13	5	2050.19
c50L_2	25	2561.96	15	6	1902.09
c50L_3	25	2450.48	15	3	2152.99
c100L_1	47	5253.75	26	9	4133.01
c100L_2	41	3647.72	25	10	3188.80
c150L_1	63	5447.52	32	15	4197.73
c150L_2	71	8323.27	38	17	6179.26
nc25L	12	1650.09	8	3	1668.98
nc50L	24	2775.68	16	4	2642.49
nc100L	45	5683.17	30	9	5402.78
nc150L	69	7778.81	40	15	6568.37

Table 5.6: Results for scenarios with short tasks using the CWS initial solution and driver sharing

	No Sharing		Sharing		
	n_vehicles	Cost	n_vehicles	n_drivers	OBS
c25S_1	4	638.59	3	1	640.22
c25S_2	4	601.02	-	-	-
c25S_3	5	717.05	-	-	-
c50S_1	8	1231.62	7	1	1207.48
c50S_2	8	966.10	7	1	990.45
c50S_3	8	968.12	7	1	1008.06
c100S_1	15	2155.71	-	-	-
c100S_2	13	1548.10	12	1	1628.63
c150S_1	20	2209.78	18	1	2304.43
c150S_2	22	3112.95	-	-	-
nc25S	5	1079.65	-	-	-
nc50S	8	1386.87	-	-	-
nc100S	15	2623.45	-	-	-
nc150S	22	3351.74	-	-	-

5.3.3 Randomized Clarke and Wright Results

Finally, the results presented in Tables 5.7 to 5.9 are given by using the modified RCWS. Again, we want to observe, with this different initial solution generators, the effect of having dedicated drivers on the sharing and number of drivers used. As shown in Table 5.7, in some instances, using CWS works better while having a worse initial solutions. Specific cases showing this are $c25\_1$ for 25 nodes, or $nc100$. Although in general, if we achieve better results initially we end up having better results.

Again by looking at the results of all day tasks (Table 5.8) we have a higher number of workers sharing vehicles. Some unexpected results appear in the short duration tasks (Table 5.9). Contrary to the previous chapter, better initial solutions lead to less shareability as tasks are more compact within workers schedules.

Table 5.7: Results for the base scenarios using the RCWS initial solution and driver sharing

	No Sharing		Sharing		
	n_vehicles	Cost	n_vehicles	n_drivers	OBS
c25_1	11	1566.75	4	4	909.87
c25_2	10	1295.77	4	4	824.88
c25_3	11	1533.58	8	3	1253.50
c50_1	20	2502.67	9	7	1668.71
c50_2	19	2074.79	14	4	1857.69
c50_3	19	2009.01	14	3	1900.21
c100_1	38	4277.76	24	10	3958.97
c100_2	33	2984.00	21	9	2944.97
c150_1	51	4678.99	29	13	3954.24
c150_2	58	7000.28	35	15	5675.97
nc25	10	1511.78	9	1	1528.90
nc50	19	2451.85	17	2	2513.4
nc100	35	4991.33	30	5	4898.72
nc150	56	6640.86	39	15	6680.63

Table 5.8: Results for scenarios with all day tasks using the RCWS initial solution and driver sharing

	No Sharing		Sharing		
	n_vehicles	Cost	n_vehicles	n_drivers	OBS
c25L_1	14	1983.44	7	3	1260.66
c25L_2	12	1546.69	5	4	934.32
c25L_3	14	1760.33	7	3	1143.84
c50L_1	24	3101.39	12	7	1998.32
c50L_2	25	2553.23	15	6	2018.26
c50L_3	24	2345.86	14	5	1929.86
c100L_1	46	5071.28	28	10	4103.07
c100L_2	40	3588.11	23	12	3197.20
c150L_1	62	5346.91	32	16	4174.81
c150L_2	71	8145.83	35	19	5699.29
nc25L	12	1650.09	9	2	1699.08
nc50L	23	2759.32	16	5	2622.64
nc100L	44	5525.67	30	9	5258.47
nc150L	68	7654.07	43	15	6695.39

Table 5.9: Results for scenarios with short tasks using the RCWS initial solution and driver sharing

	No Sharing		Sharing		
	n_vehicles	Cost	n_vehicles	n_drivers	OBS
c25S_1	4	627.70	3	1	591.41
c25S_2	4	592.36	-	-	-
c25S_3	5	717.05	-	-	-
c50S_1	8	1215.97	7	1	1202.66
c50S_2	7	963.13	-	-	-
c50S_3	8	959.00	7	1	981.61
c100S_1	15	2003.37	14	1	2094.73
c100S_2	13	1521.99	12	1	1584.40
c150S_1	19	2152.97	18	0	2275.95
c150S_2	22	2968.71	21	0	3064.16
nc25S	5	1028.70	-	-	-
nc50S	8	1314.17	-	-	-
nc100S	15	2545.80	14	1	2720.55
nc150S	22	3242.66	21	1	3425.25

5.4 Trade-off between Drivers and Vehicles

One of the main consequences of using drivers to allow vehicle sharing is an increase in the number of employees in the final solution. After running all the experiments, we have better insight of using drivers which has allowed us to further study this effect in every instance of our problem. While there is not a right or wrong approach for this trade off it is worth to study its effects.

For each one of the tests, we have plotted our results in Fig. 5-1 - 5-3. Plotted are the increment in the number of workers, given by the number of drivers, while on the other half of the graph there is the reduction in number of vehicles. For example, in Fig. 5-1 for $c25\_1$, we have 11 vehicles for the non-sharing solution with 11 workers. Once we apply sharing, there is a reduction of 7 vehicles, while the total number of employees increases by 4. Ideally, what we would like to achieve is to considerably reduce the number of vehicles compared to the increment of drivers. Good examples of this can be seen in the all day job duration scenarios (Fig. 5-2), for which there is a good trade off and also for larger instances there seems to be a greater improvement. All the remaining results and plots for clustered based and CWS can be seen in Appendix B.

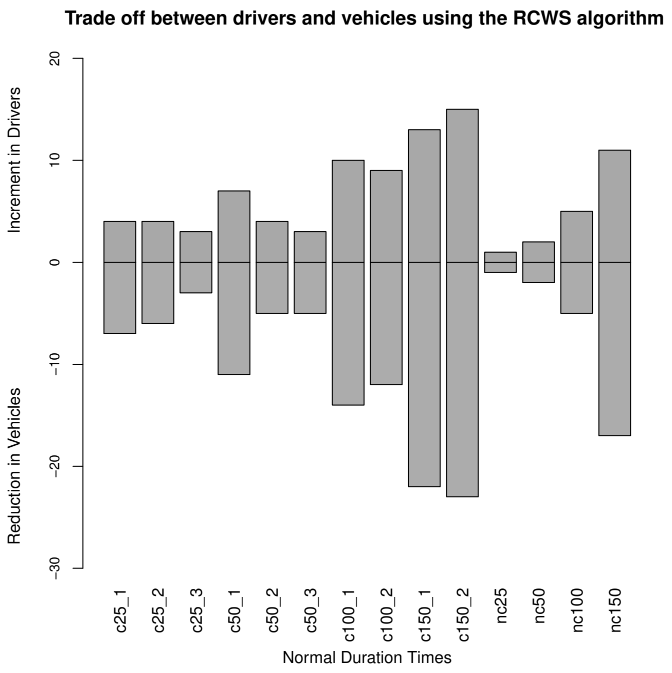

Figure 5-1: Trade off between drivers and number of vehicles on base scenarios using the RCWS algorithm as initial solution

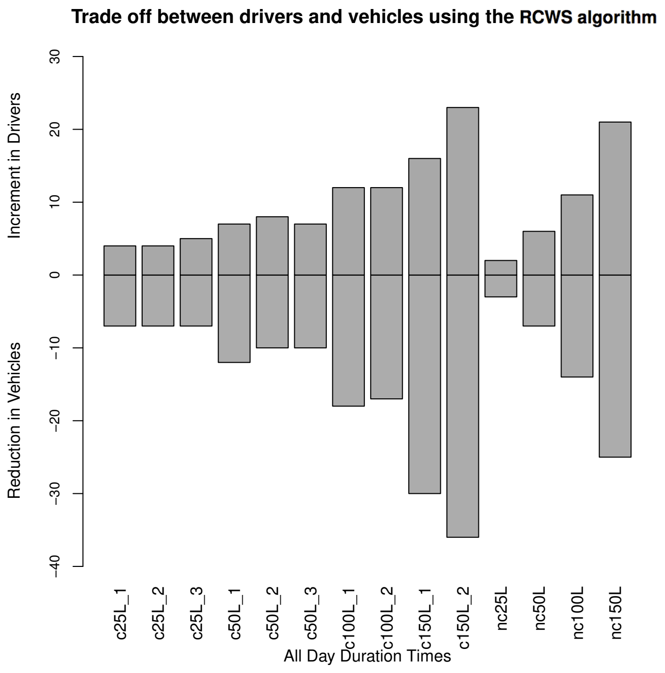

Figure 5-2: Trade off between drivers and number of vehicles on all day jobs using the RCWS algorithm as initial solution

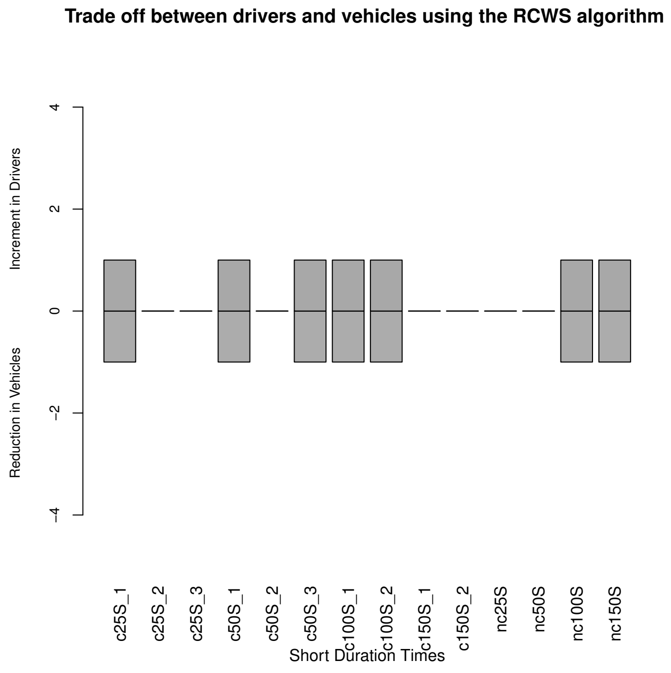

Figure 5-3: Trade off between drivers and number of vehicles on short jobs using the RCWS algorithm as initial solution

5.5 Comparison between using or not using drivers

Further to our previous discussion where we compare our results to a non-sharing solution, this section will focus on analysing the results obtained from sharing with and without drivers. The objective of our problem is to use as few vehicles as possible while maintaining or reducing (if possible) the total distance of vehicles. In the case of using drivers, there is a third component that affects the final outcome of the solutions; the drivers. Summarized, we introduce two graphs Fig. 5-4 and Fig. 5-5, presenting the results obtained from this chapter and the comparison with the previous one. We have used the RCWS results for both graphs as it gives the best non-sharing solutions, we present the normal duration time scenarios, while the subsequent graphs can be found in the Appendx B.

Figures 5-4 shows the comparison between the number of vehicles used in the non-sharing solution, and the sharing solutions with and without drivers. On the graph, if at any point, the value is 0, it means that we could not find a sharing solution (i.e. c_50_2, c_50_3, etc.). As it can be seen, by using drivers the algorithm always finds better solutions vehicle wise. Moreover, it clearly shows that in difficult cases such as the non-clustered scenarios, where sharing was more difficult, by using drivers it does allow to reduce vehicles.

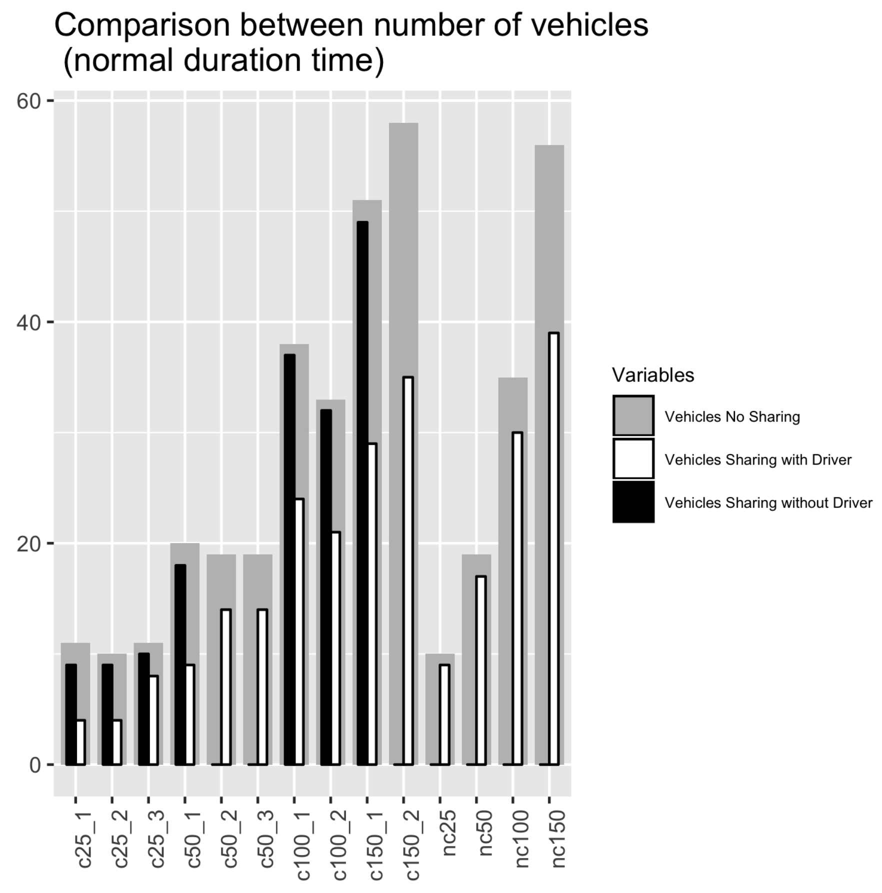

Figure 5-4: Comparison of the total number of vehicles between non-sharing, sharing without drivers, and sharing with drivers, for normal duration time jobs.

On the other hand, Figure 5-5 presents the results in terms of the number of employees (both workers and drivers). We represent the initial number of workers needed to undergo all the jobs in white and the number of drivers in grey. In general, by allowing the use of drivers, though not all the routes will use drivers, in all cases there is at least one driver, if there is a possibility to share.

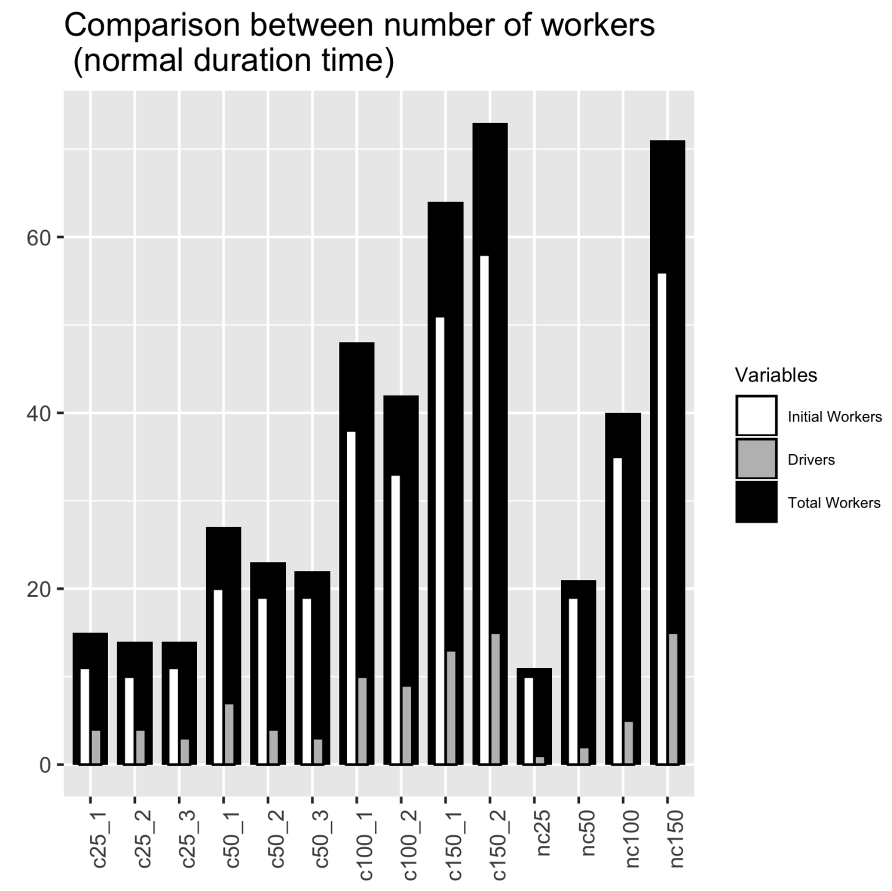

Figure 5-5: Comparison of the total number of employees (drivers plus workers assigned to jobs), the number of workers without sharing and the number of drivers, for normal duration time jobs

5.6 Summary and Conclusions

The usage of dedicated drivers for sharing purposes has just recently started to receive attention within the research community. In this chapter we aimed to show how adding drivers would affect the vehicle sharing possibilities. By using the same scenarios it allows us to do a complete overview of different sharing techniques and have a framework in which we can compare our results.

We have shown that by using drivers, we may be able to further reduce the number of vehicles and the traveling distances, but at the cost of increasing the number of employees. This trade-off should be considered from a managerial perspective depending on their objective, as in some cases the number of drivers is overly high.

Also, if exact costs are known and available for each one of the terms and its weights for the objective function, we could consider the cost of drivers, workers and vehicles.

Another consideration is the number of drivers a company is willing to use. Let us say that a company aims to employ a maximum of two drivers. By applying the procedure presented above, we just have to limit the number of vehicles merged to be up to two, thus, it would allow us to present results bounded by the resources given by the company.

Chapter 6

Using short-distance walking to reduce vehicle trips

Including the additional feature of walking between two points for routing and scheduling problems is a recently developing concept within research. It commonly appear in two forms: as an inherited problem constraint where a person has to walk between two points or as a method to simplify and solve larger instances of problems. There are a number of services which use a combination of driving and walking, such as parcel deliveries and post mail. In this chapter we aim to study the effects of clustering jobs within a certain walking distance on the size of the problem and on sharing capability. Some of the questions we want to answer can be summarized as:

- By clustering jobs, each worker will be assigned to longer non-separated jobs. As seen in previous chapters, this generally leads to better sharing results.
- Will the total distance be less than that in the case without applying the walking-distance clustering? Is this going to require use of more or less workers? And how many drivers will be used?

6.1 Background

We first review some studies that consider walking in routing and scheduling problems which aim to create vehicle routes with the possibility to walk between nodes.

One of the first works to specifically deal with short-distance walking between nodes is the so-called Multi-vehicle Covering Tour Problem (m-CTP) presented by Hachicha et al. (2000). Given two sets of locations, set V are locations that must be visited by some vehicles while set W are locations which are not in a vehicle route but within an acceptable walking distance to a location on the vehicle route, the aim of the problem is to construct vehicle routes through set V with the objective of minimizing the length (distance) and subject to some side constraints, such that every location in set W is in a reasonable distance of the route. The authors state that this problem can be seen in areas such as healthcare monitoring services or vehicle going through some villages and that every location not visited by the vehicle must be in walking distance.

