Master Thesis

Optimisation of a demand-responsive transit system

Author : Corentin Pirson

Supervisors : Mario Cools and Quentin Louveaux

Master thesis submitted for the degree of

MSc in Physical Engineering

Academic year 2016-2017
Remerciements

Je tiens tout d’abord à remercier sincèrement mes promoteurs Messieurs Quentin Louveaux et Mario Cools qui m’ont guidé et encouragé dans l’accomplissement de ce travail tout au long de l’année. Je remercie également Thibault Cuvelier de m’avoir fait découvrir l’optimisation discrète. Ses conseils dans les premiers pas de ce projet ont été précieux.

Merci à Madame Sabine Limbourg et Monsieur Bertrand Cornélusse d’avoir accepté de faire partie du jury de ce travail de fin d’études. Je leur souhaite une agréable lecture.

Je désire tout particulièrement remercier Kevin Bulthuis pour la relecture et les conseils avisés qu’il m’a donnés pour parachever ce mémoire. Son aide dans ce cadre ne représente qu’une infime partie de ce qu’il a pu m’apporter durant 10 années de discussions multidisciplinaires dans les transports en commun.

Mes remerciements vont enfin à ma famille qui a largement contribué à ce que mes études et ce mémoire puissent être menés à bien dans des conditions optimales.
Optimisation of a demand-responsive transit system

Pirson Corentin – Master in physical engineering – 2016-2017

The development of technologies such as GPS, real time operating smartphone applications and autonomous vehicles in a relatively near future allow to think about ways of reinventing public transport. A way to make it more flexible is to replace classical bus lines by a fleet of public vehicles that would systematically adapt their route to serve clients. This work aims at studying such a system from an optimisation point of view. The objective is, given origin and destination points of the clients, to optimise the routes of the different vehicles of the fleet. The optimisation is based on the satisfaction of the clients, i.e. on the minimisation of the time spent waiting for a vehicle and inside this vehicle. The equity of the service among the different clients is also taken into account. It is established in this work that the problem can be modelled as a Vehicle Routing Problem for which the objective function is adapted. An algorithm based on the simulated annealing metaheuristic method is developed in order to optimise the routes of the vehicles under different hypotheses. The model is applied in several situations (general trips demand in a large area and peak hour situation in an idealised city) in order to assess the performances of the system. The system appears to be more efficient in high demand-density areas such as the centres of big cities. For more general travel demand, the combination of such a system with more classical mass transport systems should be considered.

Key-words: transport planning, discrete optimisation, Vehicle Routing Problem, simulated annealing.
# Contents

Table of Contents ................................................................. 1

1 Introduction ............................................................................... 3
  1.1 Related work ........................................................................ 4
  1.2 Problems to be solved ......................................................... 6
    1.2.1 Demand: where and when clients want to go .................... 7
    1.2.2 Geography: definition of time costs and saturation of roads .. 8
    1.2.3 Objective: definition of the optimisation criteria ............... 8
    1.2.4 Fleet: characteristics of the available vehicles ................. 9
    1.2.5 Limiting the number of taxi stop points ......................... 9
    1.2.6 Structure in the network ............................................... 9

2 Mathematical formulation of the optimisation problems ............... 11
  2.1 Vehicles with one passenger capacity (car sharing) ............... 11
    2.1.1 Problem seen as a task scheduling problem with several machines .. 12
    2.1.2 Inspiration from the Vehicle Programming Problem ............ 14
  2.2 Problem seen as a variation of the classical VRP (the MinAvg and the MinMax problems) ..................................................... 17
  2.3 Vehicles with capacity of more than one passenger (ride sharing) .. 20
  2.4 Adding release times ....................................................... 20
  2.5 Towards more fairness ..................................................... 21

3 Methodology of resolution ...................................................... 23
  3.1 Local search and metaheuristics ......................................... 23
    3.1.1 Solution space and local search ..................................... 23
    3.1.2 Metaheuristics .......................................................... 25
  3.2 Simulated annealing ......................................................... 25
    3.2.1 Principle .................................................................. 25
    3.2.2 The cooling schedule ............................................... 25
    3.2.3 Summary ............................................................... 26
  3.3 Application to the combined MinSum and MinMax VRP ............ 26
    3.3.1 Car sharing problem: local search for a combined MinSum-Minmax VRP variation ......................................................... 26
    3.3.2 Ride sharing problem with maximum arrival times .............. 31
    3.3.3 Ride sharing with maximum relative arrival time ................ 33
    3.3.4 Multi-passengers vehicles with release times .................... 34
  3.4 Performances of the method .............................................. 34
    3.4.1 Computation times .................................................... 34
    3.4.2 Quality of the solutions .............................................. 37
  3.5 Avenues of improvement .................................................... 38
    3.5.1 Ameliorations for the simulated annealing process ............ 40
    3.5.2 Pre-processing of the data (clustering) ............................ 41
4 Results analysis

4.1 Solved problems definitions and notations ........................................ 44
4.2 Influence of the weight factors $W$ .................................................. 46
   4.2.1 Evolution of the maximum waiting time with $W$ (Problem 1) ......... 46
   4.2.2 Penalty factor of Problem 4 ....................................................... 47
   4.2.3 Quality of service as a function of the distance from the center ....... 48
4.3 Influence of the demand density on the quality of the service. Modelling with
   problems 2 and 4 .............................................................................. 50
   4.3.1 Influence of the density of the demand ....................................... 50
   4.3.2 Other considerations ................................................................. 50
4.4 Quality of the service as a function of the available fleet ..................... 52

5 Conclusions and perspectives ................................................................. 55
Chapter 1

Introduction

Nowadays, in order to contribute to the reduction of the number of vehicles on the roads and the subsequent problems of congestion inside the cities and on the main axes, citizen have mainly two alternatives to the use of their individual car to achieve their daily displacements: classical public transports (bus, train, tramway and underground) and car sharing. Except in very specific cases (living and working next to train stations linked by direct lines, displacements inside a city along well deserved axes, same place and hours of work as a neighbour), the practice of these alternatives requires the user to be flexible and to adapt departure and arrival times consequently. The time devoted to commuting can be significantly increased.

These flexibility requirements reflect a mismatch between the transport offer and demand, especially for specific journeys that are performed by a limited number of people. The simple example of the bus service of countryside areas (except school services) is self speaking: only a few buses serve the area each day, and only a very low number of people can be flexible enough to use them. This situation leads to economic and ecological aberrations.

There is thus a huge interest in working at a more refined adaptation of such transportation systems to the demand in order to make them more efficient on the user point of view, but also more sustainable on economic and ecological aspects. The development of the GPS technology in the past decades as well as systematisation of fast and easy communication between users and transport networks more recently (real time operating smartphone applications,...) allow to think about ways of reinventing public transport. The upcoming technology of autonomous cars reinforces the idea of the possibility of an optimised and demand-based transport network.

This thesis focuses on a system that unifies the features of public transport, as it will be composed of a fleet of public vehicles (e.g. taxis or autonomous cars) and of car sharing, as these vehicles are expected to take along their routes several users from their own starting points and to their destinations. The satisfaction of the users will be used as a depart criterion for determining possible operation of the system, and the economical and environmental impact of such a system will be deduced and studied in a second time. It is clear that if the circulation conditions are kept as they are, the use of such a system cannot be advantageous in terms of travel times. However, one of the objective of such a system, as a public transport system, is to lower the number of vehicles on the roads. This is expected to decrease travel times due to congestion.

In this work, the fact that the mentioned vehicles are autonomous or not is not crucial: the focus is set on the planning of the routes they follow. However, in subsequent works, the difference between classical taxis and autonomous vehicles could be taken into account, the autonomous cars allowing to predict travel times more precisely (if the circulation conditions are known) and to avoid the drivers mandatory rest times.

The goal of this thesis is to investigate the ways to optimise the behaviour of such a system of shared self driving vehicles. An optimal behaviour signifies a routing of the available vehicles which satisfies quality of service criteria as well as possible. The research for such a routing can
be conducted on the basis of specifications of the individual demand of the clients (origin and
destination point, hours,...) known in advance, typically for home - workplace commuting. Ob-
taining good routings or good solutions of the problem is the priority of this work. Nevertheless,
as the real advantage of the aforementioned technologies lies in their high potential for real-time
applications, attention will also be devoted to the fast obtaining of solutions throughout this
work.

The content of this thesis is divided as follows. In the remainder of this introductory chapter,
we look at the main conclusions drawn by some recently-conducted theoretical studies on the use
of a fleet of shared self driving cars in European cities. We then look at the different problems -
and their related hypotheses - that one can address in the framework of the optimisation of the
routes of a such a fleet in order to minimise the losses of times incurred by the users of such a
collective transport system with respect to the use of an individual vehicle.

Chapter 2 is devoted to the formal modelling of the optimisation problems to be solved.
The formulations are inspired from classical problems of discrete optimisation. Variants of the
Vehicle Routing Problem are namely studied in order to address the problems described in
Chapter 1.

In Chapter 3, we present the methods used to solve these variations of the Vehicle Routing
Problem. The simulated annealing meta heuristic method is used in the hope of obtaining good
solutions, as close as possible to the optima. The technical aspects taken into account in order to
code the method in a way that keeps computational costs as low as possible are also described.
Finally, we present approaches to pre-process the data (grouping clients in clusters in order to
cut the main problem into smaller ones) in order to render the computations faster.

Chapter 4 presents the solutions to different problems as well as their analyse. When the
hypothesis make allow it, these results are compared with the conclusions of the studies described
in Chapter 1. The analyses include the observation of the quality of the service as a function
of the number vehicles and their size and the disparities between the different users regarding
their geographical situation. Both real data and artificial simple structure of demand randomly
generated will be used.

The conclusions of these investigations are gathered in Chapter 5, together with prospectives
to enhance the techniques used in this work, and to integrate the obtained tools in a practical
shared self-driven vehicles system.

1.1 Related work

Several recent studies have already focused on the simulation of the behaviour of a fleet of public
taxicabs and their influence on mobility. A recent case study on the city of Berlin [13] brought
several conclusions from the spatio-temporal analysis of the performances of such a service of
autonomous taxis. This study was conducted with the following hypothesis taking as a main
hypotheze that the demand of trips with personal car within the city is totally supplied by a
fleet of autonomous taxis. This amounts to a demand of 2.5 millions of trips to be provided
by a fleet of 100 000 taxis. The ratio number of taxis/number of trips is thus equal to 4 %. Besides,
the rest of the traffic, i.e. trips from and towards the interior of the city performed
by classical private car and public transport is taken into account in the real time simulation.
Concerning the way of assigning routes to the shared cars, a dispatching centers analyses the
situation in real time and either assigns the closest taxi to a customer (in time of oversupply)
or the closest customer to an empty vehicle (in times of undersupply). There is no possibility
for advance booking: the whole demand is treated in real time, as the trip requirements appear.
Concerning the unused vehicles, it is supposed that they can park between two services. The
system is expected to appear to the clients at least as efficient as a private car: the waiting times
for an autonomous vehicle cannot be much longer than the time required to park and unpark
a private car, and no detour is allowed: a vehicle serves different clients sequentially. We talk
in this case of car sharing. In order to validate the obtained routing, the 95 percentile waiting time should not overcome a given threshold.

The methodology of this study consists thus in a real time simulation following the aforementioned rules, that are locally applied by all the vehicles of the system thanks to a central dispatching system gathering all the information about the positions of the clients and vehicles in time. The advantage of such a simulation is that it allows to take into account the different physical aspects of traffic, to represent for example congestion and allow to compute alternative routes.