Another common problem dealing with walking distance constraints can be seen in Park and Kim (2010), which presents a review of the school bus routing problem. In some instances of the problem, some authors take into consideration the walking distance between students and the possible bus stops for the creation of clusters.

In Eiró et al. (2011) the authors present a minibus service applied to the city of Lisbon which considers how much walking customers need to do to each possible stop given an origin/destination matrix of user's demand. To tackle the problem they divide the problem into four phases. Firstly, they compute the possible demand for each one of the areas of the study, checking the willingness of customers to use this service. Then, they calculate the possible location for the minibus stops depending on different factors such as demand periods and the demand for each period. The third phase focuses on computing links between each stop, taking into account the possible demand. Finally, they decide the routes for each vehicle maximizing the profit of the total tour considering each of the arcs demand.

Another proposed method for integrating walking distance constraints can be seen in Lang et al. (2014). The aim of their problem is to create alternative stopping points for delivery services to reduce the fuel consumption of vehicles. While they do not tackle directly the concept of walking distance, to create these new points, the authors take into consideration the delivery man tolerance towards this new point which can be seen as a distance constraint.

A similar approach to what we present in this chapter can be seen in De Grancy and Reimann (2015). The authors present two heuristics to create customer clusters with time windows. There is a set of parking stations where trucks can stop, and the assigned workers need to deliver the goods from each parking station to as many customers as possible. Each customer will be assigned to a parking station and all customers assigned to that parking station will form a cluster. The walking distance between the parking stations and each customer must meet specific time windows constraints.

6.2 Adding walking distance to the heuristic

For our approach we will use walking distance as a clustering method, by doing so, it will also allow us to reduce the size of the problem. This will be considered a pre-process before the assignment phase (Phase 1) of the algorithm (Fig. 6-1). Once this new clustering mechanism finishes we will apply the previously presented algorithm to solve the sharing problem, both with and without drivers.

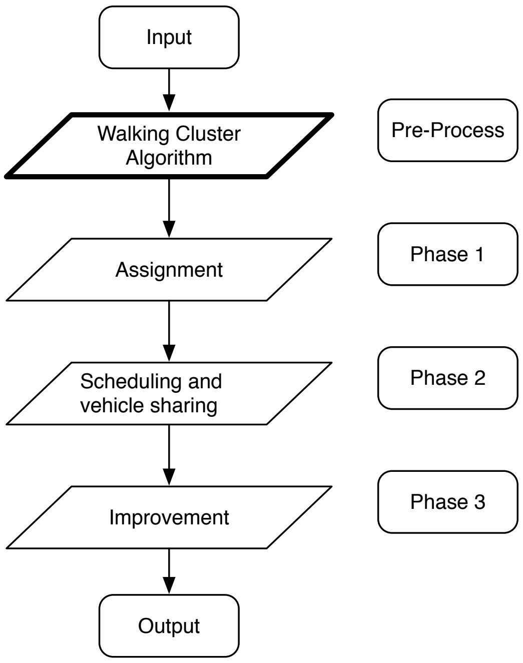

Figure 6-1: Summary of the algorithm process after adding clustering by walking at the beginning

The clustering for walking distance works as seen in Figure 6-2. Let us assume three points A, B, and C, which are within walking distance to each other. We will create a new point (Z) with new coordinates which will be the centroid of A, B, and C. For our problem, we assume that each point has to be within a distance of 10 minutes from the vehicle stop, and for every point we merge we will add 20 minutes to the new point (time for walking to the point and back), simulating the walking process. In Senarclens de Grancy and Reimann (2015), assume that people are willing to walk around 15 to 20 minutes, hence for the purpose of this study we will use 20 as added time. Thus, point Z will be the merging of three jobs done by the same worker, but they will be considered as a single job.

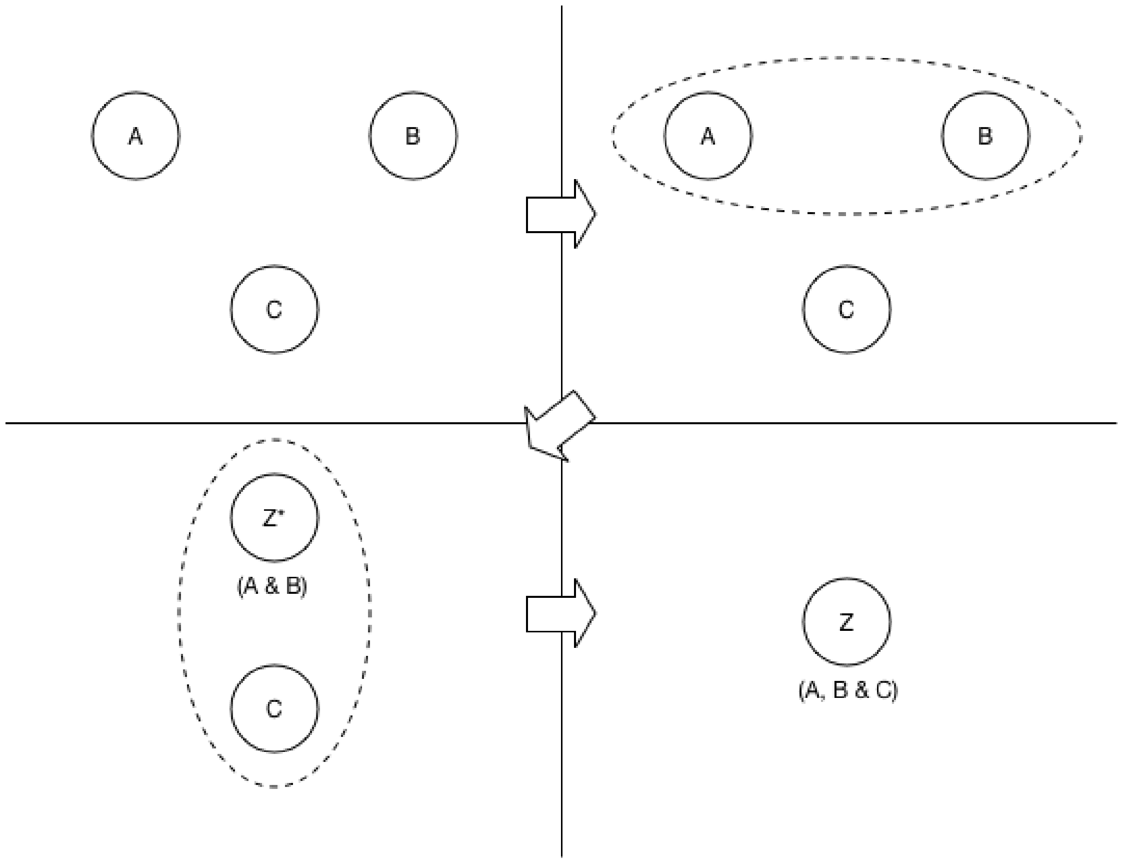

Figure 6-2: Graphical process of the clustering procedure by using walking as main motivator

An important aspects that should be considered is the transformation from driving distance to walking distance. This can highly vary from dense areas to less populated ones. In this problem we consider the distance using the time to drive between points by a vehicle, hence, we will estimate that a distance travelled by walking takes twice as long as by car. For example, a trip which takes 10 minutes by car, will take 20 minutes by foot. But, depending on the topography of each city/town these times might vary.

Finally, we have decided to apply this step as a pre-process and not after the assignment of jobs to workers. This is because, the Phase 1 algorithm tends to give very tight routes of nodes, and so merging after Phase 1 may not be feasible in most of the cases.

6.3 Example of the reduction

Previously we have presented how the clustering approach by walking works and the steps to group nodes into clusters. Figure 6-3 illustrates applying the method to scenario $c25\_1$. Two versions of this scenario are considered with and without clustering, where three different types of nodes can be seen. The circle nodes are for the scenario before applying the clustering, the square ones are after applying the clustering algorithm, and finally the triangle ones are nodes shared by both, i.e. these are points that could not be merged. Clearly a reduction on the number of points after clustering can be seen.

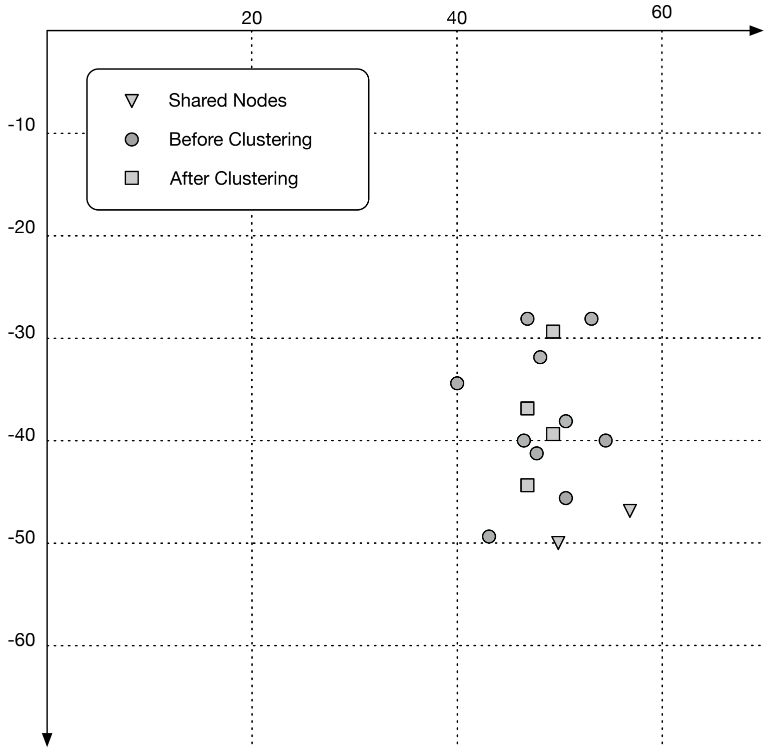

Figure 6-3: Example of the reduction in number of nodes for the $c25\_1$ scenario

6.4 Pseudocode

In this section we introduce the pseudocode (Alg. 14) for the clustering algorithm based on walking distance. The structure of it is quite straight forward.

By using the nodes from each instance, we start an iterative process to merge as many nodes as possible. Hence, while we still have nodes to merge we create a new list of nodes called *citiesToMerge* that represent the nodes which will be combined. Then another iterative process will start which will go through each one of the nodes remaining and check if it can be possibly merged. If a node can be merged it will be added to the list. Notice that at each iteration *citiesToMerge* can contain more than one node depending on if previous mergings are possible, if no more can be found it will become a single node. Finally, we will remove this *citiesToMerge* from the list of initial nodes, create the new node and add it to the new list of merged nodes.

Algorithm 14 ClusteringWalk(cities)

```
1: while cities is not empty do
2:     citiesToMerge ← FirstCity(cities)
3:     for $c$ in cities do
4:         if canBeMerged(citiesToMerge,c) then
5:             citiesToMerge ← Add(c)
6:         end if
7:     end for
8:     cities ← RemoveCities(cities,citiesToMerge)
9:     newCity ← Clustering(citiesToMerge)
10:    solCities ← Add(newCity)
11: end while
12: return solCities
```

6.5 Results

We will present the results obtained by applying the clustering algorithm based on walking distance. As this is aimed as a pre-process for the main algorithm, it allows us to reduce the number of nodes for each instance. In Figure 6-4, we present such reductions for each one of the scenarios compared to the number of initial nodes (shown as ”Before” in the graph). It can be seen that, in general, including walking can greatly reduce the size of the problem regardless of the duration of jobs. More precisely, short duration scenarios, as expected, are the ones where the number of nodes can be reduced the most, while in all day jobs, grouping nodes only work in such jobs with shorter duration.

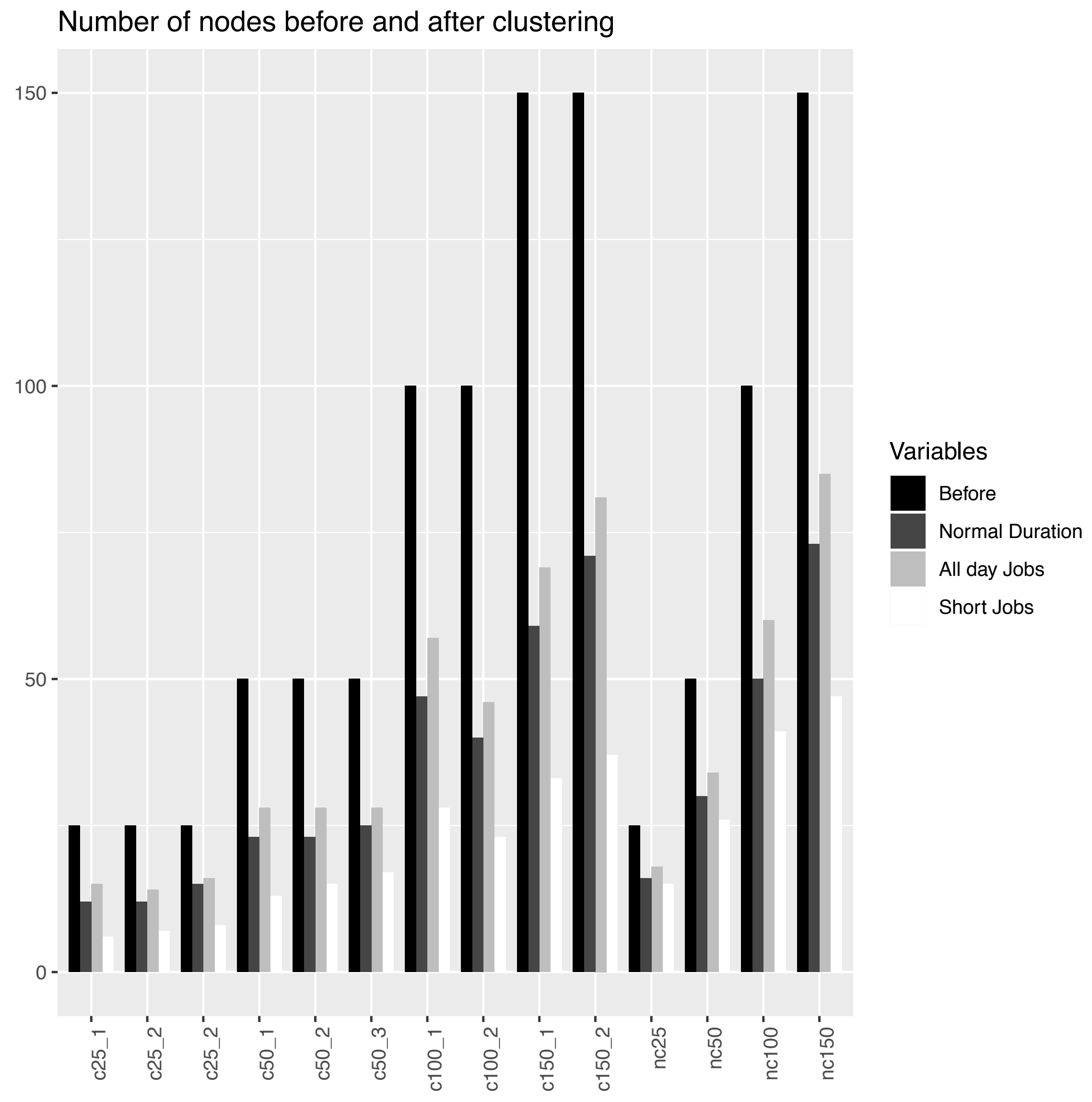

Figure 6-4: Number of nodes after applying the clustering walking distance based algorithm

We will next compare the results using walking distance for both workers as drivers and dedicated drivers problems. Having seen the reduction in the number of nodes in Figure 6-4 there might be an advantage of travelling by foot to reduce both the distance of the vehicles and the size of the problem. But, how does it perform compared to not using walking? The next sections will try to give some insight to this question.

6.5.1 Non-sharing Results

To start with, we compare the solutions without sharing vehicles using a new set of instances resulted from applying the walking distance pre-process. Again, we will present the results separately according to which of the three algorithms presented in chapter 4 is used to solve the non-sharing problem in Phase 1.

The results for RCWS are shown in the next graphs (Figures 6-5 and 6-6). The figures show two main characteristics for each one of the scenarios. Firstly, the percentage difference in distance units is shown as the scenarios have different total distances. Hence the percentage difference of total distance between before sharing and after sharing can be seen. The second graph shows the number of vehicles used, before sharing and after sharing. The remaining figures can be seen in Appendix C.

The number of vehicles used before and after applying walking show quite interesting results. In general, for all scenarios and algorithms the number of vehicles without sharing is higher after applying the walking distance clustering. The workers need more time for walking and therefore more workers are required to complete the jobs increasing the total number of vehicles. Therefore, the total cost increases in almost all of them.

After analyzing these results, we can clearly state that by using our clustering technique for walking, it does not give good results if vehicle sharing is not considered. Thus, it would be a better approach to solve the problem without traveling on foot in almost all of our instances. But, the aim of this chapter is to test if there is any improvement for sharing, of which the results are presented in the section below.