We present now some of the main conclusions derived from these simulations. The aforementioned 100 000 taxis succeed in satisfying the demand for 2.5 millions daily trips inside the city with average waiting times reaching between 4 and 5 minutes during the peak periods, and does not exceed 2 minutes in the rest of the time. The 95 percentile, however, reaches 14 minutes during the peak hours and 7 minutes during the off hours. There is also a spatial distribution of the waiting times: the clients of the periphery encounter on a regular basis longer waiting times than the others. During the morning peak, the waiting times in some outskirts zones can reach 20 minutes against less than 100 seconds in central zones.

The large number of clients served with a few vehicles allows to liberate huge amounts of space initially devoted to parking. The counterpart is an increase of the traffic due to empty rides. In the center of the city, these empty rides are however limited (10 %), but they become particularly significant in the outskirts (45 %). 13 % of the driven distances are achieved by empty vehicles. By contrast with the private cars that are considered as an underused asset, the vehicles are occupied on average during 6.8 hours a day (and are on the road during 7.6 hours).

All in all, the system as it operates in this study seems to be efficient only in the city center. This fact was not a discovery from these simulations, as [6] et [7] already arrived to the conclusion that such systems of autonomous vehicles are more efficient in high density population areas. Then, the authors suggest that if the autonomous taxis system should operate only in such limited areas, for example with hub parkings installed around the city in order to perform the change from and to the private car.

Except during short periods within peak hours, the simulations show that there is always a significant amount of idle taxis. An intelligent handling of these taxis would allow to manage better time and space-concentrated demands for example from the peripheral hub parkings.

About the size of the demand taken into account, it has been noticed that when performing the simulation for 10 % of this demand with 11 % of the 100 000 taxis, very similar results are obtained.

Another study, conducted by the OECD [1], with a case study based on Lisbon this time, went further by distinguishing shared vehicles that drive clients sequentially from their origin point to their destination point without any (such vehicles are referred as Auto Vots which constitute a car sharing system) and vehicles allowed to pick up several people on a same journey (referred as Taxi Vots, constituting a ride sharing system). Like the precedent research, the investigations of the OECD are based on the hypothesis that there should not be any flexibility from the passengers point of view, and that their demand should be exactly satisfied. The goal was to observe the influence of the use of Taxi Vots or Auto Vots, with or without the presence of high-capacity public transport on the required fleet size, the volume of traffic and the required parking areas. High-capacity public transport or high-capacity transit is characterised by carrying a larger volume of passengers using larger vehicles and/or more frequent service than a standard fixed route bus system. The main result of the study also based on real-time simulations, is a significant reduction of the required number of required cars. The number of cars could be divided by 5 in the case of the use of an AutoVots system without high-capacity public transport (during the peak hours, the number of required cars is reduced by 23 %), and by 10 using a TaxiVots system together with a high-capacity public transport network (reduction of 65% for peak hours).
In the former case, in term of travelled car-km, one observe an augmentation of 6%. This augmentation is principally due to the fact that the taxis need to replace the buses, which are not a part of the high-capacity public transports. As expected, this augmentation is by far much higher in the second case and reaches 89%. Beside the service that could have been performed by the high-capacity public transport, a significant part of the kilometres are travelled by empty vehicles going to their next client origin point or repositioning for waiting for a new demand.

The real-time operation of the system is, as in the case of Berlin, based on a central dispatching system, taking into account for the routing not only the waiting times, but also the detours the passengers of Taxi Vots have to encounter.

1.2 Problems to be solved

The objectives of the present study differ from the aforementioned ones in several aspects.

First of all, rather than performing a real time simulation with fixed rules for the system (e.g. a free vehicle always picks up the nearest client), we aim at designing a system that can act intelligently with a broader vision of the set of demands. For example, in this system, for choosing which client a free vehicle will serve next, the position and the demand of the other clients that could be served subsequently are taken into account. In other words, the goal is, given a set of demands, to optimise the routes of the different vehicles of the fleet according to objective functions that will be discussed in the following sections. These objective functions will involve the waiting / arrival times, which are thus not seen as constraints. The idea is thus to solve several optimisation problems with different values of the parameters in order to see for which of their values, the objectives are reached.

Secondly, the idea is to see how such a system can manage not only the demand inside a city, but also in the rural areas, between city centres,... . The demand that we should be able to treat is very general. The quality of the service in the suburban areas will be one of the principal concerns of the study.

Treating an a priori given demand and looking for the methods that can lead to the best solution possible may be seen as a work taking place in the framework of purely academic research. However, such good solutions, if found, could be ultimately concretely implemented, for example in the case of home-work commuting, which are typically trips that can be known in advance. As a matter of fact, studies have shown that a significant part of the population would be ready to communicate about its precise agenda for the use of an demand-based public transport system: a conclusion from an experiment in Belgium is that the proportion rises 47.54 percent of the citizens [17]. Furthermore, 63.43 % of the population would accept that the vehicle transporting them makes a detour if it brings benefits to the community. A reasonable detour should not overcome 25 % of the initial journey duration. Having the possibility of computing routes fast could also lead to the possibility of a system that optimises routes taking into account real-time incoming data.

For the optimisation, the adopted transport policy defines the criteria on which the research of the best routes (i.e. the solution) is be based. In this work, it is typically the minimisation of the loss of time the users of the system face compared to a situation where they would simply drive from their starting point to their destination in their private car. These losses arise from the time they need to wait for a vehicle to take them in charge, but also from deviations a vehicle could use in order to serve other clients at the same time. In more structured networks of such taxis, it is also possible that a client loses time because of a change of vehicle (e.g. people get off different cars at a given point and get in a single bus).
The optimisation process that this work aims at designing should be used in the way represented in Figure 1.1.

This optimisation process can be constructed by considering different problems and aspects. The remainder of this section is devoted to the enumeration and description of the challenges that can come to the mind of someone designing such a process, but only a few of them will be taken throughout the rest of this report.

1.2.1 Demand: where and when clients want to go

Each journey request of a user of the system is characterised by an origin and a destination point. The system should also be able to take into account the time by which a client wants to start travelling and/or the time before which he requires to be arrived, referred as release times later in the text. In this report, two cases are studied.

- **A simultaneous demand.** This situation consists in the worst case in terms of congestion and amount of required vehicles because all the clients want to travel at the same time. However, it is more simple to model and it is interesting to see how many vehicles would be required in such a situation. This situation can be linked to a peak period.

- **Clients leaving at different times (release times).** This situation is more realistic than the previous one: it is considered that the clients cannot be taken in charge before they are ready to leave their origin point, i.e. before their corresponding release time. This situation could potentially allow to better notice the advantages of the system, with possibilities that all the clients served by a same car get a similar level of satisfaction, despite some arrive at destination earlier than the others.

Note that we consider here that the agenda of the user are known in advance, i.e. there is no update of the data. The optimisation process is able to impose the initial positions of the vehicles (for example at random origin points of clients) or to let them as an unknown of the problem. The set of demands that will be used in this work is a sample of about 800 real trips performed in Flanders. They all correspond to the first trips of these 800 people in the day. These data originate from a survey conducted by Cornelis et al. [10]. Artificially generated sets of demand will also be used in order to put certain aspects in evidence.
1.2.2 Geography: definition of time costs and saturation of roads

As mentioned in Figure 1.1, it is necessary to specify information about the time it takes to travel between all the interest points (origin and destination points of the clients, departure points of the vehicles, ...). This amounts to describe a graph of which these points are the vertices, and the routes used to link them are the edges. This information can be gathered in what will be called the cost matrix \( C \), such that \( C_{i,j} \) is the time required to go from node \( i \) to node \( j \). Ideally, such costs are pre-computed using a route planner in order to take speed limits and roads geometries into account.

In the present report, \( C_{i,j} \) will be computed as via the distance as the crow flies. It is assumed that the vehicles travel at constant speed along straight lines linking two adjacent points of their route. Except when explicitly mentioned (clustering methods based on demand orientation and point-straight line distance, ...), the use of such a definition of time cost, despite it largely deforms the reality of the demand, does not involve any loss of generality, and it is possible, in a further use of the optimisation process, to use a \( C \)-matrix that better reflects reality. The only requirement on the \( C \)-matrix to keep a physical sense is that the time costs respect the triangular inequality, i.e. that going from a point \( A \) to a point \( B \) via a point \( C \) takes is always at least as expensive as going directly from \( A \) to \( B \).

Throughout this work, our network of public transport will be considered as completely independent of its environment. This means that congestion will not be taken into account in the model. In other words, the capacity of the roads is considered as infinite. The time costs of passing on a edge is thus constant in time (it does not increase around peak hours because of the higher density of private cars) and it does not depend on the number public vehicles density. The latter simplification allows to keep the problem linear: if the flow rate of fully charged public vehicles going from a point \( A \) to point \( B \) is multiplied by \( n \), the time required to transport a same number of clients from \( A \) to \( B \) will simply be divided by \( n \).

1.2.3 Objective: definition of the optimisation criteria

As it has been mentioned earlier, the optimisation of the routes of the vehicles is based on the efficiency of the system on a client point of view. The goal of the system is that they all arrive at destination as fast as possible. In the case of a simultaneous demand, the objective is to minimise the arrival times, while in the case involving release times, the objective is to minimise the time gap between the release time and the arrival time. In the remainder of this paragraph, we will use the term "arrival time" for both the situations. Classically, two approaches are used to deal with this kind of problems: either the objective is to minimise the average arrival times or the objective is to minimise the maximum arrival time. Both approaches however present drawbacks. If minimising the mean arrival time, it can happen that people with isolated origin and destination points are systematically served after the other.

Therefore, in this study, the goal is to minimise both quantities at the same time, and to study the difference in the resulting routes when modifying the importance credited to each aspect.

The minimisation of the maximum arrival can sometimes lead to very unfair results. As a matter of fact, vehicles will tend to serve clients with isolated origin and/or destination points before the others even if these clients need to travel very long distance, to the detriment of other more centred clients who often have demands for shorter journeys. From this observation, the idea of minimising the relative arrival times (i.e. the ratio between the arrival time) instead of simple arrival times arises. A version of the optimisation process is implemented in which both the average arrival times and the maximum relative arrival time are minimised, in order to avoid the aforementioned unfairness.
1.2.4 Fleet: characteristics of the available vehicles

The optimisation process represented in Figure 1.1 takes as a parameter the size of the vehicle fleet as well as their capacity, i.e. the number of clients that can occupy a vehicle at the same time. This number is the same for the whole fleet of vehicles taken into account by the process. Two main cases will be studied:

- **Fleet of single-place vehicles (Car sharing):** Using only vehicles with a capacity of only one passenger trivially leads to an increased number of travelled vehicle kilometres, and thus also to congestion. It is however interesting to see the number of cars that such a system would require, and to see by how much it could reduce the problematic of parking in urban areas. As it will be explained in the next chapter, the solutions of such a problem reduced to an ordered list of clients to serve, which renders the problem easier to model and solve.

- **Fleet of shared vehicles (Ride sharing):** The most interesting version of the problem consist in the planning of routes of vehicles that pick up and drop people in any order provided that they respect the vehicles capacity. A vehicle could for example pick up two people at different points, drop one a it further, pick up another one before dropping the two remaining ones. This allow to study whether the capacity of the vehicles has a real impact on the number of travelled kilometres amongst other things.

1.2.5 Limiting the number of taxi stop points

Instead of considering as many pairs of origin points and destination points as clients, on can consider gathering them at a limited number of taxi stops. The locations of such stops can be easily and optimally chosen as origin or destination points of some clients. This amounts to solve the well known facility location problem\(^1\).