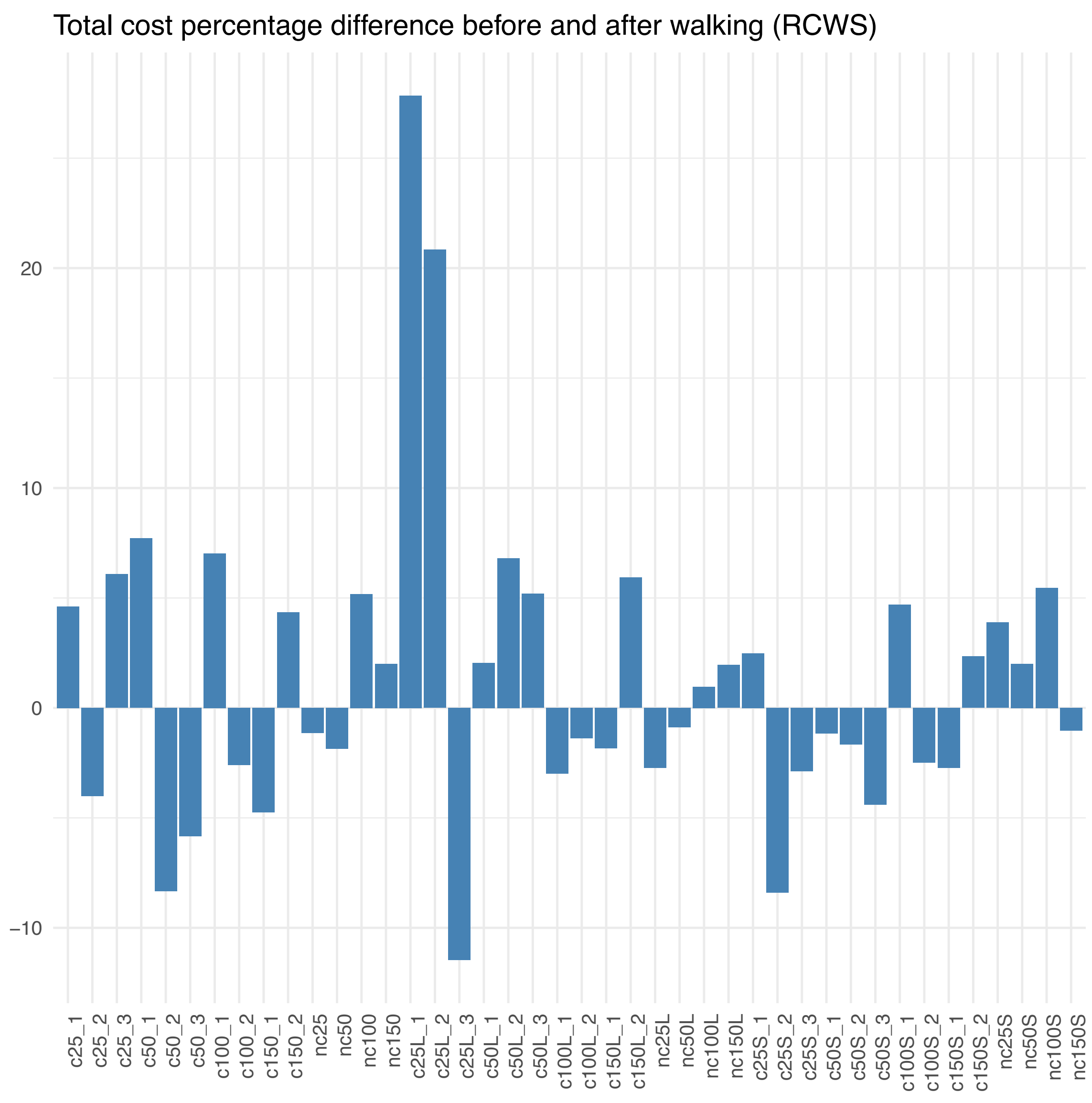

Figure 6-5: Percentage difference between the total distance cost using the RCWS approach without sharing and after sharing allowing to walking between jobs

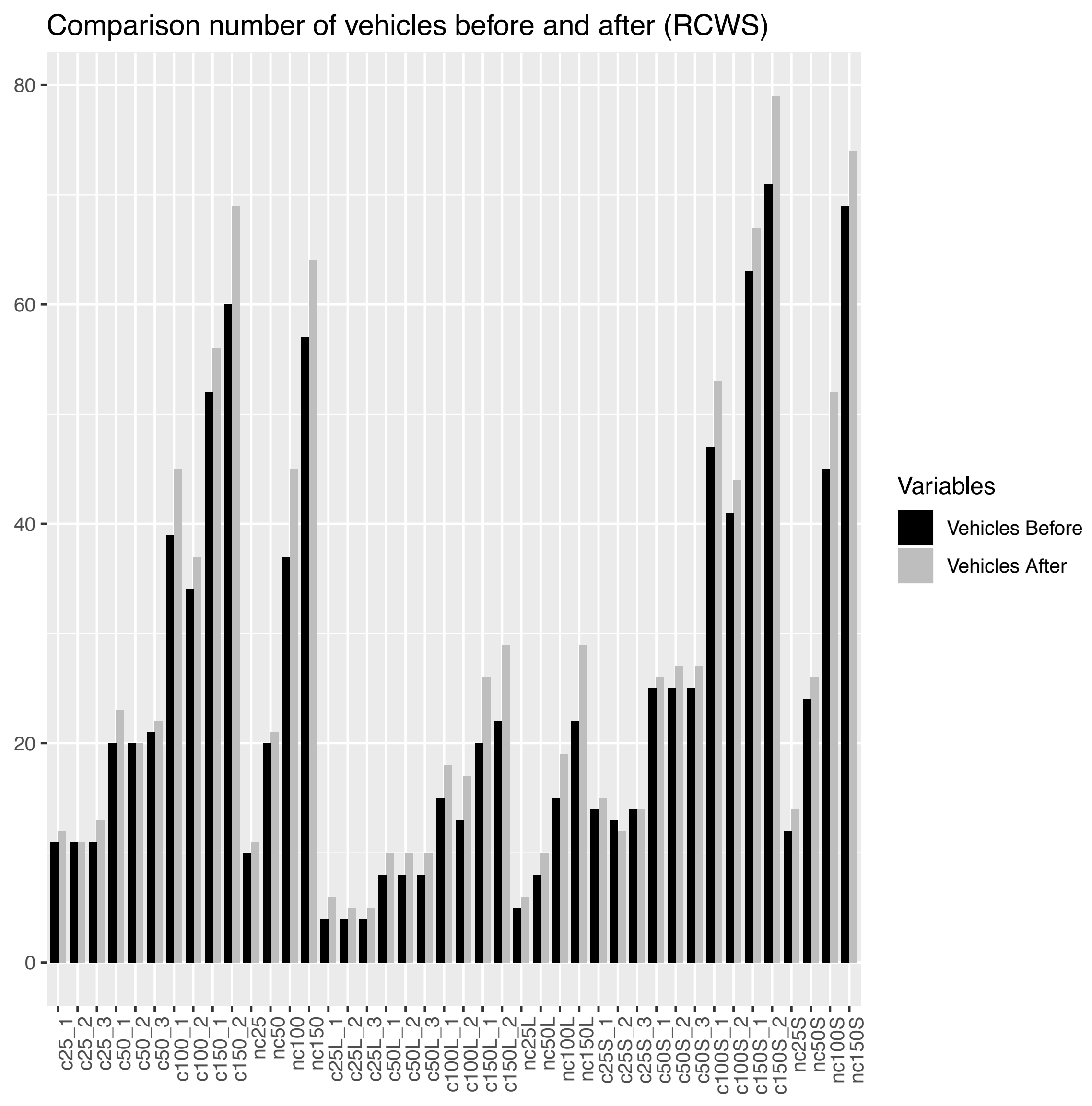

Figure 6-6: Number of vehicles used for the RCWS approach without sharing and after sharing allowing to walking between jobs

6.5.2 Sharing Results

To analyse the effects of including walking on the case of sharing vehicles, there are two comparisons possible; using or not using drivers. Again, we will separate the results obtained by means of the three algorithms in Phase 1. To start with, we will present the results without drivers and then examine the use of dedicated drivers.

Sharing without Drivers

A summary of the results of using the workers as drivers can be seen in Tables 6.1 to 6.3 for the RCWS. To compare the results we use the number of vehicles and the cost of total distance before and after the application of the walking pre-process for different job durations, where Best Solution Before (BSB) and Best Solution After (BSA) are introduced to show the results before and after sharing.

The results show that, with walking, the number of vehicles and the total distance of vehicles decreases significantly. Moreover, we are able to find sharing possibilities in scenarios where it was not possible without walking. One of the most interesting points, is that for our scenarios by allowing workers to walk between jobs but not share vehicles the results perform much worse. While in contrast, by sharing vehicles and walking between jobs seems to highly improve the results.

Randomized Clarke and Wright results

Table 6.1: Results comparing the number of vehicles, and total distance, before and after applying the clustering pre-process using the RCWS with normal job times.

	nvehicles before	nvehicles after	BSB	BSA
c25_1	9	4	1395.54	604.93
c25_2	9	4	1203.84	579.76
c25_2	10	5	1411.80	838.36
c50_1	18	9	2318.85	1341.61
c50_2	-	11	-	1316.66
c50_3	-	9	-	1187.45
c100_1	37	17	4273.32	2388.74
c100_2	32	15	2984.76	1689.75
c150_1	50	23	4691.82	2477.82
c150_2	-	27	-	3688.79
nc25	-	8	-	1472.17
nc50	-	14	-	2091.23
nc100	-	19	-	3305.83
nc150	-	30	-	4759.82

Table 6.2: Results comparing the number of vehicles, and total distance, before and after applying the clustering pre-process using the RCWS with all day job times.

	nvehicles before	nvehicles after	BSB	BSA
c25L_1	10	6	1518.60	874.14
c25L_2	8	5	1200.59	752.82
c25L_3	9	8	1248.63	1100.80
c50L_1	16	11	2295.72	1644.62
c50L_2	17	11	2081.53	1259.62
c50L_3	18	10	2027.97	1213.79
c100L_1	34	21	4186.73	2976.16
c100L_2	34	17	3282.80	1913.89
c150L_1	48	27	4685.99	2790.04
c150L_2	54	30	6737.75	3900.74
nc25L	10	8	1607.96	1519.11
nc50L	20	14	2566.27	2098.64
nc100L	36	21	5136.76	3424.22
nc150L	54	34	6708.44	5011.06

Table 6.3: Results comparing the number of vehicles, and total distance, before and after applying the clustering pre-process using the RCWS with short job times.

	nvehicles before	nvehicles after	BSB	BSA
c25S_1	-	2	-	349.94
c25S_2	-	4	-	689.95
c25S_3	-	-	-	-
c50S_1	-	5	-	734.23
c50S_2	7	7	968.83	963.69
c50S_3	7	8	952.59	1017.68
c100S_1	-	11	-	1656.50
c100S_2	-	11	-	1382.99
c150S_1	-	11	-	1391.49
c150S_2	-	14	-	2150.43
nc25S	-	5	-	1021.93
nc50S	-	8	-	1335.52
nc100S	-	16	-	2558.20
nc150S	-	17	-	2867.57

Finally, an interesting finding is that while in previous chapters the differences between the algorithms used in Phase 1 lead to significant disparities in the final solution, once we apply the walking pre-process, the differences between them are minimal. This situation might be due to the reduction in the size of the problem which, asides of reducing the number of nodes, also reduces the complexity of the problem, as the duration of each new node is longer.

Sharing using dedicated drivers

Reducing the problem by clustering nodes using walking distance can also be applied when using dedicated drivers to solve the problem. The results shown in this section aim to show the best result for each of the scenarios compared to using drivers before applying the clustering by walking approach.

Randomized Clarke and Wright results

Table 6.4: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the RCWS with normal job times.

	nvehicles before	nvehicles after	ndrivers before	ndrivers after	BSB	BSA
c25_1	4	4	4	2	909.87	635.01
c25_2	4	4	4	1	824.88	582.05
c25_3	8	5	3	1	1253.50	849.19
c50_1	9	8	7	3	1668.71	1556.98
c50_2	14	8	4	5	1857.69	1227.62
c50_3	14	8	3	3	1900.21	1179.42
c100_1	24	17	10	6	3958.97	2583.86
c100_2	21	14	9	6	2944.97	1736.61
c150_1	29	24	13	5	3954.24	2713.49
c150_2	35	26	15	11	5675.97	3820.35
nc25	9	7	1	2	1528.90	1622.54
nc50	17	13	2	4	2513.40	2237.19
nc100	30	18	5	6	4898.72	3591.56
nc150	39	29	15	6	6680.63	4790.22

Table 6.5: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the RCWS with all day job times.

	nvehicles before	nvehicles after	ndrivers before	ndrivers after	BSB	BSA
c25L_1	7	6	3	1	1260.66	904.93
c25L_2	5	5	4	1	934.32	785.98
c25L_3	7	7	3	2	1143.84	1048.16
c50L_1	12	11	7	3	1998.32	1699.99
c50L_2	15	10	6	2	2018.26	1437.77
c50L_3	14	10	5	2	1929.86	1253.03
c100L_1	28	21	10	5	4103.07	3091.95
c100L_2	23	16	12	7	3197.20	2004.96
c150L_1	32	27	16	6	4174.81	3012.40
c150L_2	35	30	19	7	5699.29	4076.17
nc25L	9	8	2	2	1699.08	1765.80
nc50L	16	13	5	3	2622.64	2296.58
nc100L	30	21	9	7	5258.47	4018.48
nc150L	43	32	15	5	6695.39	5134.08

Table 6.6: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the RCWS with short job times.

	nvehicles before	nvehicles after	ndrivers before	ndrivers after	BSB	BSA
c25S_1	3	2	1	0	591.41	357.73
c25S_2	-	4	-	0	-	689.95
c25S_3	-	-	-	-	-	-
c50S_1	7	5	1	0	1202.66	736.62
c50S_2	-	7	-	0	-	963.69
c50S_3	7	8	1	0	981.61	1024.48
c100S_1	14	11	1	0	2094.73	1656.50
c100S_2	12	11	1	0	1584.40	1382.99
c150S_1	18	12	0	1	2275.95	1556.60
c150S_2	21	13	0	3	3064.16	2145.62
nc25S	-	5	-	1	-	1099.79
nc50S	-	8	-	1	-	1431.07
nc100S	14	15	1	2	2720.55	2600.90
nc150S	21	16	1	2	3425.25	2872.47

As in the previous section, by using walking we further manage to greatly reduce the number of vehicles, the number of drivers and the total distance of each scenario. Moreover, in most of the cases we can see a considerable improvement on the number of vehicles, while at the same time reducing the number of drivers needed. The results of the RCWS show that we are also able to share vehicles in scenarios, which were not possible if walking was not used. Again, we can see that the differences between algorithms are minimal. But, the inclusion of drivers, is one of the main discrepancies with the results shown above, as they play an important role in the final solution.

6.6 Summary and conclusions

In this chapter we have introduced the possibility of walking between jobs to reduce the size of the problems and to try to further minimize the usage of vehicles. This is quite a novel idea introduced in recent years for routing applications. In real life applications, it is not uncommon to find this feature in areas such as home healthcare systems and service technicians working in big cities.

For our case the results show it greatly affects the final solution given by the sharing algorithm. The results show an improvement in all the aspects of the problem when sharing (both distance and number of workers). It is quite interesting to note that when we allow walking and solve the problem without sharing, the results are quite worse compared to not using walking. Hence in most of our scenarios, it is not recommended to walk if we are not aiming to share vehicles. This might happen due to the clustering aspect as if we increase the duration of the artificial nodes and workers cannot be assigned the same number of jobs, therefore needing more vehicles.

Chapter 7

Conclusions and Future Research

In this thesis, we have formulated the Vehicle Sharing and Workforce Scheduling Problem (VSWSP). We have considered three variants of the problem, in the first the workers are the drivers, in the second dedicated drivers as well as workers are used to drive the vehicles and in the third we also allow workers to walk between jobs. The main aim has been to reduce the number of vehicles used while maintaining the same service.

7.1 Contributions

To understand the existing work related to the problem at hand, a comprehensive review of the state of the art literature has been presented. We have defined a new approach to classify sharing type problems and created a more systematic framework to locate possible gaps. To the best of our knowledge no research has considered sharing vehicle under the work schedule constraints. Most research focuses on either trip sharing when a vehicle is driven by a driver, which takes a person between two specific locations or the sharing of the vehicle by renting it for a period of time. In this thesis we proposed to share vehicles, and allow workers to revisit nodes and to travel through nodes which they are not working in to arrive to their assigned jobs. This research is the first to consider such a formulation for sharing vehicles.

The problem is defined through mathematical formulation in Chapter 3. Small instances of the problem can be solved using exact methods and we presented an example where sharing can be seen as a viable and better option than non-sharing. While the initial results support the idea of sharing vehicles, solving larger problems is not computationally viable due to the problem complexity.

Hence we proposed a heuristic approach which allows us to solve larger instances with up to 150 nodes in a fast and flexible manner. Most importantly we showed that if sharing is possible, improvement can be achieved both in computation time and in solution quality over the case of not sharing vehicles in most of the scenarios. We have implemented a variant of the Randomized Clarke and Wright algorithm presented in Juan et al. (2011) to solve the standard VRP problem i.e., non-sharing vehicles and use this to make comparison with the sharing vehicles approach. We compared the results of this randomized CWS algorithm with the state of the art benchmarks, the GAP from the best known solution (Section 4.3.2) shows very good results in short computational time. Hence we can use this with confidence to make comparison between sharing and non-sharing vehicles. Moreover, for smaller instances we managed to solve the problem without sharing to optimality, and then by applying our sharing heuristic we further improved those results. We also applied well-known improvement techniques commonly used to solve several VRPs variants (ILS, LS, LNS) and showed through experiment that generally, the LNS algorithm tends to find better results in short computational time. Hence, given the results, the presented methodology establishes a solid starting point to achieve our objective.

Another characteristic we consider is sharing vehicles using dedicated drivers. In Chapter 5 we formulated the problem by adapting the mathematical formulation previously presented. Both the formulation and algorithm presented allowed us to easily adapt to new variants; it takes only minor modifications to include drivers. One of the advantages in this methodology is the ability to use both dedicated drivers and worker for driving, which allows the algorithm to choose the best option among using dedicated drivers for all vehicles, not using them at all, or in some cases using them for only some vehicle. One of the drawbacks we appreciate is the high number of drivers used in the final solution, which for most of the cases seems not worthwhile considering the trade-off between the increment in drivers and reduction on the number of vehicles.

We also proposed the possibility of using walking between nodes on the VSWSP. This feature has been introduced as a pre-process of the overall algorithm. One of the goals of this thesis was to study if walking between jobs would improve the usage of vehicles. The results presented show some interesting outcomes. When there is no vehicle sharing, both the distance and the number of vehicles are higher in most of the cases which is natural as jobs become longer because of the additional time added for walking between nodes and therefore more workers will be needed. On the other hand, when considering sharing, it will perform much better. We highly reduced the usage of vehicles and total distance when allowing workers to walk between jobs.

Finally, the contributions of this thesis can be summarized as follows:

- A mathematical model for the VSWSP has been presented and its functionality shown for small instances.
- A simple and flexible heuristic was proposed and implemented to solve bigger instances of the VSWSP. Experiment results showed that it can efficiently solve the problem and, in most cases, improve the optimal solutions for non-sharing vehicles.
- Both the mathematical model and the heuristic were adapted to solve the vehicle sharing problem using dedicated drivers.
- An integrated approach was successfully developed to allow workers to walk between tasks if the distance is relatively small.
- It was shown that by sharing we can further improve the scheduling and routing of workers if the possibility to share vehicles is feasible. Furthermore, with less computational time, we cab find better results than traditional non-sharing approaches.

7.2 Future Research

There are several lines of research in this area open for future work. We present a short summary of these, separating them into two groups; operational and computational future work.

Operational side includes a group of new extensions and variants of the problem from an application perspective, and some future steps are discussed next.

- Workers with skills: One of the main features for workers in maintenance companies is the diversity of skills required in a roster. Scheduling workers with different skills was proposed by Dutot et al. (2006) as the contest problem in the 2007 ROADEF Challenge. Schedules considering worker skills might have more slack as their jobs are more restricted. This might be of use for our approach as it would lead to more possibilities for sharing.

- Environmental objective function. Currently in this thesis, we apply the standard objective function which appears in most of the VRP problems; the total distance cost. The application of such objective function is useful for comparison and validation of our algorithms. This objective could be extended in the future to a multi-objective optimization which also takes into consideration CO2 emissions, i.e. Green VRP.
- Heterogeneous vehicles. Companies tend to have more than one type of vehicles to suit the requirement of different jobs. In our case, the usage of different sized vans would affect the capacity constraint and the total number of merging and hence it would be interesting to investigate the effects of heterogenous vehicles on sharing.
- Tighter Time Windows. In the mathematical formulation we can solve problems with any time window constraints. However, when producing the sharing schedule in our algorithm, we consider time windows to be the whole working day. Thus, the implementation of such constraints and how they affect the sharing solution could be a feature to be explored.

From a computational perspective, there are some strategies that could be implemented and developed:

- Currently, our approach uses a rather sequential process to build the final sharing solution, in order to always return to a feasible solution if a better one is not found. Moreover, it allows us to return to a standard routing problem if no sharing solution can be found. Thus, the implementation of an algorithm which builds the sharing solution using an approach without multiple phases could be investigated.
- The implementation of an algorithm which allows workers to share more than one vehicle. The way sharing presented in this thesis is that as many workers as possible are assigned to a vehicle, once a worker is assigned to a vehicle this

vehicle must do all his pick-ups and drop offs. Another approach is to use a "n" vehicles and try to share them by "m" workers. Then, a worker dropped by one vehicle might be picked up by another vehicle.