Such an approach could have an impact on the methodology used, and thus the performance of the optimisation process on one hand, and on the quality of solutions on the other hand. As a matter of fact, a solution would be given by lists of points to be visited for each vehicle, but the number of points would be reduced, which tends to simplify the problem. A new unknown however, would be a list of the clients that are taken in charge by each vehicles at this point. Such an approach may induce a supplementary cost for the client which needs to travel by himself to and from the stop points, but one can observe in the subsequent results if there is a significant benefice in terms of efficiency of the system. The implementation of this specific technique is beyond the scope of this work.

1.2.6 Structure in the network

- **A jungle of cars:** In a first approach, one can simply develop the optimisation process as a solver of one single large problem. However, as it will be depicted in the following sections, it is impossible for such a problem to find the optimal solution. (Meta)-heuristics are thus used in the hope to obtain solutions even better when more time is devoted to the computations. The required time grows with the size of the problem. This one-problem based approach allows to search for a solution among all the imaginable possibilities, and does not take into account the potential structure of the demand (for example, one can expect that many people travel from a city (e.g. Aalst) to another (e.g. Antwerpen). The knowledge of such a structure could be used to get better solutions and/or to save computational time.

\(^1\)For a review of the Facility Location Problem, see [23].
• **Grouping travellers (clustering):** Better, or at least faster-produced solutions could be obtained by grouping the clients with respect to their origins and destinations (and their time of departure). In other words, this approach relies on the determination of sets passengers that could possibly travel together in a good solution. For example, clients starting and arriving in the same areas and in the same times could be part of a same group, but more sophisticated criteria for grouping the clients will also be tested in this study. The "jungle of cars" problem can then be solved with a limited number of vehicles for the small number of clients that compose the group, either in order to have cars that picks them up in the origin area and then drive them the destination area, or to have such cars driving to and from a bus stop in these two areas respectively (see figure 1.2). Such a decoupling of the different groups allows to save computation time by solving the different sub-problems in parallel, or simply by the fact that the time required to solve such problems increases more than linearly with their size. In the present work, clustering methods will be presented. They are expected to orient the solutions towards routes appearing as wise. The aforementioned combination of cars and buses is not a part of this work.

• **Picking people along the road:** Beyond this structure of the journeys, one can consider the people that are not part of a large cluster could be simply be taken by a bus or a car passing in their area when it is interesting. In the present work, this aspect will be only taken into account in the framework of the clustering of the clients: it is interesting to allow a client to travel with other ones if the origin and/or the destination point of this first client is situated along the trajectories of the other ones.

![Figure 1.2](image_url)

Figure 1.2: A possible use of the optimisation process in the case of the framework of a car-bus system: ideally, the bus should be able to pick up on its way a client isolated between the departure and the arrival zone.
Chapter 2

Mathematical formulation of the optimisation problems

In this Chapter, we present some of the aforementioned problems on a more formal way. Mathematical formulations are shown and links with generic classical optimisation problems are made. These mathematical formulations always correspond to Mixed Integer Linear Problems. The objective functions that will be presented, as well as the related constraints, are thus always linear combinations of the variables. Such models can thus theoretically be treated with common linear discrete optimisation solvers such as Cplex or Gurobi. These softwares use the branch and bound algorithm which is designed to find the best feasible solution according to the objective. This approach will unfortunately only work in the case of very small problems. The solutions of these small problems can be used as a comparison with the approximation method that will be developed in the next chapter.

As it has been explained earlier, the objective functions used in the different variants of the problem are always two-fold: the objective is to minimise a combination of both a quantity averaged among the set of clients (which amounts to minimise this quantity summed over the set of clients) and the maximum value of a quantity. Therefore, the objective function will every time be presented under the form

$$\sum_{i\in\{1,...,n_p\}} p_i + W \max_{i\in\{1,...,n_p\}} q_i,$$  \hspace{1cm} (2.1)

where $n_p$ denotes the total number of clients (or the total number of requests) taken into account by the system, and $p_i$ and $q_i$ both represent a given quantity attributed to the client $i$, which can be the departure time, the relative arrival time,... The $W$ factor, which has to be carefully chosen, illustrates the importance that is accorded to the minimisation of $\max_{i\in\{1,...,n_p\}} q_i$ with respect to $\sum_{i\in\{1,...,n_p\}} p_i$. In the remainder of this chapter devoted to the formulation of problems as MILP, some of the objective function presented will have to be written in a more sophisticated way, but turn then back to simple forms as (2.1) in the subsequent chapters.

Note that excepted if it is mentioned, we always consider the cases where all the clients want to leave their origin point at the same time.

2.1 Vehicles with one passenger capacity (car sharing)

The problem of the single place vehicles has the advantage to be easy to model. In this case, the clients are served one after another and the goal is simply to establish the best ordered lists of clients possible for each vehicle. The problem can thus be seen for example as a task scheduling problem with several machines or as a variant of the vehicle routing problem. The basics about task scheduling modelling can be found in [5].

We introduce the following notations used for the specification of the data of this problem:
• $n_p$ is the number of clients to be served, which is assumed to correspond to the number of required individual trips to perform.
• $n_v$ is the number of vehicles that are supposed to serve the demand.
• $dO_{j,i}$ is the time required to travel between the depot point of vehicle $j$ and the origin point of client $i$.
• $DO_{i,l}$ is the time required to travel between the destination point of client $i$ and the origin point of client $l$.
• $OD_i$ is the time required to travel between the origin and the destination points of client $i$. It corresponds thus to commuting time that would be observed with a private car.

2.1.1 Problem seen as a task scheduling problem with several machines

For this problem, each client can be assimilated to a task to be attributed to one of the available vehicles, which are assimilated to machines that can perform tasks sequentially. The following formulation is inspired from the classical the Modeles

Let $P$ denote the set of clients ($|P| = n_p$) and $D$ the set of vehicles ($|D| = n_v$).