Chapter 8

Appendices

A Chapter 4 Appendix

Complete results

Table A1: Results for base scenarios using clustering algorithm

	Non Sharing		Sharing				
	n_vehicles	totalcost	n_vehicles	totalcost	Shuffle	ILS	LNS
c25_1	11	1575.61	9	1400.75	1399.23	1416.37	1421.55
c25_2	11	1328.93	9	1237.00	1234.51	1234.51	1237.01
c25_3	12	1600.60	10	1387.00	1376.44	1376.44	1385.78
c50_1	20	2669.47	18	2484.27	2475.67	2475.79	2482.23
c50_2	20	2137.39	19	2145.08	2125.28	2131.68	2137.55
c50_3	20	2172.35	19	2052.97	1905.45	1906.58	1920.87
c100_1	38	4430.33	36	4455.49	4438.32	4430.88	4444.18
c100_2	34	3264.03	32	3272.96	3228.95	3231.95	3272.97
c150_1	51	4827.87	-	-	-	-	-
c150_2	59	7277.50	57	7335.86	7292.98	7286.41	7277.15
nc25	11	1665.23	10	1696.05	1642.03	1644.99	1636.61
nc50	21	2605.00	20	2571.7	2558.55	2558.55	2571.7
nc100	36	5220.09	-	-	-	-	-
nc150	57	7005.58	-	-	-	-	-

Table A2: Results for all day jobs scenarios using clustering algorithm

	Non Sharing		Sharing				
	n_vehicles	totalcost	n_vehicles	totalcost	Shuffle	ILS	LNS
c25L_1	15	2129.20	9	1411.67	1404.61	1404.61	1398.02
c25L_2	12	1592.18	8	1242.65	1234.50	1234.50	1237.04
c25L_3	14	1842.70	10	1604.65	1582.79	1582.79	1584.96
c50L_1	26	3280.55	17	2366.03	2360.07	2359.59	2316.73
c50L_2	25	2640.14	19	2334.79	2296.57	2296.57	2323.85
c50L_3	24	2491.99	17	2117.66	2104.65	2104.65	2114.73
c100L_1	46	5242.00	35	4614.53	4595.81	4598.15	4610.53
c100L_2	40	3708.77	34	3381.09	3353.06	3354.89	3374.64
c150L_1	62	5581.59	48	4874.84	4851.23	4847.84	4870.93
c150L_2	70	8318.66	57	7361.42	7327.14	7324.63	7359.16
nc25L	13	1800.61	10	1766.75	1750.19	1758.57	1760.48
nc50L	24	2870.31	19	2538.12	2516.95	2521.68	2531.31
nc100L	45	5779.35	36	5366.58	5353.23	5350.18	5342.65
nc150L	69	8019.25	57	7052.88	6995.81	6992.99	7052.88

Table A3: Results for short jobs scenarios using clustering algorithm

	Non Sharing		Sharing				
	n_vehicles	totalcost	—n_vehicles	totalcost	Shuffle	ILS	LNS
c25S_1	4	688.95	-	-	-	-	-
c25S_2	4	639.23	-	-	-	-	-
c25S_3	5	781.67	-	-	-	-	-
c50S_1	8	1269.61	-	-	-	-	-
c50S_2	8	1120.89	-	-	-	-	-
c50S_3	8	1111.33	-	-	-	-	-
c100S_1	15	2246.81	-	-	-	-	-
c100S_2	14	1820.20	-	-	-	-	-
c150S_1	20	2569.39	-	-	-	-	-
c150S_2	23	3416.40	-	-	-	-	-
nc25S	6	1281.62	5	1181.93	1181.93	1181.12	1177.61
nc50S	9	1636.97	-	-	-	-	-
nc100S	16	3086.69	-	-	-	-	-
nc150S	23	3817.21	-	-	-	-	-

Table A4: Results for base scenarios using Clarke and Wright algorithm

	Non Sharing		Sharing				
	n_vehicles	totalcost	n_vehicles	totalcost	Shuffle	ILS	LNS
c25_1	11	1576.93	10	1463.89	1362.85	1362.85	1352.13
c25_2	11	1382.81	9	1234.08	1219.25	1219.25	1213.88
c25_3	11	1534.64	10	1435.83	1423.36	1423.36	1423.37
c50_1	20	2529.78	18	2347.44	2336.44	2336.44	2336.44
c50_2	20	2115.55	19	2125.21	2115.88	2115.88	2115.06
c50_3	21	2168.86	19	2097.25	2082.98	2097.25	2059.36
c100_1	39	4376.10	37	4276.60	4268.05	4268.05	4268.18
c100_2	34	3097.94	33	3103.85	3096.39	3096.39	3096.39
c150_1	52	4768.55	51	4798.08	4798.04	4798.04	4779.03
c150_2	60	7225.45	58	7033.37	6984.73	6984.73	6984.73
nc25	10	1511.78	-	-	-	-	-
nc50	20	2510.72	-	-	-	-	-
nc100	37	5177.85	36	5239.48	5239.48	5236.77	5173.15
nc150	57	6787.09	-	-	-	-	-

Table A5: Results for all day jobs scenarios using the Clarke and Wright Algorithm

	Non Sharing		Sharing				
	n_vehicles	totalcost	n_vehicles	totalcost	Shuffle	ILS	LNS
c25L_1	14	1983.44	10	1518.60	1518.60	1518.60	1518.61
c25L_2	13	1622.65	7	1107.26	1092.95	1092.95	1073.39
c25L_3	14	1760.33	9	1267.23	1267.23	1267.23	1248.63
c50L_1	25	3197.39	17	2385.80	2383.89	2385.8	2368.8
c50L_2	25	2561.96	17	2089.46	2089.10	2089.10	2078.59
c50L_3	25	2450.48	18	2178.23	2177.78	2177.78	2141.44
c100L_1	47	5253.75	37	4358.60	4357.39	4357.51	4356.16
c100L_2	41	3647.72	32	3225.73	3222.13	3223.73	3220.20
c150L_1	63	5447.52	48	4815.54	4807.73	4807.46	4784.02
c150L_2	71	8323.27	55	7043.86	7039.28	7039.28	7039.29
nc25L	12	1650.09	10	1607.96	1607.96	1607.96	1607.97
nc50L	24	2775.68	20	2585.03	2585.03	2585.03	2578.22
nc100L	45	5683.17	35	5228.85	5208.51	5208.51	5208.51
nc150L	69	7778.81	53	6710.22	6697.07	6697.07	6701.88

Table A6: Results for short jobs scenarios using the Clarke and Wright Algorithm

	Non Sharing		Sharing				
	n_vehicles	totalcost	n_vehicles	totalcost	Shuffle	ILS	LNS
c25S_1	4	638.59	-	-	-	-	-
c25S_2	4	601.02	-	-	-	-	-
c25S_3	4	717.05	-	-	-	-	-
c50S_1	8	1231.62	-	-	-	-	-
c50S_2	8	966.10	7	981.14	976.20	981.14	971.73
c50S_3	8	968.12	-	-	-	-	-
c100S_1	15	2155.71	-	-	-	-	-
c100S_2	13	1548.10	-	-	-	-	-
c150S_1	20	2209.78	19	2216.93	2216.93	2216.93	2216.93
c150S_2	22	3112.95	-	-	-	-	-
nc25S	5	1079.65	-	-	-	-	-
nc50S	8	1386.87	-	-	-	-	-
nc100S	15	2623.45	-	-	-	-	-
nc150S	22	3351.74	-	-	-	-	-

Table A7: Results for base scenario using Randomized Clarke and Wright

	Non Sharing		Sharing				
	n_vehicles	totalcost	n_vehicles	totalcost	Shuffle	ILS	LNS
c25_1	11	1566.00	9	1406.10	1395.54	1395.54	1399.99
c25_2	10	1295.76	9	1220.32	1203.84	1203.84	1205.63
c25_3	11	1476.34	10	1427.92	1412.53	1412.53	1411.80
c50_1	20	2536.81	18	2318.85	2318.85	2318.85	2318.85
c50_2	19	2090.76	-	-	-	-	-
c50_3	19	2020.12	-	-	-	-	-
c100_1	38	4329.08	37	4324.64	4273.32	4273.32	4273.32
c100_2	33	3019.53	32	3020.28	3017.74	3017.46	2984.76
c150_1	51	4702.32	50	4733.98	4733.93	4733.98	4691.82
c150_2	58	7014.11	-	-	-	-	-
nc25	10	1564.02	-	-	-	-	-
nc50	19	2479.77	-	-	-	-	-
nc100	35	5065.87	-	-	-	-	-
nc150	56	6711.09	-	-	-	-	-

Table A8: Results for all day jobs scenarios using Randomized Clarke and Wright

	Non Sharing		Sharing				
	n_vehicles	totalcost	n_vehicles	totalcost	Shuffle	ILS	LNS
c25L_1	14	1983.44	10	1518.60	1518.60	1518.61	1518.61
c25L_2	12	1546.68	8	1200.59	1200.59	1200.59	1200.59
c25L_3	14	1760.32	9	1267.23	1267.23	1267.23	1248.63
c50L_1	24	3101.38	16	2296.21	2296.21	2295.72	2295.72
c50L_2	25	2553.23	17	2092.41	2092.04	2092.04	2081.53
c50L_3	24	2345.11	18	2028.42	2027.97	2027.97	2027.97
c100L_1	46	5071.27	34	4229.82	4220.67	4220.67	4186.73
c100L_2	40	3588.11	34	3284.80	3282.80	3283.16	3282.8
c150L_1	62	5346.90	48	4698.62	4687.66	4689.72	4685.99
c150L_2	71	8145.83	54	6741.62	6737.75	6737.75	6768.33
nc25L	12	1650.09	10	1607.96	1607.96	1607.96	1607.96
nc50L	23	2759.32	20	2566.27	2566.27	2566.27	2566.27
nc100L	44	5525.67	36	5142.39	5136.76	5136.76	5136.76
nc150L	68	7654.07	54	6824.86	6708.44	6824.41	6824.41

Table A9: Results for short jobs scenarios using Randomized Clarke and Wright

	Non Sharing		Sharing				
	n_vehicles	totalcost	n_vehicles	totalcost	Shuffle	ILS	LNS
c25S_1	4	627.70	-	-	-	-	-
c25S_2	4	592.36	-	-	-	-	-
c25S_3	5	717.05	-	-	-	-	-
c50S_1	8	1215.97	-	-	-	-	-
c50S_2	8	963.48	-	-	-	-	-
c50S_3	8	958.99	7	952.59	952.59	952.59	952.59
c100S_1	15	2003.36	-	-	-	-	-
c100S_2	13	1521.98	-	-	-	-	-
c150S_1	19	2152.96	-	-	-	-	-
c150S_2	19	2152.96	-	-	-	-	-
nc25S	5	1028.70	-	-	-	-	-
nc50S	8	1314.16	-	-	-	-	-
nc100S	15	2545.79	-	-	-	-	-
nc150S	22	3242.65	-	-	-	-	-

Table A10: Comparison of the number of vehicles between sharing and non-sharing and the three different methodologies tested

	CWS		RCWS		Cluster	
	n_vehicles and n_workers	n_vehicles sharing	n_vehicles and n_workers	n_vehicles sharing	n_vehicles and n_workers	n_vehicles sharing
c25_1	11	10	11	9	11	9
c25_2	11	9	10	9	11	9
c25_3	11	10	11	10	12	10
c50_1	20	18	20	18	20	18
c50_2	20	19	19	-	20	19
c50_3	21	19	19	-	20	19
c100_1	39	37	38	37	38	36
c100_2	34	33	33	32	34	32
c150_1	52	51	51	50	51	-
c150_2	60	58	58	-	59	57
nc25	10	-	10	-	11	10
nc50	20	-	19	-	21	20
nc100	37	36	35	-	36	-
nc150	57	-	56	-	57	-
c25L_1	14	10	14	10	15	9
c25L_2	13	7	12	8	12	8
c25L_3	14	9	14	9	14	10
c50L_1	25	17	24	16	26	17
c50L_2	25	17	25	17	25	19
c50L_3	25	18	24	18	24	17
c100L_1	47	37	46	34	46	35
c100L_2	41	32	40	34	40	34
c150L_1	63	48	62	48	62	48
c150L_2	71	55	71	54	70	57
nc25L	12	10	12	10	13	10
nc50L	24	20	23	20	24	19
nc100L	45	35	44	36	45	36
nc150L	69	53	68	54	69	57
c25S_1	4	-	4	-	4	-
c25S_2	4	-	4	-	4	-
c25S_3	4	-	5	-	5	-
c50S_1	8	-	8	-	8	-
c50S_2	8	7	8	-	8	-
c50S_3	8	-	8	7	8	-
c100S_1	15	-	15	-	15	-
c100S_2	13	-	13	-	14	-
c150S_1	20	19	19	-	20	-
c150S_2	22	-	19	-	23	-
nc25S	5	-	5	-	6	5
nc50S	8	-	8	-	9	-
nc100S	15	-	15	-	16	-
nc150S	22	-	22	-	23	-

Table A11: Comparison of the objective values between sharing and non-sharing and the three different methodologies tested

	CWS		RCWS		Cluster	
	Non Sharing	Sharing	Non Sharing	Sharing	Non Sharing	Sharing
c25_1	1581.41	1362.85	1566.75	1395.54	1575.60	1399.23
c25_2	1383.91	1219.25	1310.46	1203.84	1395.86	1234.51
c25_3	1547.10	1423.36	1548.96	1412.53	1600.59	1376.44
c50_1	2545.37	2336.44	2536.81	2318.85	2669.47	2475.67
c50_2	2124.88	2115.88	2090.76	-	2137.39	2125.28
c50_3	2168.85	2082.98	2020.12	-	2172.34	1905.45
c100_1	4384.52	4268.05	4329.08	4273.32	4430.32	4430.88
c100_2	3105.40	3096.39	3019.53	3017.46	3264.02	3228.95
c150_1	4786.34	4798.04	4702.32	4733.93	4827.87	-
c150_2	7236.71	6984.73	7014.11	-	7277.49	7286.41
nc25	1540.81	-	1564.02	-	1665.23	1642.03
nc50	2546.06	-	2479.77	-	2604.99	2558.55
nc100	5211.65	5236.77	5065.87	-	5220.09	-
nc150	6811.25	-	6711.09	-	7005.58	-
c25L_1	1983.44	1518.60	1983.44	1518.60	2129.19	1404.61
c25L_2	1622.65	1092.95	1546.68	1200.59	1592.18	1234.50
c25L_3	1760.32	1267.23	1760.32	1267.23	1842.69	1582.79
c50L_1	3197.39	2383.89	3101.38	2295.72	3280.54	2359.59
c50L_2	2561.96	2089.10	2553.23	2092.04	2640.14	2296.57
c50L_3	2450.47	2177.78	2345.11	2027.97	2491.99	2104.65
c100L_1	5253.74	4357.39	5071.27	4220.67	5242.00	4595.81
c100L_2	3647.72	3222.13	3588.11	3282.80	3708.77	3353.06
c150L_1	5447.52	4807.46	5346.90	4687.66	5581.59	4847.84
c150L_2	8323.27	7039.28	8145.83	6737.75	8318.66	7324.63
nc25L	1650.09	1607.96	1650.09	1607.96	1800.61	1750.19
nc50L	2775.68	2585.03	2759.32	2566.27	2870.31	2516.95
nc100L	5683.16	5208.51	5525.67	5136.76	5779.35	5350.18
nc150L	7778.81	6697.07	7654.07	6708.44	8019.25	6992.99
c25S_1	638.58	-	627.70	-	688.94	-
c25S_2	601.02	-	592.36	-	639.23	-
c25S_3	717.05	-	717.05	-	781.66	-
c50S_1	1231.62	-	1215.97	-	1269.60	-
c50S_2	966.09	976.20	963.48	-	1120.89	-
c50S_3	968.12	-	958.99	952.59	1111.33	-
c100S_1	2155.70	-	2003.36	-	2246.80	-
c100S_2	1548.09	-	1521.98	-	1820.19	-
c150S_1	2209.77	2216.93	2152.96	-	2569.38	-
c150S_2	3112.95	-	2152.96	-	3416.39	-
nc25S	1079.64	-	1028.70	-	1281.61	1181.12
nc50S	1386.86	-	1314.16	-	1636.96	-
nc100S	2623.45	-	2545.79	-	3086.68	-
nc150S	3351.73	-	3242.65	-	3817.02	-

B Chapter 5 Appendix

Trade off between drivers and vehicles

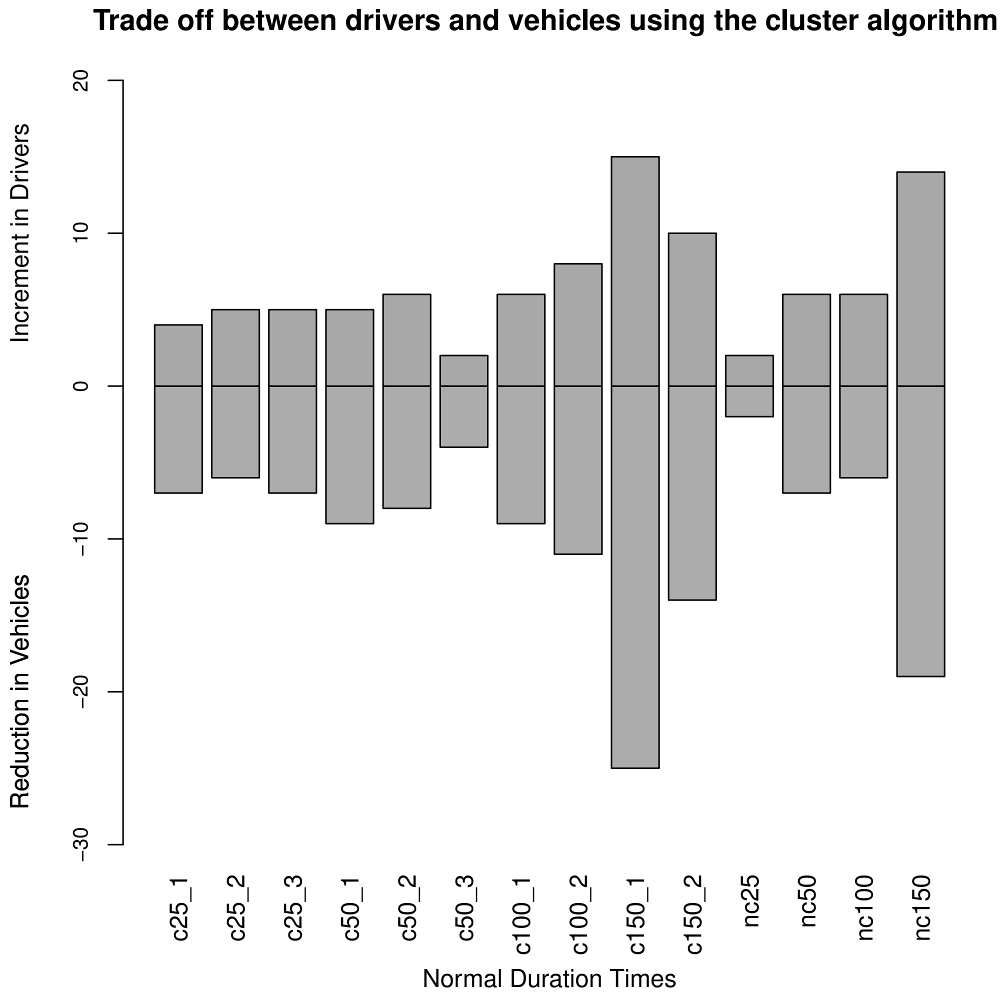

Figure B1: Trade off between drivers and number of vehicles on base scenarios using the clustering algorithm as initial solution

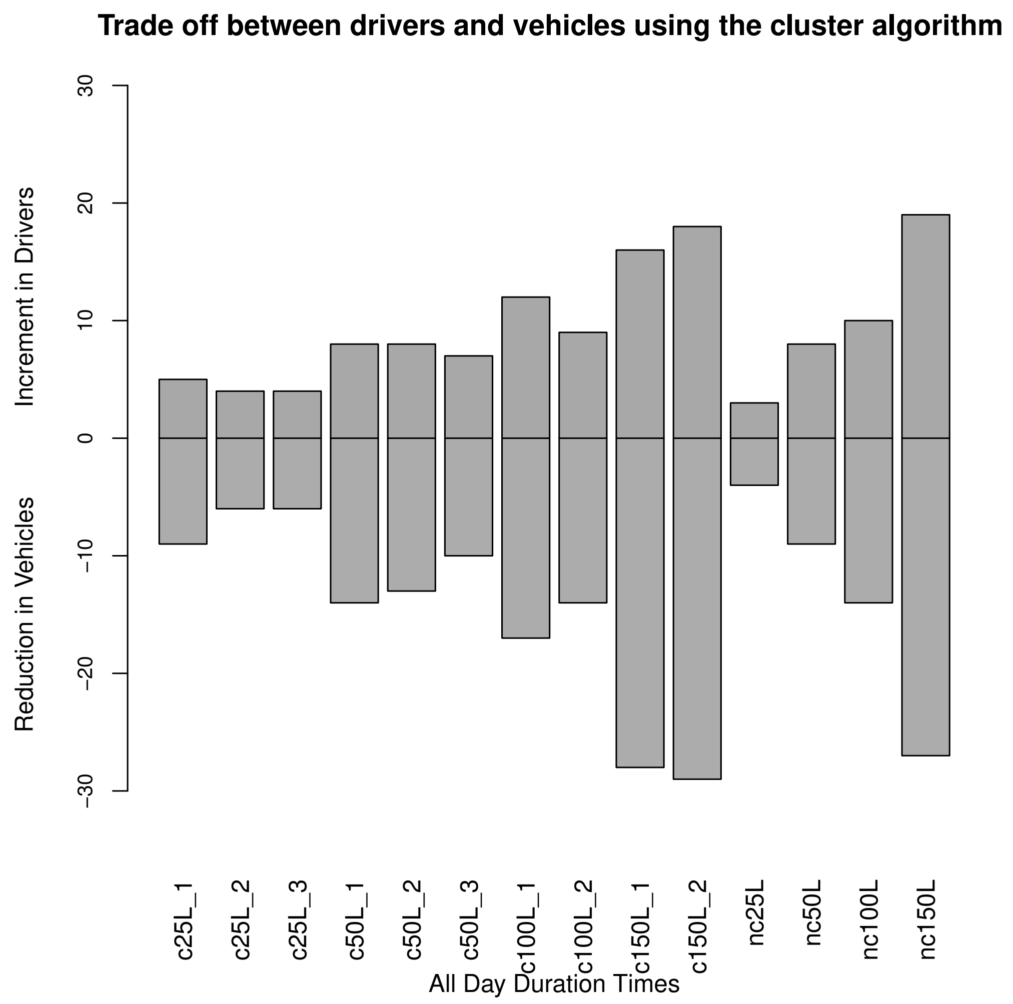

Figure B2: Trade off between drivers and number of vehicles on all day jobs using the clustering algorithm as initial solution

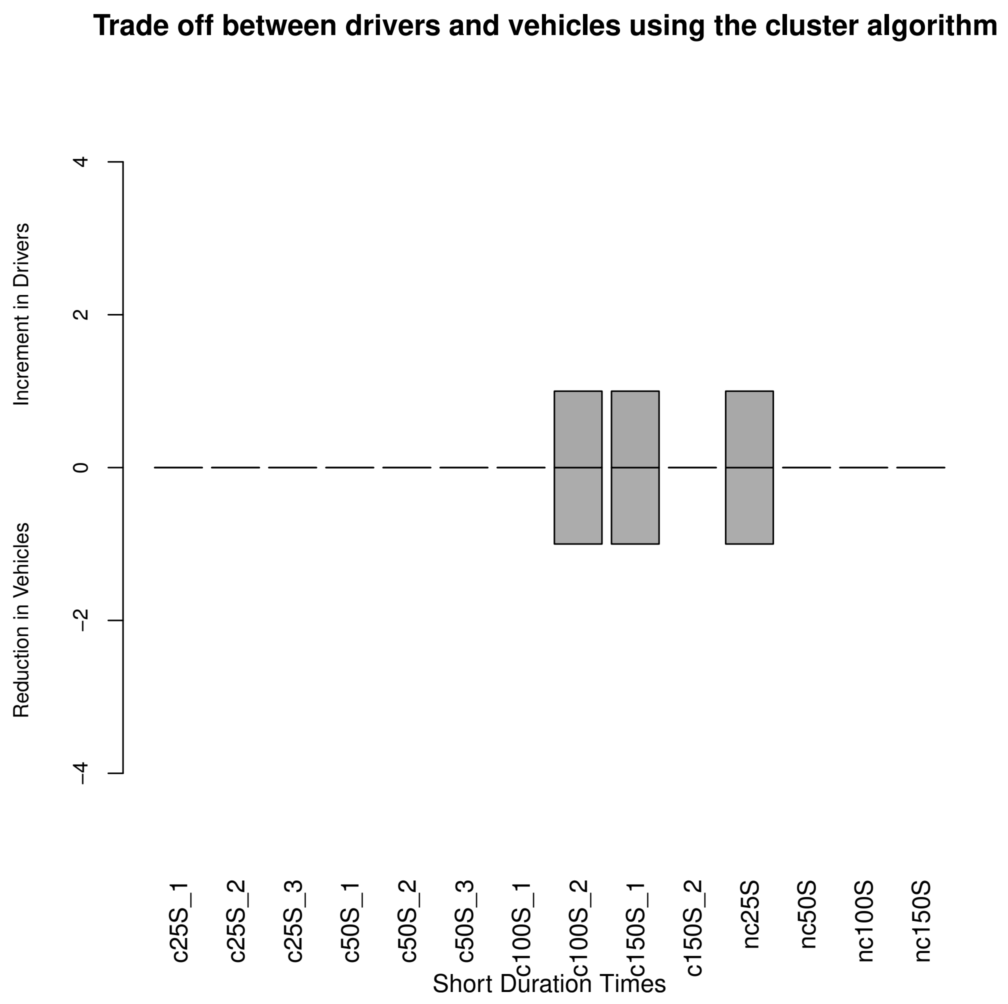

Figure B3: Trade off between drivers and number of vehicles on short jobs using the clustering algorithm as initial solution

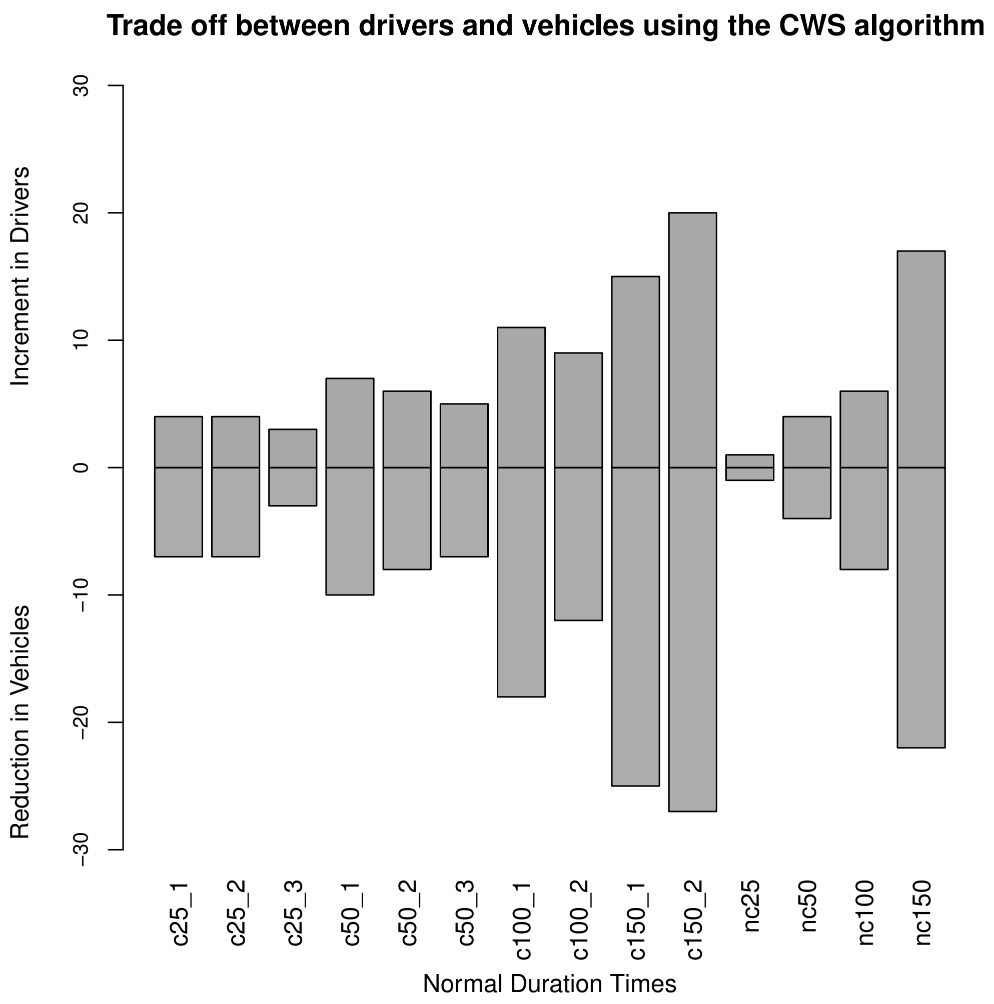

Figure B4: Trade off between drivers and number of vehicles on base scenarios using the CWS algorithm as initial solution

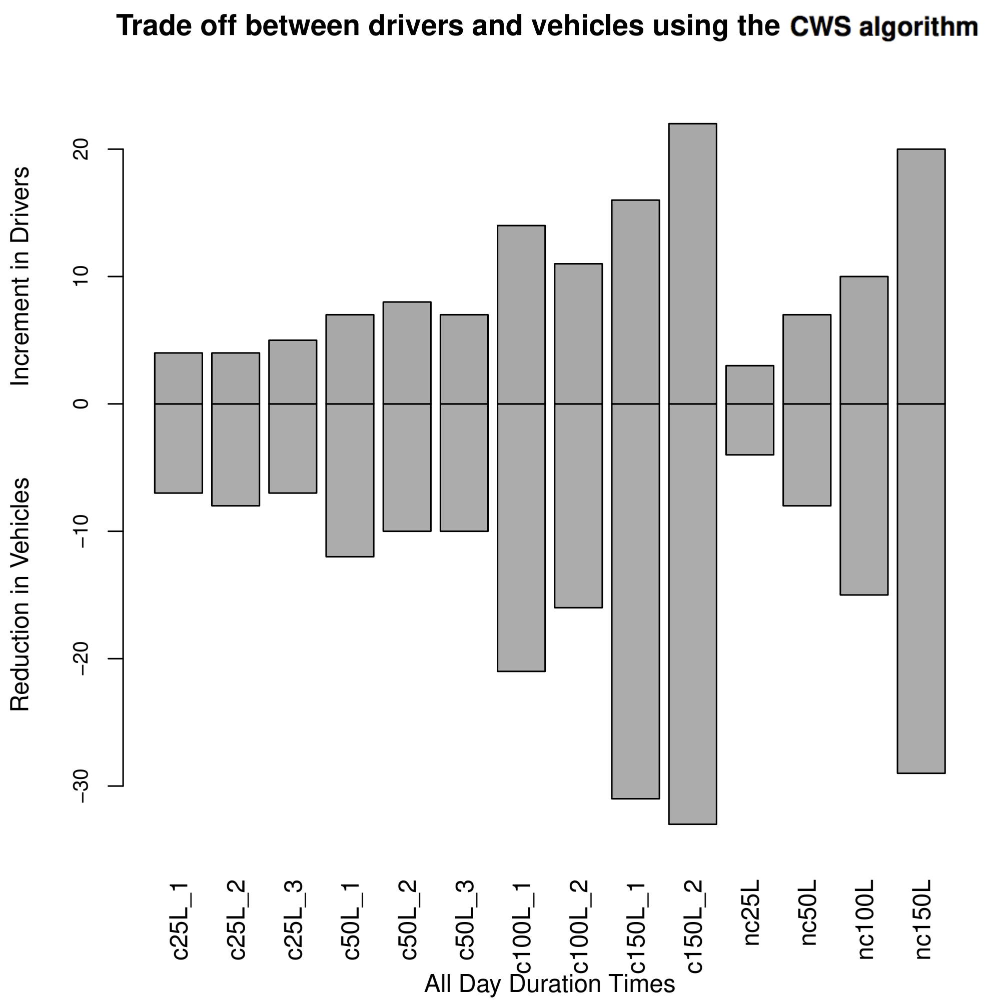

Figure B5: Trade off between drivers and number of vehicles on all day jobs using the CWS algorithm as initial solution

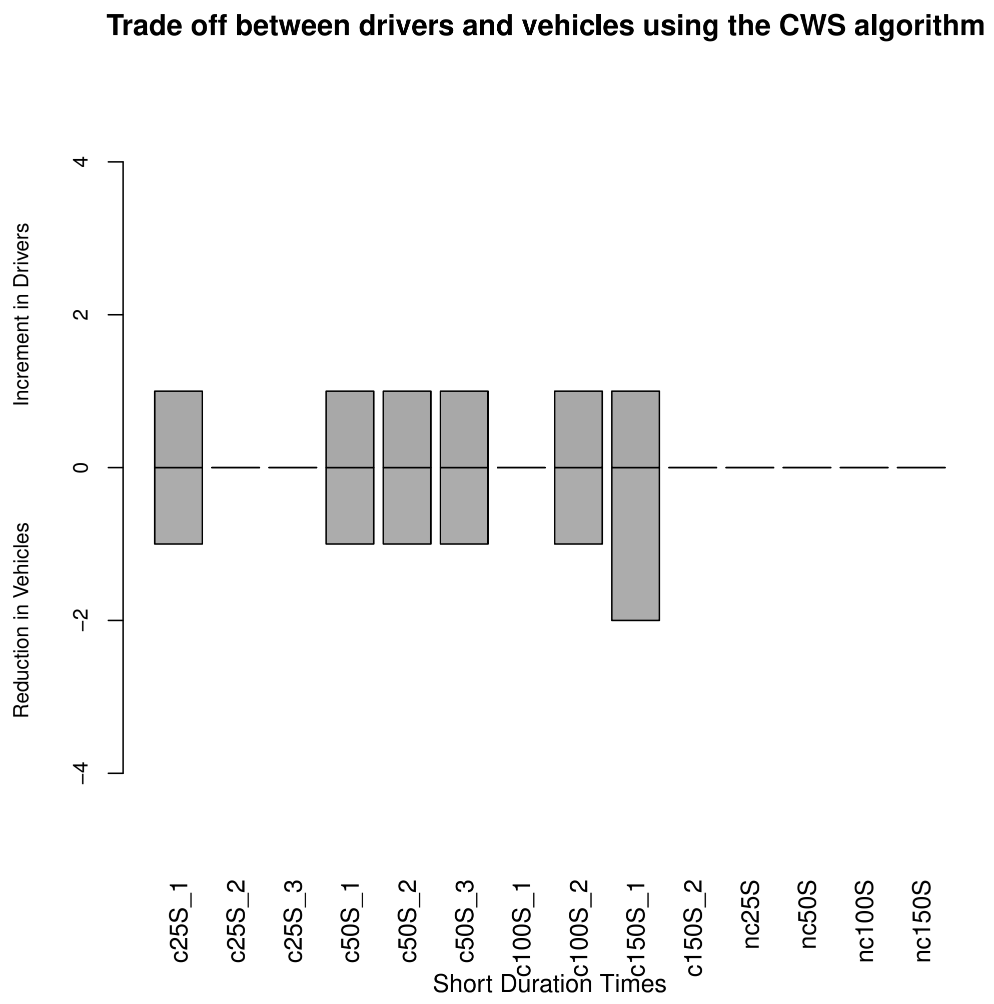

Figure B6: Trade off between drivers and number of vehicles on short jobs using the CWS algorithm as initial solution

Results using dedicated drivers

Table B1: Results for base scenarios using clustering algorithm and drivers

	No Sharing		Sharing					
	n_vehicles	Cost	n_vehicles	n_drivers	Cost	Shuffle	ILS	LNS
c25_1	11	1575.61	4	4	899.71	880.97	897.23	858.77
c25_2	10	1328.93	5	4	1000.47	961.14	954.1	941.58
c25_3	12	1600.60	5	5	1181.09	1161.08	1172.86	1107.02
c50_1	20	2669.47	11	5	2051.31	2000.40	1947.46	1965.18
c50_2	20	2137.39	12	6	1783.61	1754.43	1739.91	1712.57
c50_3	20	2172.35	16	2	2009.01	1978.65	1979.86	1990.34
c100_1	38	4430.33	29	6	4261.09	4216.86	4233.08	4231.43
c100_2	34	3264.03	23	8	3059.78	3022.57	3015.62	3019.99
c150_1	51	4827.87	26	15	4180.69	4038.87	4022.00	3983.89
c150_2	59	7277.50	45	10	6692.46	6635.78	6655.83	6626.14
nc25	11	1665.23	9	2	1716.03	1638.25	1638.25	1698.25
nc50	21	2605.00	14	6	2690.89	2663.26	2658.61	2648.94
nc100	36	5220.09	30	6	5050.45	5026.25	5022.13	5035.27
nc150	57	7005.58	38	14	6861.51	6780.03	6786.1	6735.85

Table B2: Results for all day jobs scenarios using clustering algorithm and drivers

	No Sharing		Sharing					
	n_vehicles	Cost	n_vehicles	n_drivers	Cost	Shuffle	ILS	LNS
c25L_1	15	2129.20	6	4	1089.63	1070.30	1077.30	1074.07
c25L_2	12	1592.18	6	3	1088.22	1058.37	1069.98	1062.37
c25L_3	14	1842.70	8	2	1445.03	1442.87	1442.87	1445.03
c50L_1	26	3280.55	12	5	2021.82	2002.39	1992.27	1970.70
c50L_2	25	2640.14	12	7	1945.06	1915.89	1915.89	1905.95
c50L_3	24	2491.99	14	5	2249.95	2210.90	2222.48	2233.63
c100L_1	46	5242.00	29	11	4194.28	4176.58	4146.73	4127.65
c100L_2	40	3708.77	26	9	3379.10	3317.43	3319.15	3318.52
c150L_1	62	5581.59	34	16	4559.18	4507.8	4513.33	4487.21
c150L_2	70	8318.66	41	17	6535.56	6450.31	6435.62	6433.06
nc25L	13	1800.61	9	2	1876.20	1842.66	1858.88	1819.28
nc50L	24	2870.31	15	5	2606.78	2584.82	2586.53	2582.31
nc100L	45	5779.35	31	10	5310.98	5299.41	5302.60	5288.14
nc150L	69	8019.25	42	16	6898.64	6812.66	6838.37	6827.58

Table B3: Results for short jobs scenarios using clustering algorithm and drivers