The variables used to model this problem are the following for $i, l \in P$, $j \in D$ and $k \in \{1, ..., n_p\}$:

• $x_{i,j,k}$, which is equal to 1 if client $i$ is served by vehicle $j$ in $k^{th}$ position and which is equal to 0 in any other case
• $t_{j,k}$, the time that the $k^{th}$ passenger of car $j$ needs to wait before being picked up by a vehicle.
• $startmax$: the maximum waiting time.
• $z_{i,j,k-1,l}$: the product of the binary variables $x_{i,j,k-1}$ and $x_{l,j,k}$ for $k \geq 2$, so that $z_{i,j,k,l}$ is equal to 1 iff client $i$ is the $k^{th}$ on the list of vehicle $j$ and client $l$ is the $k + 1^{th}$ on the same list.

The objective function and the constraints corresponding to this model are shown in Formulation 2.1
The objective function is given by (2.2), where $W$ is the weight factor which specifies how much we care about the maximum waiting time (trying to avoid that people far from the city centers are too much disadvantaged).

The constraints (2.3) express the time the first passenger taken by any vehicle $j$ needs to wait should be greater or equal to the time required for the vehicle that serves him to go from its starting point to the origin point of this client. In practice, because of the objective function that minimises starting times, this constraint will always be satisfied to equality. The equation (2.4) has a similar purpose, but for the subsequent passengers: the first term is the time the previous passenger of the car has been waiting before being taken, the term in $OD_i$ accounts for the trip duration of this previous passenger, and the term in $DO_{i,l}$ accounts for the time needed to go from the destination point of the previous passenger to the origin point of the considered passenger. The term in $M$ vanishes if the car $j$ takes at least $k$ passengers and is equal to $-M$ in the other cases. $M$ is chosen large enough to guarantee that in the latter case, the right hand side of (2.4) is lower than zero, so that the variable $t_{j,k}$ is automatically set to zero to indicate that this $k^{th}$ passenger of car $j$ -who does not exist- has no contribution to the objective function.
The $M$-value should however not be too large in order to avoid complications during the linear relaxation process\(^1\). The waiting times are non negative values as specified in (2.5). Constraints (2.6) ensure that every passengers travels one and only one time, and constraints (2.7) specifies that each possible trip $(j, k)$ corresponds to no more than one client. The constraints (2.8) state that if a car does not take a $k^{th}$ passenger, it will not take any passenger of superior rank. Constraints (2.10) to (2.12) ensure that $z_{i,j,k-1}$ is the product of the binary variables $x_{i,j,k-1}$ and $x_{i,j,k}$ for $k \geq 2$.

This formulation gives an idea of the complexity of the problem and of the number of related variables ($n^2p^2$ for the $z$ variables only). In the rest on the report, only the VRP approaches, which are more convenient, will be used.

**2.1.2 Inspiration from the Vehicle Programming Problem**

Instead of seeing the services to the different clients as tasks, it is tempting to use a representation of the problem that emphasizes more the geographical aspect. A very classical problem of discrete optimisation that comes to the mind is the Vehicle Routing Problem, or VRP. The VRP is the generalisation of the Traveling Salesman Problem, or TSP with several vehicles are available: the goal is to find the routes of vehicles (in general the number of used vehicles can be a part of the problem) starting from one or several depots in order to visit a set of geographically scattered points referred as cities with a minimum total cost (e.g. using a minimum amount of fuel or minimising the sum of the travel times). The cities can be modelled as the nodes of a graph, the routes used to link the cities being represented by the edges. The cost of the use of an edge going from city $i$ to city $j$ is represented by a the cost matrix element $C_{i,j}$. The total cost to be minimised is classically defined as the sum of the costs of the edges that have been used by the vehicles.

For the car sharing problem, the definition of the cost matrix has to be modified. In the case of taxis with on passenger capacity, the approach used does not consist in considering each origin or destination point as a city, but rather to assimilate each client to one city.

Note that the capacity of the vehicles in the framework of the VRP (which is linked to the number of cities that a vehicle can visit because of restrictions on the amount of merchandise it can transport to deliver them for example), is different from the capacity of the taxis of our problem, which is in this case of one client.

Our problem is thus similar to a VRP with the following specificities

1. The distance matrix is not symmetric: the lapse of time between arriving at the client $A$ origin point and then at the client $B$ origin point is different from the time required to serve these clients in the opposite order. One can define the cost for going from client $i$ to client $j$ as $C_{i,j} = OD_i + DO_{i,j}$ (see Figure 2.1).

2. Since the time of service of a client is taken into account in the cost matrix, the time spent in each city is zero.

3. The capacities of our vehicles in the VRP meaning is infinite: a same vehicle can serve as many clients as necessary.

4. The number of vehicles in use is fixed and is thus not a part of the problem.

5. There are as many depots as available vehicles. Their positions represent the initial positions of the different vehicles, which is formally fixed. The clients correspond to nodes with numbers from 1 to $n_p$, and the depots correspond to the nodes $n_p + 1$ to $n_p + n_v$. After dropping off their last clients, the vehicles formally go back to their own depot, but as this last trip is not a part of the problem, one takes $C_{i,j} = 0 \forall i \in \{1, \ldots, n_p\}$ and $j \in \{1, n_p\}$.

---

\(^1\)In a first time, we choose $M = \max_{i,j}(dO_{i,j}) + n_p(\max_i(OD_i + \max_{i,j}(DO_{i,j}))$
\( \{n_p + 1, \ldots, n_p + n_v\} \). Besides, \( C_{i,j} \) with \( i \) and \( j \in \{n_p + 1, \ldots, n_p + n_v\} \) is never used, and thus not defined. In order to simulate free starting points of the vehicles, one can replace all the \( C_{i,j} \) values \((i > n_p, j \leq n_p)\) by zero, and consider that the journey of each vehicle begins at the origin of its first client, which is then chosen optimally.

![Oriented graph with geographical points as vertices](image1)

(a) Oriented graph with geographical points as vertices

![Oriented graph with clients as vertices](image2)

(b) Oriented graph with clients as vertices

Figure 2.1: Illustration of the construction of the asymmetric cost matrix used for the VRP approach.