	No Sharing		Sharing					
	n_vehicles	Cost	n_vehicles	n_drivers	Cost	Shuffle	ILS	LNS
c25S_1	4	688.95	-	-	-	-	-	-
c25S_2	4	639.23	-	-	-	-	-	-
c25S_3	5	781.67	-	-	-	-	-	-
c50S_1	8	1269.61	-	-	-	-	-	-
c50S_2	8	1120.89	-	-	-	-	-	-
c50S_3	8	1111.33	-	-	-	-	-	-
c100S_1	15	2246.81	-	-	-	-	-	-
c100S_2	14	1820.20	13	1	1863.09	1819.68	1833.18	1862.35
c150S_1	20	2569.39	19	1	2664.09	2656.54	2644.31	2660.52
c150S_2	23	3416.40	-	-	-	-	-	
nc25S	6	1281.62	5	0	1181.93	1180.88	1181.93	1177.61
nc50S	9	1636.97	-	-	-	-	-	-
nc100S	16	3086.69	-	-	-	-	-	-
nc150S	23	3817.21	-	-	-	-	-	-

Table B4: Results for base scenarios using CWS algorithm and drivers

	No Sharing		Sharing					
	n_vehicles	Cost	n_vehicles	n_drivers	Cost	Shuffle	ILS	LNS
c25_1	11	1576.93	4	4	947.95	938.29	941.39	905.61
c25_2	11	1382.81	4	4	902.40	846.89	840.81	824.19
c25_3	11	1534.64	8	3	1258.08	1250.27	1258.08	1248.66
c50_1	20	2529.78	10	7	1827.07	1752.33	1762.86	1727.85
c50_2	20	2115.55	12	6	1748.23	1669.21	1668.48	1658.08
c50_3	21	2168.86	14	5	2137.78	1997.89	2061.47	1953.89
c100_1	39	4376.10	21	11	3706.13	3619.12	3676.89	3550.14
c100_2	34	3097.94	22	9	3121.60	3030.34	3036.39	3016.56
c150_1	52	4768.55	27	15	4229.77	4130.23	4136.42	4057.16
c150_2	60	7225.45	33	20	5958.95	5759.08	5779.21	5661.16
nc25	10	1511.78	9	1	1580.79	1528.90	1580.79	1528.90
nc50	20	2510.72	16	4	2545.67	2485.46	2488.90	2458.37
nc100	37	5177.85	29	6	5171.62	5158.92	5168.09	5113.85
nc150	57	6787.09	35	17	6742.83	6677.22	6711.94	6561.40

Table B5: Results for all day jobs scenarios using CWS algorithm and drivers

	No Sharing		Sharing					
	n_vehicles	Cost	n_vehicles	n_drivers	Cost	Shuffle	ILS	LNS
c25L_1	14	1983.44	7	3	1269.47	1260.66	1262.38	1215.23
c25L_2	13	1622.65	5	3	982.60	951.76	951.76	921.66
c25L_3	14	1760.33	7	3	1150.87	1137.15	1137.87	1102.03
c50L_1	25	3197.39	13	5	2117.61	2075.77	2088.87	2050.19
c50L_2	25	2561.96	15	6	1948.43	1914.79	1914.79	1902.09
c50L_3	25	2450.48	15	3	2181.17	2167.52	2167.52	2152.99
c100L_1	47	5253.75	26	9	4257.88	4147.19	4230.35	4133.01
c100L_2	41	3647.72	25	10	3258.65	3188.80	3209.62	3189.57
c150L_1	63	5447.52	32	15	4420.71	4259.75	4294.98	4197.73
c150L_2	71	8323.27	38	17	6278.01	6215.76	6220.36	6179.26
nc25L	12	1650.09	8	3	1702.60	1686.39	1686.39	1668.98
nc50L	24	2775.68	16	4	2714.37	2711.39	2713.14	2642.49
nc100L	45	5683.17	30	9	5471.48	5402.78	5413.67	5403.68
nc150L	69	7778.81	40	15	6788.59	6648.36	6686.48	6568.37

Table B6: Results for short jobs using CWS algorithm and drivers

	No Sharing		Sharing					
	n_vehicles	Cost	n_vehicles	n_drivers	Cost	Shuffle	ILS	LNS
c25S_1	4	638.59	3	1	640.22	640.22	640.22	640.22
c25S_2	4	601.02	-	-		-	-	-
c25S_3	5	717.05	-	-		-	-	-
c50S_1	8	1231.62	7	1	1218.31	1218.31	1212.55	1207.48
c50S_2	8	966.10	7	1	1011.19	1007.60	993.09	990.45
c50S_3	8	968.12	7	1	1034.51	1008.06	1029.57	1031.96
c100S_1	15	2155.71	-	-	-	-	-	-
c100S_2	13	1548.10	12	1	1651.92	1639.51	1647.08	1628.63
c150S_1	20	2209.78	18	1	2304.53	2304.53	2304.43	2304.53
c150S_2	22	3112.95	-	-	-	-	-	-
nc25S	5	1079.65	-	-	-	-	-	-
nc50S	8	1386.87	-	-	-	-	-	-
nc100S	15	2623.45	-	-	-	-	-	-
nc150S	22	3351.74	-	-	-	-	-	-

Table B7: Results for base scenarios using RCWS algorithm and drivers

	No Sharing		Sharing					
	n_vehicles	Cost	n_vehicles	n_drivers	Cost	Shuffle	ILS	LNS
c25_1	11	1566.75	4	4	972.82	909.87	921.67	921.67
c25_2	10	1295.77	4	4	866.74	829.84	824.88	824.88
c25_3	11	1533.58	8	3	1264.48	1253.50	1253.50	1253.50
c50_1	20	2502.67	9	7	1716.31	1675.50	1668.71	1668.71
c50_2	19	2074.79	14	4	1915.86	1857.69	1882.81	1882.81
c50_3	19	2009.01	14	3	1931.27	1900.21	1910.37	1910.37
c100_1	38	4277.76	24	10	4016.78	3958.97	3970.56	3970.56
c100_2	33	2984.00	21	9	2977.19	2953.54	2944.97	2944.97
c150_1	51	4678.99	29	13	4053.70	3954.65	3954.24	3954.24
c150_2	58	7000.28	35	15	5795.99	5675.97	5722.71	5722.71
nc25	10	1511.78	9	1	1580.79	1528.9	1580.79	1580.79
nc50	19	2451.85	17	2	2539.84	2513.40	2514.25	2514.25
nc100	35	4991.33	30	5	4919.53	4898.72	4900.27	4900.27
nc150	56	6640.86	39	15	6718.47	6682.16	6680.63	6680.63

Table B8: Results for all day jobs scenarios using RCWS algorithm and drivers

	No Sharing		Sharing					
	n_vehicles	Cost	n_vehicles	n_drivers	Cost	Shuffle	ILS	LNS
c25L_1	14	1983.44	7	3	1269.47	1260.66	1262.38	1262.38
c25L_2	12	1546.69	5	4	971.55	939.86	934.32	934.32
c25L_3	14	1760.33	7	3	1159.60	1143.84	1145.81	1145.81
c50L_1	24	3101.39	12	7	2040.96	1998.32	2020.05	2020.05
c50L_2	25	2553.23	15	6	2045.16	2018.26	2021.79	2021.79
c50L_3	24	2345.86	14	5	1959.64	1929.86	1935.95	1935.95
c100L_1	46	5071.28	28	10	4145.55	4103.07	4103.12	4103.12
c100L_2	40	3588.11	23	12	3249.52	3200.3	3197.20	3197.20
c150L_1	62	5346.91	32	16	4350.43	4174.81	4214.52	4214.52
c150L_2	71	8145.83	35	19	5843.59	5699.29	6023.82	6023.82
nc25L	12	1650.09	9	2	1699.08	1699.08	1699.08	1699.08
nc50L	23	2759.32	16	5	2685.30	2622.64	2633.58	2633.58
nc100L	44	5525.67	30	9	5313.30	5258.47	5267.76	5267.76
nc150L	68	7654.07	43	15	7022.77	6902.36	6695.39	6695.39

Table B9: Results for short jobs scenarios using RCWS algorithm and drivers

	No Sharing		Sharing					
	n_vehicles	Cost	n_vehicles	n_drivers	Cost	Shuffle	ILS	LNS
c25S_1	4	627.70	3	1	591.41	591.41	591.41	591.41
c25S_2	4	592.36	-	-	-	-	-	-
c25S_3	5	717.05	-	-	-	-	-	-
c50S_1	8	1215.97	7	1	1202.66	1202.66	1202.66	1202.66
c50S_2	7	963.13	-	-	-	-	-	-
c50S_3	8	959.00	7	1	984.40	984.40	981.61	981.61
c100S_1	15	2003.37	14	1	2102.38	2102.38	2094.73	2094.73
c100S_2	13	1521.99	12	1	1608.76	1584.4	1584.4	1584.40
c150S_1	19	2152.97	18	0	2275.95	2275.95	2275.95	2275.95
c150S_2	22	2968.71	21	0	3064.16	3064.16	3064.16	3064.16
nc25S	5	1028.70	-	-	-	-	-	-
nc50S	8	1314.17	-	-	-	-	-	-
nc100S	15	2545.80	14	1	2720.55	2720.55	2720.55	2720.55
nc150S	22	3242.66	21	1	3425.25	3425.25	3425.25	3425.25

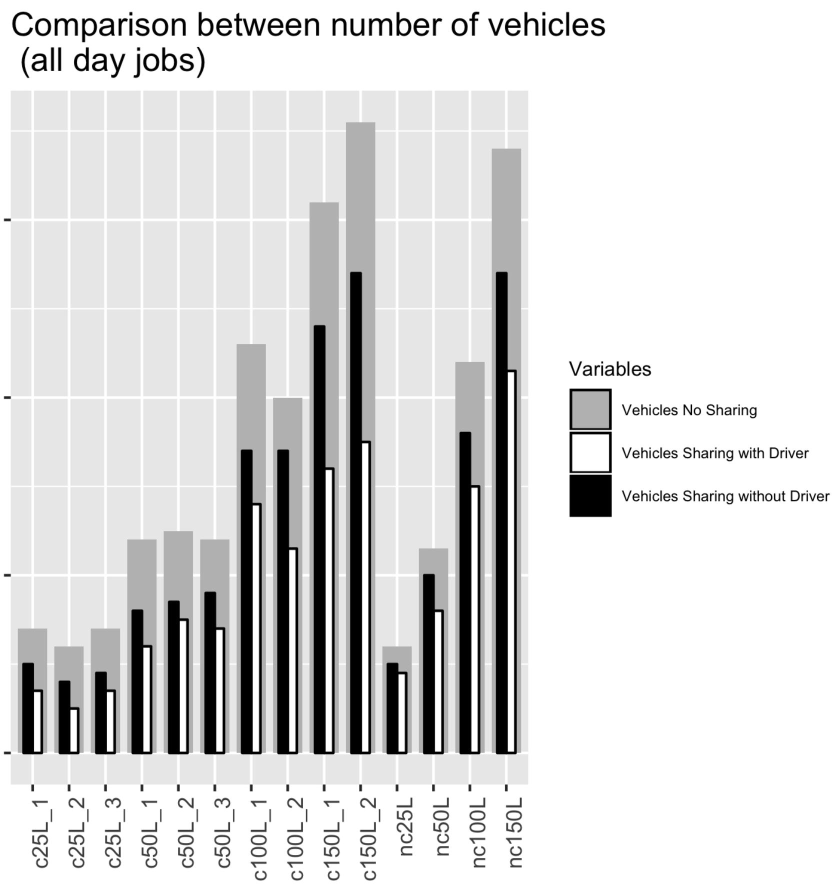

Figure B7: Comparison of the total number of vehicles between non-sharing, sharing without drivers, and sharing with drivers, for all day jobs.

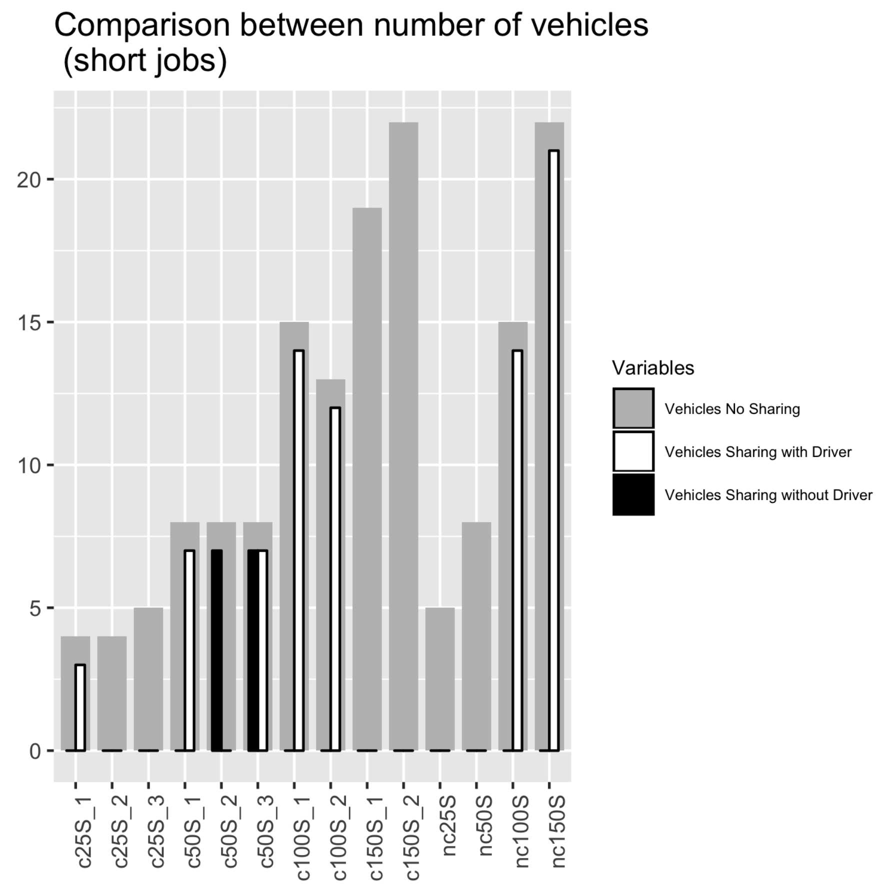

Figure B8: Comparison of the total number of vehicles between non-sharing, sharing without drivers, and sharing with drivers for short jobs.

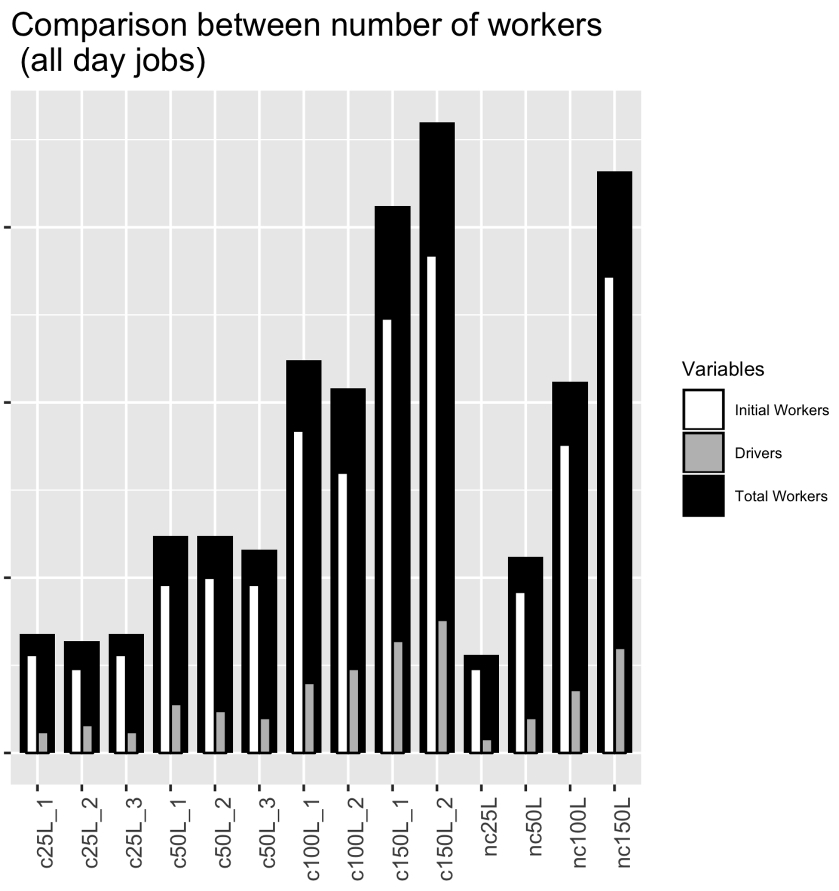

Figure B9: Comparison of the total number of employees (drivers plus workers assigned to jobs), the number of workers without sharing and the number of drivers, for all day jobs.

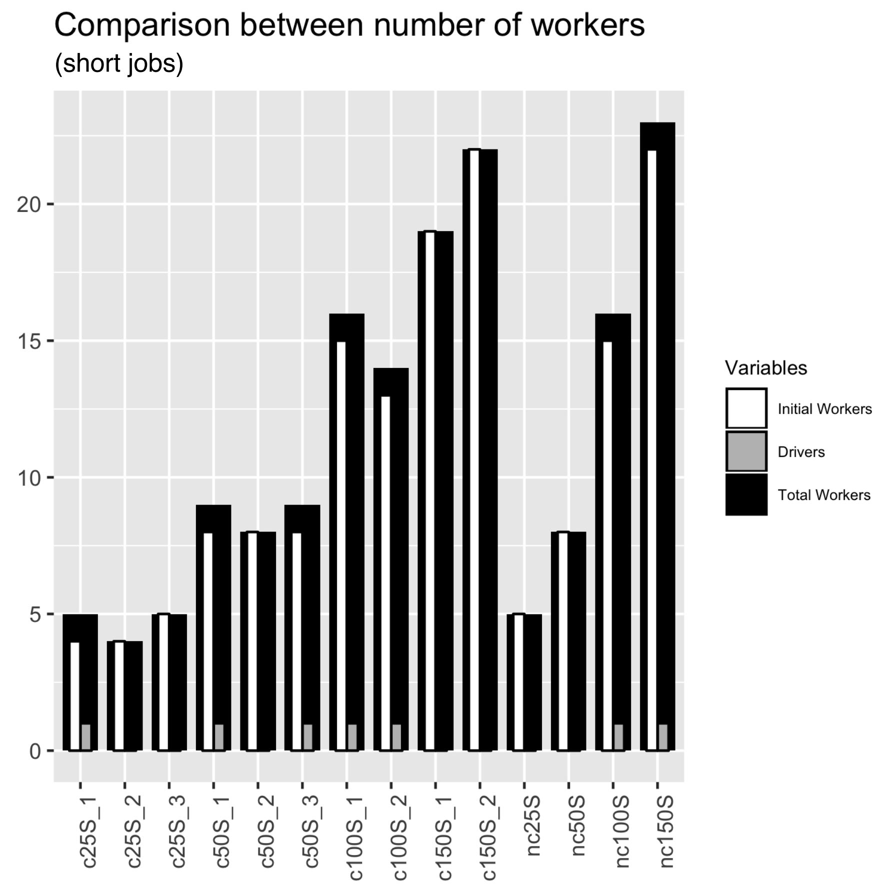

Figure B10: Comparison of the total number of employees (drivers plus workers assigned to jobs), the number of workers without sharing and the number of drivers, for short jobs

C Chapter 6 Appendix

Non-sharing Results

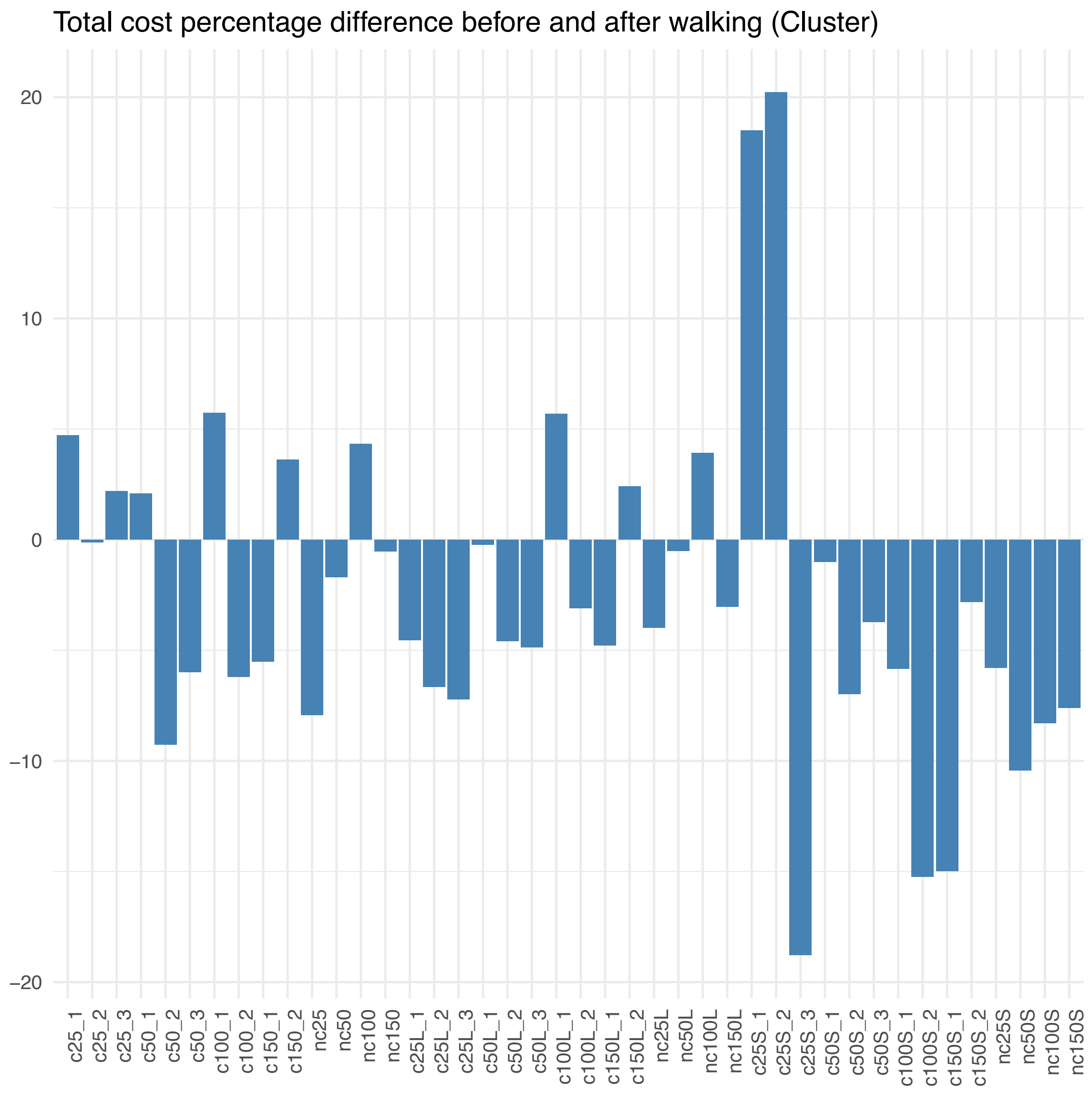

Figure C1: Percentage difference between the total distance cost using the cluster based approach without sharing and after sharing allowing to walking between jobs

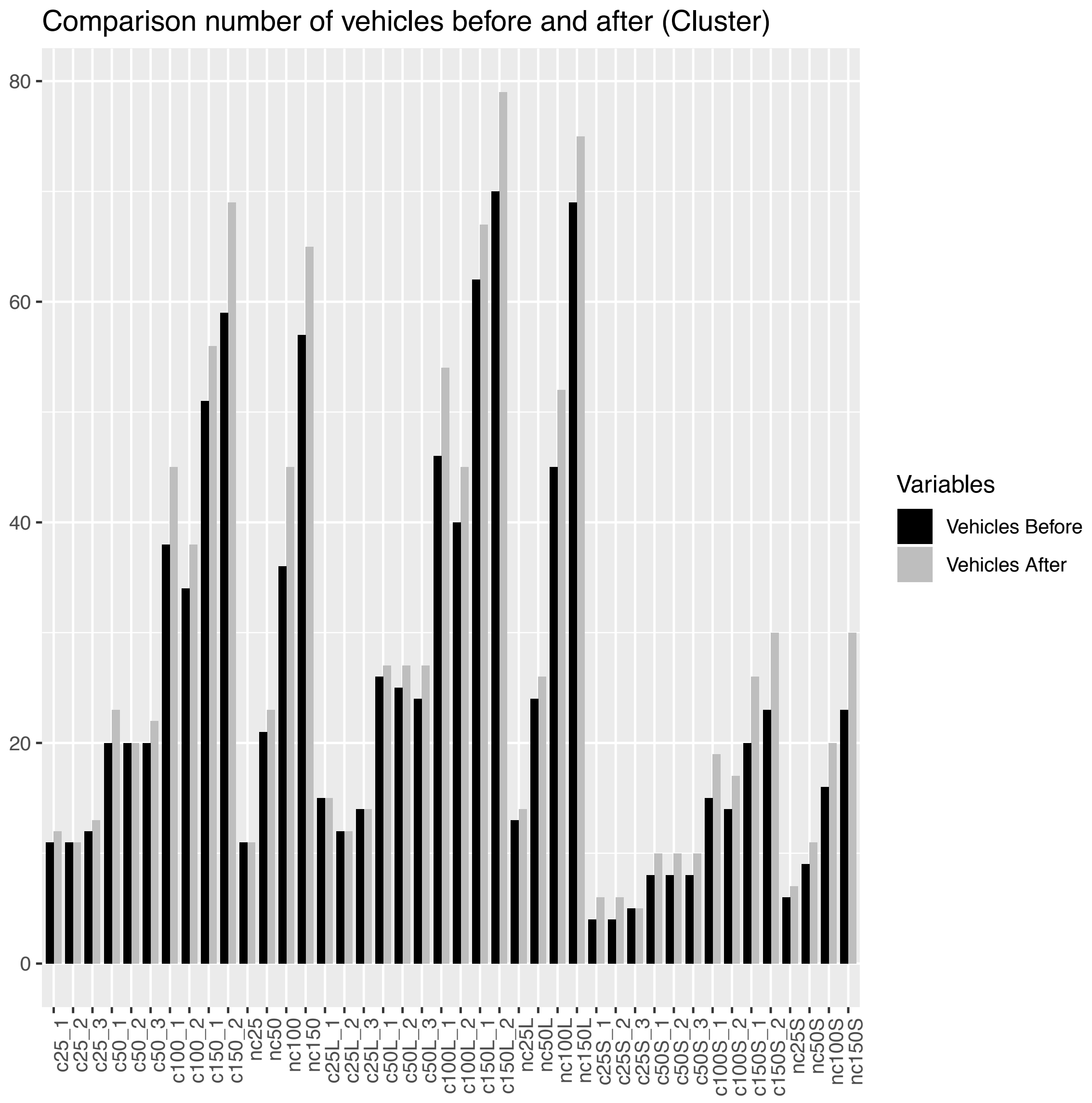

Figure C2: Number of vehicles used for the cluster based approach without sharing and after sharing allowing to walking between jobs

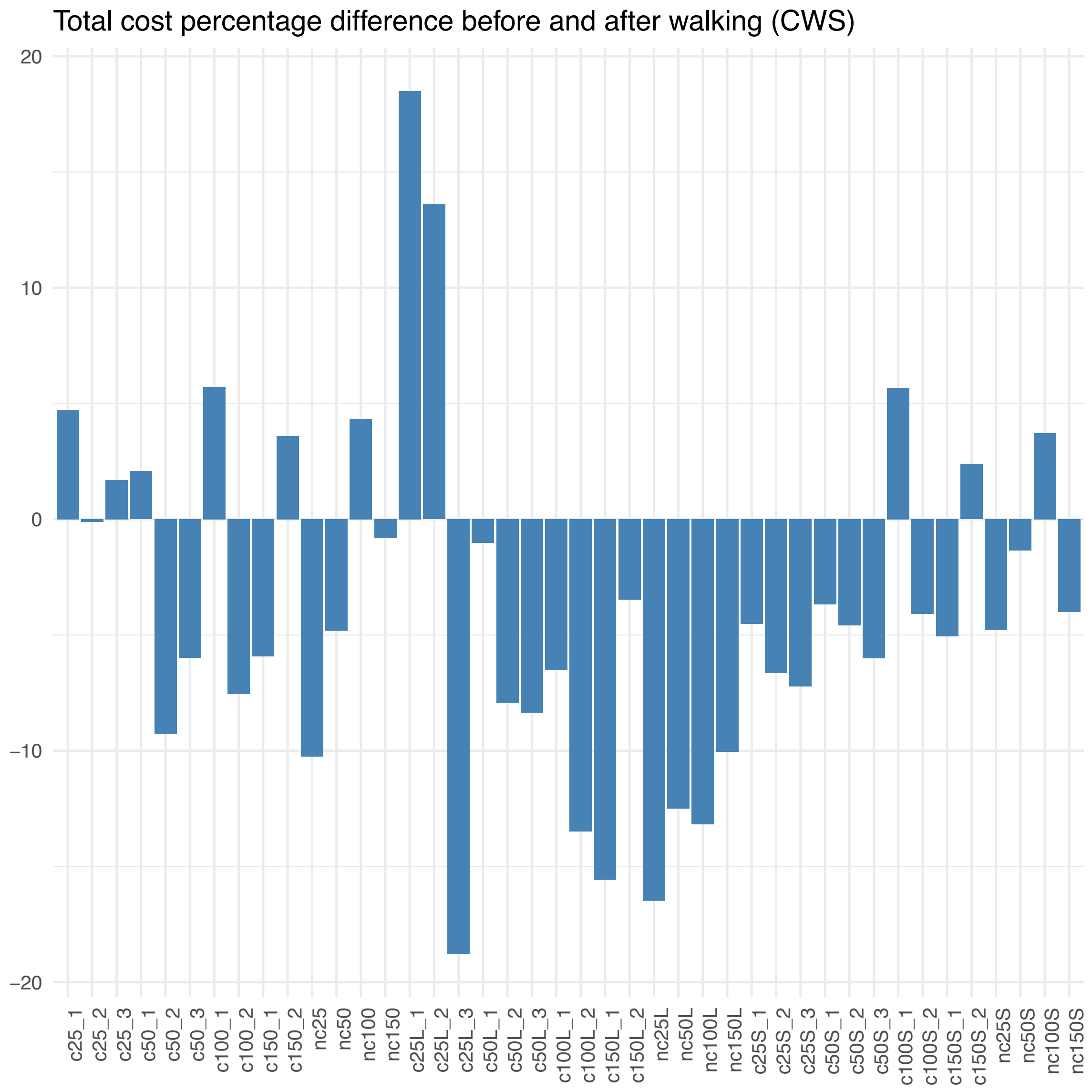

Figure C3: Percentage difference between the total distance cost using the CWS approach without sharing and after sharing allowing to walking between jobs

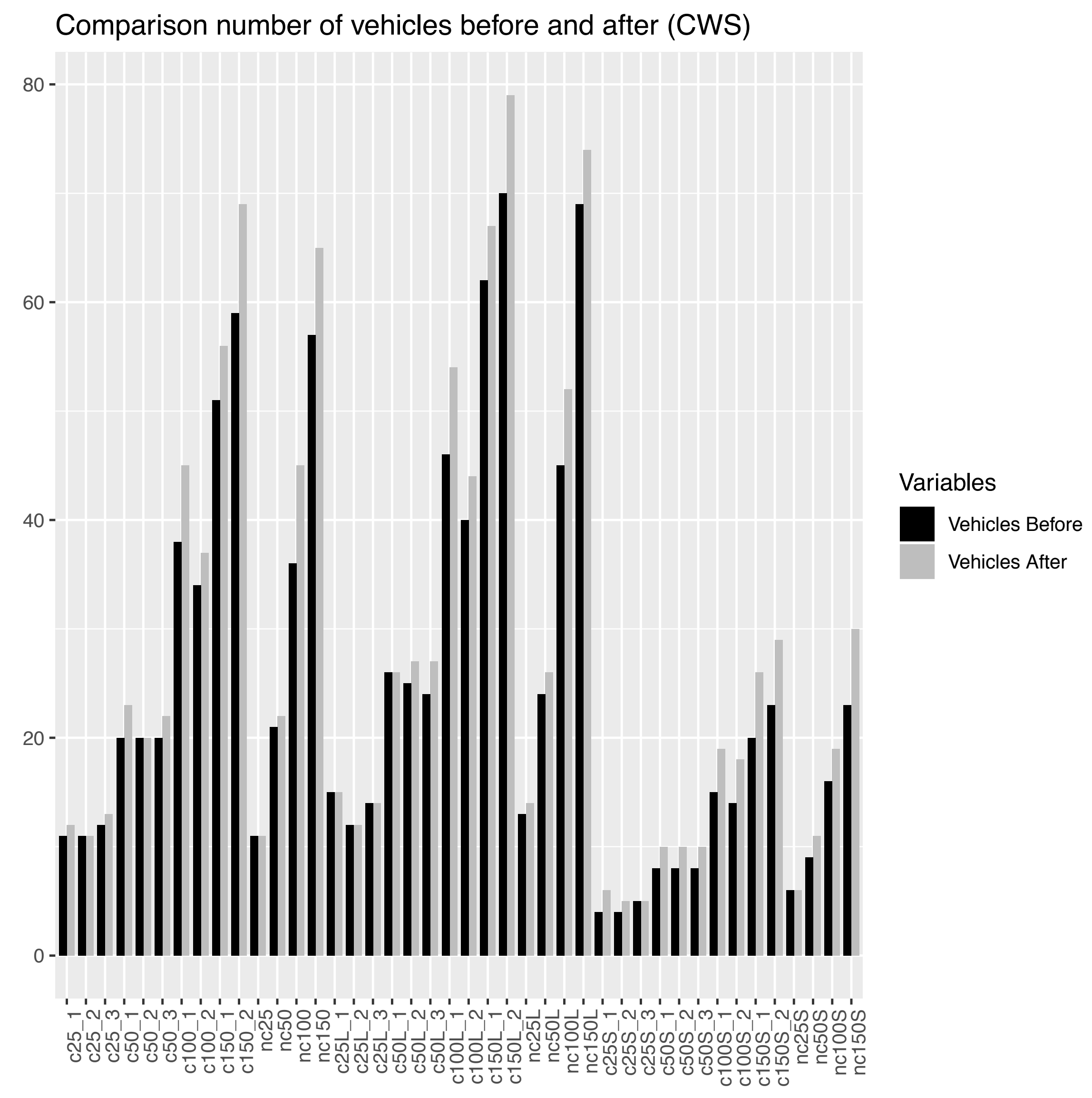

Figure C4: Number of vehicles used for the CWS approach without sharing and after sharing allowing to walking between jobs

Cluster results

Table C1: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the using the cluster algorithm with normal job times.

	nvehicles before	nvehicles after	BSB	BSA
c25_1	9	4	1399.23	604.93
c25_2	9	4	1234.51	579.76
c25_2	10	5	1376.44	857.28
c50_1	18	9	2475.67	1341.61
c50_2	19	11	2125.28	1316.66
c50_3	19	9	1905.45	1187.45
c100_1	36	17	4438.32	2388.74
c100_2	32	15	3228.95	1611.78
c150_1	-	23	-	2567.91
c150_2	57	28	7292.98	3804.71
nc25	10	8	1642.03	1544.72
nc50	20	12	2558.55	1883.56
nc100	-	19	-	3305.83
nc150	-	30	-	4623.65

Table C2: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the cluster algorithm with all day jobs.

	nvehicles before	nvehicles after	BSB	BSA
c25L_1	9	6	1404.61	874.14
c25L_2	8	5	1234.50	752.82
c25L_3	10	8	1582.79	1100.80
c50L_1	17	11	2360.07	1650.83
c50L_2	19	11	2296.57	1409.49
c50L_3	17	11	2104.65	1376.83
c100L_1	35	21	4595.81	2875.55
c100L_2	34	16	3353.06	1803.92
c150L_1	48	26	4851.23	2730.66
c150L_2	57	29	7327.14	3901.71
nc25L	10	8	1750.19	1498.73
nc50L	19	14	2516.95	2284.12
nc100L	36	21	5353.23	3469.92
nc150L	57	34	6995.81	5016.25

Table C3: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the cluster algorithm with short job times.

	nvehicles before	nvehicles after	BSB	BSA
c25S_1	-	2	-	349.94
c25S_2	-	4	-	520.94
c25S_3	-	-	-	-
c50S_1	-	5	-	734.23
c50S_2	-	8	-	1044.96
c50S_3	-	7	-	1012.64
c100S_1	-	10	-	1534.66
c100S_2	-	11	-	1424.15
c150S_1	-	11	-	1391.49
c150S_2	-	13	-	2036.24
nc25S	5	6	1181.93	1231.21
nc50S	-	9	-	1429.16
nc100S	-	15	-	2493.04
nc150S	-	16	-	2812.93

Sharing without Drivers

Clarke and Wright results

Table C4: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the CWS algorithm with normal job times.

	nvehicles before	nvehicles after	BSB	BSA
c25_1	10	4	1352.13	604.93
c25_2	9	4	1213.88	579.76
c25_2	10	5	1423.36	838.36
c50_1	18	9	2336.44	1341.61
c50_2	19	11	2115.06	1316.66
c50_3	19	9	2059.36	1187.45
c100_1	37	17	4268.05	2388.74
c100_2	33	15	3096.39	1689.75
c150_1	51	23	4779.03	2477.82
c150_2	58	27	6984.73	3688.79
nc25	-	8	-	1472.17
nc50	-	13	-	1999.64
nc100	36	19	5173.15	3305.83
nc150	-	29	-	4588.51

Table C5: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the CWS algorithm with long job times.

	nvehicles before	nvehicles after	BSB	BSA
c25L_1	10	6	1518.60	874.14
c25L_2	7	5	1073.39	752.82
c25L_3	9	8	1248.63	1100.80
c50L_1	17	11	2368.80	1644.62
c50L_2	17	11	2078.59	1259.62
c50L_3	18	10	2141.44	1213.79
c100L_1	37	21	4356.16	2862.7
c100L_2	32	17	3220.20	1913.89
c150L_1	48	27	4784.02	2790.04
c150L_2	55	30	7039.28	3900.74
nc25L	10	8	1607.96	1519.11
nc50L	20	14	2578.22	2098.64
nc100L	35	21	5208.51	3424.22
nc150L	53	34	6697.07	5011.06

Table C6: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the CWS algorithm with short job times.

	nvehicles before	nvehicles after	BSB	BSA
c25S_1	-	2	-	349.94
c25S_2	-	4	-	689.95
c25S_3	-	-	-	-
c50S_1	-	5	-	734.23
c50S_2	7	7	971.73	963.69
c50S_3	-	8	-	1017.68
c100S_1	-	11	-	1614.78
c100S_2	-	9	-	1313.04
c150S_1	19	11	2216.93	1391.49
c150S_2	-	14	-	2150.43
nc25S	-	5	-	1075.35
nc50S	-	9	-	1395.21
nc100S	-	15	-	2392.68
nc150S	-	17	-	2858.67