It is important to keep in mind that classically, a VRP with infinite capacities reduces to a TSP: if there is a cost for going back to the single depot, the minimum cost routing will be achieved by letting one single vehicle visit all the cities (see Figure 2.2).

However, in the remainder of the work, the use of the whole available fleet will be systematic as the system is expected to serve the clients as fast as possible, requiring the vehicles to serve different clients simultaneously.

As a matter of fact, compared to the VRP described so far, our objective function is different, and the fact that we consider the average or the sum of the times at which the different cities are visited has an impact on the formulation of the constraints of the problem.

For a classical VRP, one can simply define the binary variables \( x_{i,j,k} \) for \( i, j \in \{1, \ldots, n_p + n_v\} \) and \( k \in \{1, \ldots, n_v\} \) such that \( x_{i,j,k} = 1 \) if vehicle \( k \) rides along the edge linking cities \( i \) and \( j \) and impose constraints such as continuity (if a vehicle arrives in a city, it also needs to leave it), the fact that every city is visited once and only once, and the fact that each vehicle passes through its own depot. Considering depots as cities for which specific constraints are applied (see constraints (2.17) and (2.18)), the classical VRP linear integer model would be written as in Formulation 2.2.
Formulation 2.2 (Classical VRP, 3 index version)

\[
\text{Min } \sum_{i=1}^{n_p+n_v} \sum_{j=1}^{n_p+n_v} C_{i,j} \sum_{k=1}^{n_v} x_{i,j,k} \tag{2.13}
\]

\[\text{s.t.}\]

\[
\sum_{i=1}^{n_p+n_v} \sum_{k=1}^{n_v} x_{i,j,k} = 1, \forall j \in \{1, \ldots, n_p + n_v\} \tag{2.14}
\]

\[
\sum_{j=1}^{n_p+n_v} \sum_{k=1}^{n_v} x_{i,j,k} = 1, \forall i \in \{1, \ldots, n_p + n_v\} \tag{2.15}
\]

\[
\sum_{i=1}^{n_p+n_v} x_{i,l,k} = \sum_{j=1}^{n_p+n_v} x_{l,j,k}, \forall l \in \{1, \ldots, n_p + n_v\}, \text{ and } k \in \{1, \ldots, n_v\} \tag{2.16}
\]

\[
\sum_{j=1}^{n_p+n_v} x_{n_p+k,j,k} = 1, \forall k \in \{1, \ldots, n_v\} \tag{2.17}
\]

\[
\sum_{i=1}^{n_p+n_v} x_{i,n_p+k,k} = 1, \forall k \in \{1, \ldots, n_v\} \tag{2.18}
\]

\[x_{i,i,k} = 0, \forall i \in \{1, \ldots, n_p + n_v\}, \text{ and } k \in \{1, \ldots, n_v\} \tag{2.19}\]

\[
\sum_{i \in S} \sum_{j \in S} x_{i,j,k} \leq |S| - 1, \forall S \subset \{1, \ldots, n_p\} : |S| \leq 2 \text{ and } \forall k \in \{1, \ldots, n_v\}, \tag{2.20}
\]

Formulation 2.2, constraints (2.14) and (2.15) ensure that each city is visited one and only one time, constraints (2.16) ensure continuity \(^2\) (i.e. if a vehicle \(k\) arrives at a city \(l\), it also has to leave it), constraints (2.17) and (2.18) specify that each vehicle visits its own depot. Then, (2.18) constraints prevent a vehicle to stay in a city, and constraints (2.20), usually added lazily in practice, guarantee the absence of subtours in the solution that would not pass to a depot. The VRP as it is presented here has been widely studied, and different methods can be used in order to compute either the optimal solution or approximations of these solutions. An overview of the main methods to solve different variations of this problem is presented in [15]. Another possibility to avoid the use of lazy constraints is to modify the formulation by adding variables representing the times by which the cities are visited and to add constraints on these times so that they are consistent with the chosen routes. Such a method is applied in the next paragraph, where the objective function (2.13) of the classical VRP is modified to describe the problem this work aims at solving.

\(^2\)Note that constraint (2.16) renders the constraints set (2.14) or (2.15) redundant, as if (2.14) are satisfied, (2.15) will be automatically satisfied via (2.16) ans vice versa.
2.2 Problem seen as a variation of the classical VRP (the MinAvg and the MinMax problems)

In our problem however, where the maximum visit times and the average (or equivalently the sum) of these times are taken into account by the objective function, it is necessary to specify explicitly the time at which the cities are visited. Let \( t_i \) denote the time after which city \( i \) is visited, i.e. the time after which the client \( i \) is picked up by a vehicle. The objective function reads

\[
\text{Min} \sum_{i=1}^{n_p} t_i + W \text{maxTime},
\]

where \( \text{maxTime} = \max_{i \in \{1, \ldots, n_p\}} t_i \).

The variant of the VRP when only the first term of (2.21) is used is referred as a MinSum, a MinAvg problem, or a Cumulative Capacited Vehicle Routing Problem (CCVRP)\(^3\). When the objective function is given by maxTime only, one talks about a MinMax problem.

These alternatives to the classical VRP have been investigated only quite recently. The main paper introducing these two problems, exploring their differences with the classical VRP and proposing heuristic algorithms to solve them was written by A.M. Campbell in 2008 [8].

These problems arose in the context of delivering humanitarian aid in large zones touched by natural disasters. In such situations, the fact that the populations obtain help quickly is more important than the minimisation of the costs induced by the operation. The deliveries have to be both fast and fair: the waiting times have to be as short as possible on average, but as leaving populations without help for a long time can be extremely damageable (one can guess the number of causalities does not grow linearly with the waiting time), it can be sometimes interesting to let the average waiting time slightly increase in order to lower significantly the waiting times of the populations served in last positions.