Sharing using dedicated drivers

Cluster results

Table C7: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the cluster algorithm with normal job times.

	nvehicles before	nvehicles after	ndrivers before	ndrivers after	BSB	BSA
c25_1	4	4	4	2	858.77	635.01
c25_2	5	4	4	1	941.58	582.05
c25_3	5	5	5	1	1107.02	868.11
c50_1	11	8	5	3	1947.46	1556.98
c50_2	12	8	6	5	1712.57	1227.62
c50_3	16	8	2	3	1978.65	1179.42
c100_1	29	17	6	6	4216.86	2583.86
c100_2	23	14	8	3	3015.62	1672.13
c150_1	26	24	15	6	3983.89	2746.00
c150_2	45	25	10	12	6626.14	3736.06
nc25	9	8	2	2	1638.25	1664.95
nc50	14	9	6	5	2648.94	2088.85
nc100	30	18	6	6	5022.13	3591.56
nc150	38	29	14	8	6735.85	4829.38

Table C8: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the cluster algorithm with all day job times.

	nvehicles before	nvehicles after	ndrivers before	ndrivers after	BSB	BSA
c25L_1	6	6	4	1	1070.30	904.93
c25L_2	6	5	3	1	1058.37	785.98
c25L_3	8	7	2	2	1442.87	1048.16
c50L_1	12	11	5	3	1970.70	1713.55
c50L_2	12	10	7	2	1905.95	1437.77
c50L_3	14	11	5	2	2210.90	1406.62
c100L_1	29	20	11	5	4127.65	3046.21
c100L_2	26	17	9	6	3317.43	2101.77
c150L_1	34	26	16	6	4487.21	2939.58
c150L_2	41	29	17	8	6433.06	4072.08
nc25L	9	8	2	2	1819.28	1730.93
nc50L	15	11	5	5	2582.31	2263.16
nc100L	31	22	10	5	5288.14	3943.54
nc150L	42	32	16	6	6812.66	5212.17

Table C9: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the cluster algorithm with short job times.

	nvehicles before	nvehicles after	ndrivers before	ndrivers after	BSB	BSA
c25S_1	-	2	-	0	-	357.73
c25S_2	-	4	-	0	-	521.88
c25S_3	-	-	-	-	-	-
c50S_1	-	5	-	0	0	736.62
c50S_2	-	8	-	0	0	1044.96
c50S_3	-	7	-	0	0	1012.64
c100S_1	-	10	-	0	0	1534.66
c100S_2	13	10	1	1	1819.68	1445.52
c150S_1	19	11	1	2	2644.31	1633.81
c150S_2	-	12	-	3	-	2020.74
nc25S	5	6	0	1	1177.61	1294.13
nc50S	-	9	-	0	-	1430.12
nc100S	-	14	-	3	-	2668.92
nc150S	-	16	-	1	-	2957.07

Clarke and Wright results

Table C10: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the CWS with normal job times.

	nvehicles before	nvehicles after	ndrivers before	ndrivers after	BSB	BSA
c25_1	4	4	4	2	905.61	635.01
c25_2	4	4	4	1	824.19	582.05
c25_3	8	5	3	1	1248.66	849.19
c50_1	10	8	7	3	1727.85	1556.98
c50_2	12	8	6	5	1658.08	1227.62
c50_3	14	8	5	3	1953.89	1179.42
c100_1	21	17	11	6	3550.14	2583.86
c100_2	22	14	9	6	3016.56	1736.61
c150_1	27	24	15	5	4057.16	2713.49
c150_2	33	26	20	11	5661.16	3820.35
nc25	9	7	1	2	1528.90	1622.54
nc50	16	10	4	4	2458.37	2121.03
nc100	29	18	6	6	5113.85	3591.56
nc150	35	29	17	6	6561.40	4895.56

Table C11: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the CWS with all day job times.

	nvehicles before	nvehicles after	ndrivers before	ndrivers after	BSB	BSA
c25L_1	7	6	3	1	1215.23	904.93
c25L_2	5	5	3	1	921.66	785.98
c25L_3	7	7	3	2	1102.03	1048.16
c50L_1	13	11	5	3	2050.19	1699.99
c50L_2	15	10	6	2	1902.09	1437.77
c50L_3	15	10	3	2	2152.99	1253.03
c100L_1	26	20	9	5	4133.01	3023.52
c100L_2	25	16	10	7	3188.80	2004.96
c150L_1	32	27	15	6	4197.73	3012.40
c150L_2	38	30	17	7	6179.26	4076.17
nc25L	8	8	3	2	1668.98	1765.80
nc50L	16	13	4	3	2642.49	2296.58
nc100L	30	21	9	7	5402.78	4018.48
nc150L	40	32	15	5	6568.37	5134.08

Table C12: Results comparing the number of vehicles, drivers and total distance, before and after applying the clustering pre-process using the CWS with short job times.

	nvehicles before	nvehicles after	ndrivers before	ndrivers after	BSB	BSA
c25S_1	3	2	1	0	640.22	357.73
c25S_2	-	4	-	0	-	689.95
c25S_3	-	-	-	-	-	-
c50S_1	7	5	1	0	1207.48	736.62
c50S_2	7	7	1	0	990.45	963.69
c50S_3	7	8	1	0	1008.06	1024.48
c100S_1	-	11	-	-	-	1614.78
c100S_2	12	9	1	0	1628.63	1313.04
c150S_1	18	12	1	1	2304.43	1556.6
c150S_2	-	13	-	3	-	2143.84
nc25S	-	5	-	0	-	1106.39
nc50S	-	8	-	1	-	1434.12
nc100S	-	13	-	4	-	2545.36
nc150S	-	15	-	3	-	2850.91

Bibliography

2016. URL `https://www.gov.uk/green-taxes-and-reliefs/overview`.

Tomtom traffic index. `https://www.tomtom.com/en_gb/trafficindex`, 2018. Accessed: 10-05-2018.

Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223 (2):295–303, 2012.

Niels A H Agatz, Alan L. Erera, Martin W P Savelsbergh, and Xing Wang. Dynamic ride-sharing: A simulation study in metro Atlanta. *Transportation Research Part B: Methodological*, 45 ‘(9):1450–1464, 2011.

Agostinho Agra, Marielle Christiansen, Alexandrino Delgado, and Lars Magnus Hvattum. A maritime inventory routing problem with stochastic sailing and port times. *Computers & Operations Research*, 61:18–30, 2015.

Mahdi Alinaghian and Nadia Shokouhi. Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. *Omega*, 76: 85–99, 2018.

Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, pages 462–467, 2017.

David L Applegate, Robert E Bixby, Vasek Chvátal, and William J Cook. The Traveling Salesman Problem: A Computational Study. *Princeton University Press*, page 593, 2006.

Ph Augerat, JM Belenguer, E Benavent, A Corberán, D Naddef, and G Rinaldi. *Computational results with a branch and cut code for the capacitated vehicle routing problem.* IMAG, 1995.

Philippe Augerat, José M Belenguer, Enrique Benavent, Angel Corbéran, and Denis Naddef. Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106(2):546–557, 1998.

Roberto Baldacci, Eleni Hadjiconstantinou, and Aristide Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations research*, 52(5):723–738, 2004.

Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6, 2012.

Matthew Barth and Michael Todd. Simulation model performance analysis of a multiple station shared vehicle system. *Transportation Research Part C: Emerging Technologies*, 7(4):237–259, 1999.

John R Birge and Francois Louveaux. *Introduction to stochastic programming.* Springer Science & Business Media, 2011.

Burak Boyaci, Konstantinos G. Zografos, and Nikolas Geroliminis. An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational Research*, 240(3):718–733, 2015.

David Bredström and Mikael Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European journal of operational research*, 191(1):19–31, 2008.

Jan Brinkmann, Marlin W Ulmer, and Dirk C Mattfeld. Inventory routing for bike sharing systems. 2015.

Jose Caceres-Cruz, Pol Arias, Daniel Guimarans, Daniel Riera, and Angel A Juan. Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR)*, 47(2): 32, 2015.

Nicos Christofides, Mingozzi A, Paolo Toth, and C Sandi. The vehicle routing ptoblem. In Nicos Christofides, editor, *Combinatorial Optimization*, page 315 338. Wiley, Chichester, New York, 1979.

Nicos Christofides, Aristide Mingozzi, and Paolo Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282, 1981.

G. Clarke and J. W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581, 1964.

William Cook. Applications of the tsp, 2012.

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

Jean François Cordeau and Gilbert Laporte. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *4or*, 1(2):89–101, 2003.

Jean François Cordeau and Gilbert Laporte. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

Gerald Senarclens De Grancy and Marc Reimann. Evaluating two new heuristics for constructing customer clusters in a vrptw with multiple service workers. *Central European Journal of Operations Research*, 23(2):479–500, 2015.

Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.

A. Di Febbraro, E. Gattorna, and N. Sacco. Optimization of Dynamic Ridesharing Systems. *Transportation Research Record: Journal of the Transportation Research Board*, 2359(December 2013):44–50, 2013.

Pedro D'Orey and Michel Ferreira. Can ride-sharing become attractive? A case study of taxi-sharing employing a simulation modelling approach. *IET Intelligent Transport Systems*, 9(September 2013):1–19, 2014.

Michael Drexl. Synchronization in vehicle routinga survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

Pierre-François Dutot, Alexandre Laugier, and Anne-Marie Bustos. Technicians and interventions scheduling for telecommunications. *France Telecom R&D*, 2006.

Tomás Eiró, José M Viegas, and Luis M Martínez. A new optimisation procedure to design minibus services: an application for the lisbon metropolitan area. *Procedia-Social and Behavioral Sciences*, 20:856–865, 2011.

Daniel J Fagnant and Kara M Kockelman. The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transportation Research Part C: Emerging Technologies*, 40:1–13, 2014.

Daniel J Fagnant and Kara M Kockelman. Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas. *Transportation*, 45(1): 143–158, 2018.

Sebastian Fahnenschreiber, Felix G??ndling, Mohammad H. Keyhani, and Mathias Schnee. A Multi-modal Routing Approach Combining Dynamic Ride-sharing and Public Transport. *Transportation Research Procedia*, 13:176–183, 2016.

Christian Fikar and Patrick Hirsch. A matheuristic for routing real-world home service transport systems facilitating walking. *Journal of Cleaner Production*, 105:300–310, 2015.

Christian Fikar, Angel A. Juan, Enoc Martinez, and Patrick Hirsch. A discrete-event driven metaheuristic for dynamic home service routing with synchronised trip sharing. *European J. of Industrial Engineering*, 10(3):323, 2016.

Gerd Finke, Armin Claus, and Eldon Gunn. A two-commodity network flow approach to the traveling salesman problem. *Congressus Numerantium*, 41(1):167–178, 1984.

Matteo Fischetti, Juan José Salazar González, and Paolo Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

Marshall L Fisher and Ramchandran Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

Richard Freling, Dennis Huisman, and Albert PM Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6(1):63–85, 2003.

Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006.

Masabumi Furuhata, Maged Dessouky, Fernando Ordóñez, Marc Etienne Brunet, Xiaoqing Wang, and Sven Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.

Xavier Gabaix and Yannis M Ioannides. The evolution of city size distributions. In *Handbook of regional and urban economics*, volume 4, pages 2341–2378. Elsevier, 2004.

Michael R Garey and David S Johnson. Computers and intractability: a guide to np-completeness, 1979.

Michel Gendreau, Alain Hertz, and Gilbert Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40 (6):1086–1094, 1992.

Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290, 1994.

Arthur M Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972.

Arthur M Geoffrion. Lagrangean relaxation for integer programming. In *Approaches to integer programming*, pages 82–114. Springer, 1974.

Supriyo Ghosh, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Dynamic Redeployment to Counter Congestion or Starvation in Vehicle Sharing Systems. pages 230–231, 2015.

Knut Haase, Guy Desaulniers, and Jacques Desrosiers. Simultaneous vehicle and crew scheduling in urban mass transit systems. *Transportation Science*, 35(3):286–303, 2001.

Mondher Hachicha, M John Hodgson, Gilbert Laporte, and Frédéric Semet. Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, 27 (1):29–42, 2000.

Stuart L Hart et al. Beyond greening: strategies for a sustainable world. *Harvard business review*, 75(1):66–77, 1997.

Hideki Hashimoto, Mutsunori Yagiura, and Toshihide Ibaraki. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2):434–456, 2008.

Wesam Herbawi and Michael Weber. A Genetic and Insertion Heuristic Algorithm for Solving the Dynamic Ridematching Problem with Time Windows. *Gecco 2012*, pages 385–392, 2012.

Hadi Hosni, Joe Naoum-Sawaya, and Hassan Artail. The shared-taxi problem: Formulation and solution methods. *Transportation Research Part B: Methodological*, 70:303–318, 2014.

Yan Huang, Ruoming Jin, Favyen Bastani, and Xs Wang. Large Scale Real-time Ridesharing with Service Guarantee on Road Networks. *arXiv preprint*, 7(14): 2017–2028, 2013.

Diana Jorge, Goncalo H A Correia, and Cynthia Barnhart. Comparing optimal relocation operations with simulated relocation policies in one-way carsharing systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1667–1675, 2014.

Diana Jorge, Cynthia Barnhart, and Goncalo Homem de Almeida Correia. Assessing the viability of enabling a round-trip carsharing system to accept one-way trips: Application to Logan Airport in Boston. *Transportation Research Part C: Emerging Technologies*, 56:359–372, 2015.

A. A. Juan, J. Faulin, J. Jorba, D. Riera, D. Masip, and B. Barrios. On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright savings heuristics. *Journal of the Operational Research Society*, 62(6): 1085–1097, 2011.

Brian Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35(7):2307–2330, 2008.

Richard Katzev. Car Sharing: A New Approach to Urban Transportation Problems. *Analyses of Social Issues and Public Policy*, 3(1):65–86, 2003.

Zhifeng Lang, Enjian Yao, Weisong Hu, and Zheng Pan. A vehicle routing problem solution considering alternative stop points. *Procedia-Social and Behavioral Sciences*, 138:584–591, 2014.

Gilbert Laporte, Martin Desrochers, and Yves Nobert. Two exact algorithms for the distanceconstrained vehicle routing problem. *Networks*, 14(1):161–172, 1984. ISSN 10970037.

Gilbert Laporte, Yves Nobert, and Martin Desrochers. Optimal routing under capacity and distance restrictions. *Operations research*, 33(5):1050–1073, 1985.

EL Lawer, Jan Karel Lenstra, AH Rinnooy Kan, and DB Shmoys. The traveling salesman problem. *Chinchester: Wiley*, 1985.

Alan Lee and Martin Savelsbergh. Dynamic ridesharing: Is there a role for dedicated drivers? *Transportation Research Part B: Methodological*, 81:483–497, 2015.

Jan Karel Lenstra and AHG Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

Canhong Lin, King Lun Choy, George TS Ho, Sai Ho Chung, and HY Lam. Survey of green vehicle routing problem: past and future trends. *Expert Systems with Applications*, 41(4):1118–1138, 2014.

Yeqian Lin, Wenquan Li, Feng Qiu, and He Xu. Research on Optimization of Vehicle Routing Problem for Ride-sharing Taxi. *Procedia - Social and Behavioral Sciences*, 43:494–502, 2012.

Wei Liu, Hai Yang, and Yafeng Yin. Efficiency of a highway use reservation system for morning commute. *Transportation Research Part C: Emerging Technologies*, 56:293–308, 2015.

Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

Dennis Luxen and Christian Vetter. Real-time routing with openstreetmap data. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 513–516. ACM, 2011.

Jens Lysgaard, Adam N Letchford, and Richard W Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing service. *Proceedings - International Conference on Data Engineering*, pages 410–421, 2013.

Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations.* John Wiley & Sons, Inc., 1990.

Luis M Martinez, Gonçalo H A Correia, and José M Viegas. An agent-based simulation model to assess the impacts of introducing a shared-taxi system: An application to Lisbon (Portugal). *Journal of Advanced Transportation*, 49(3):475–495, 2015.

Rens Meijkamp. Changing consumer behaviour through eco-efficient services: an empirical study of car sharing in the netherlands. *Business Strategy and the Environment*, 7(4):234–244, 1998.

Mehdi Nourinejad, Sirui Zhu, Sina Bahrami, and Matthew J. Roorda. Vehicle relocation and staff rebalancing in one-way carsharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 81:98–113, 2015.

Ibrahim Hassan Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, 41(4):421–451, 1993.

King-Wah Pang and Jiyin Liu. An integrated model for ship routing with transshipment and berth allocation. *IIE transactions*, 46(12):1357–1370, 2014.

Junhyuk Park and Byung-In Kim. The school bus routing problem: A review. *European Journal of operational research*, 202(2):311–319, 2010.

Victor Pillac, Christelle Guéret, and Andrés L Medaglia. An event-driven optimization framework for dynamic vehicle routing. *Decision Support Systems*, 54(1): 414–423, 2012.

Ted K Ralphs. Parallel branch and cut for capacitated vehicle routing. *Parallel Computing*, 29(5):607–629, 2003.

Gerhard Reinelt. *The traveling salesman: computational solutions for TSP applications*. Springer-Verlag, 1994.

Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, and Carlo Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37):13290–13294, 2014.

Michael Schneider, Andreas Stenger, Fabian Schwahn, and Daniele Vigo. Territory-based vehicle routing in the presence of time-window constraints. *Transportation Science*, 49(4):732–751, 2014.

Gerald Senarclens de Grancy and Marc Reimann. Evaluating two new heuristics for constructing customer clusters in a VRPTW with multiple service workers. *Central European Journal of Operations Research*, 23(2):479–500, 2015.

Susan A Shaheen, Nelson D Chan, and Helen Micheaux. One-way carsharings evolution and operator perspectives from the americas. *Transportation*, 42(3):519–536, 2015.

Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98*, pages 417–431. Springer, 1998.

M. M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2):254–265, 1987.

Jin-Hwa Song and Kevin C Furman. A maritime inventory routing problem: Practical approach. *Computers & Operations Research*, 40(3):657–665, 2013.

Kevin Spieser, Samitha Samaranayake, and Emilio Frazzoli. Vehicle routing for shared-mobility systems with time-varying demand. In *2016 American Control Conference (ACC)*, pages 796–802. IEEE, 2016.

Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review*, 91:190–207, 2016.

R. Tachet, O. Sagarra, P. Santi, G. Resta, M. Szell, S. H. Strogatz, and C. Ratti. Scaling Law of Urban Ride Sharing. *Scientific Reports*, 7:42868, 2017.

Éric D Taillard. A heuristic column generation method for the heterogeneous fleet vrp. *RAIRO-Operations Research*, 33(1):1–14, 1999.

Timm Teubner and Christoph M. Flath. The Economics of Multi-Hop Ride Sharing: Creating New Mobility Networks Through IS. *Business and Information Systems Engineering*, 57(5):311–324, 2015.

Paolo Toth and Daniele Vigo. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1-3):487–512, 2002a.

Paolo Toth and Daniele Vigo. *The vehicle routing problem.* SIAM, 2002b.

Kentaro Uesugi, Naoto Mukai, and Toyohide Watanabe. Optimization of vehicle assignment for car sharing system. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 1105–1111. Springer, 2007.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21, 2013.

Simone Weikl and Klaus Bogenberger. Relocation strategies and algorithms for free-floating car sharing systems. *IEEE Intelligent Transportation Systems Magazine*, 5(4):100–111, 2013.