For our optimisation of a public transport network based on the satisfaction of the clients, these objective functions are thus interesting.

Note that the concept of a minimisation of the maximum cost and / or average cost does not appear only in extensions of VRPs. These notions can also be used, for example, in the context of the Facility Location Problem. In general, there exist more sophisticated objective functions than (2.21) in order to obtain fair and efficient solutions. Ogryczak and Sliwinski [19] proposed for example to enhance the MinMax problems by minimising the average costs for the part of the demand that suffers from a service with the worst performances. As in our case of the objective function (2.21), they managed to build a set of linear constraints completing their objective function.

Table 2.1 compares the different characteristics of the classical VRP and the two aforementioned variants: the MinSum and the MinMax problem. TRP is a shorthand for Travelling Repairman Problem. It corresponds to the particular case of the MinSum VRP when only one vehicle is available.

Figure 4 presents an example that illustrates the fact that the optimal solution of a CCVRP can be fundamentally different from the one of a classical VRP.

\(^3\)The word capacited is however in this case inappropriate given the infinite capacities of our vehicles in the sense of the VRP, but it is under this name that the general problem is denoted in the literature, and the notation CCVRP will thus still be used in the remainder of this text to refer to the problem with the case of \( W = 0 \)
(a) Classical VRP: if there is no constraints on the capacities, it is more interesting to use only one vehicle, as the classical VRP does not take into account the delivery times. In this case, the sense of the distribution is not important.

(b) CCVRP: if the number of cities is greater or equal to the number of available vehicles, all the vehicles are used.

Figure 2.2: Comparison of the classical VRP with the CCVRP or MinSum problem in a case of 3 cities (blue) and one depot (red), with at least 3 available vehicles. The time cost between any pair among these four points is supposed to be equal to 1. This figure is taken from [18]

<table>
<thead>
<tr>
<th>Single vehicle equivalent problem</th>
<th>VRP</th>
<th>MinSum</th>
<th>MinMax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>TSP</td>
<td>TRP</td>
<td>TSP</td>
</tr>
<tr>
<td>Function to minimise</td>
<td>(\sum_{i,j} C_{i,j} \sum_k x_{i,j,k})</td>
<td>(\sum_i t_i)</td>
<td>(\max_i t_i)</td>
</tr>
</tbody>
</table>

Table 2.1: Characteristics of the classical VRP and of the MinSum and MinMax problems derived from it.

The model is written in **Formulation 2.3**
**Formulation 2.3 (MinAvg-MinMax version of the VRP)**

\[
\begin{align*}
\text{Min} & \sum_{i=1}^{n_p} t_i + W\text{maxTime}, \\
\text{s.t.} & \\
& \sum_{j=1}^{n_p} \sum_{k=1}^{n_v} x_{i,j,k} = 1, \forall j \in \{1, \ldots, n_p + n_v\} \\
& \sum_{i=1}^{n_p} \sum_{k=1}^{n_v} x_{i,j,k} = 1, \forall i \in \{1, \ldots, n_p + n_v\} \\
& \sum_{i=1}^{n_p} \sum_{j=1}^{n_v} x_{i,j,k} = 1, \forall i, k \in \{1, \ldots, n_v\} \\
& x_{i,j,k} = 0, \forall i, j, k \in \{1, \ldots, n_p + n_v\}, \\
& t_i = 0, \forall i \in \{n_p + 1, \ldots, n_p + n_v\} \\
& t_j \geq t_i + C_{i,j} - (1 - x_{i,j,k})T, \forall j \in \{1, \ldots, n_p\}, i \in \{1, \ldots, n_p + n_v\} \\
& \text{MaxTime} \geq t_i, i \in \{1, \ldots, n_p + n_v\}.
\end{align*}
\]

This formulation exhibits important differences with the previous one for classical VRP. Constraints (2.28) specifies that all the vehicles leave their depot at time \(t = 0\). Constraints (2.29) reflect the fact that if a vehicle arrives at city \(i\) at time \(t_i\), it cannot arrive at city \(j\) (which cannot be a depot) before time \(t_i + C_{i,j}\). \(T\) is a constant large enough so that in any case, if \(x_{i,j,k} = 0\) the constraint reads \(t_j \geq h\), with \(h = t_i + C_{i,j} - T \leq 0\). In practice, if no constraints on the minimum times at which client can be served are added, constraints (2.29) are always satisfied to equality because of the objective function. Constraint (2.30) is the definition of the maximum waiting time and this inequality will be tight once again because of the objective function.

The trip from the last city to the depot is no more considered. As matter of fact, since a time \(t_i\) must be attributed to each city, the routes can no more be represented by a cycle, which is not a problem since the aforementioned trip is artificial and free of cost. This explains why constraint (2.18) of the VRP formulation have no equivalent in the present formulation. Furthermore, this formulation does not require subtour elimination since subtours are prevented by constraints (2.29).
2.3 Vehicles with capacity of more than one passenger (ride sharing)

In the case where the vehicles are allowed to carry several clients at the same time, and since when a client is picked up he will not necessarily ride directly towards its destination point, there is no point in minimising the waiting time any more. Instead, the objective function is based on the arrival times, i.e. the time at which the clients are dropped at their destination point. To model once again the problem as a MinMax and MinSum inspired from the VRP, it is now necessary go back to a graph representation where each vertex corresponds to a geographical point, i.e. an origin, a destination or a depot point. In this situation, one can thus reuse the precedent formulation with some modifications. First of all, the number of cities is now equal to $2n_p$. The number of depots remains unchanged and is equal to $n_v$. Let us denote by $P$ the set of origin and destination points, so that $|P| = 2n_p$, and for all $i \in \{1, \ldots, |P|\}$ define $\text{nature}_i = 1$ if the point is an origin point, and $\text{nature}_i = -1$ if the point is a destination point. Since only the arrival times are taken into account, the objective can be written

$$\text{Min} \sum_{i \in P} t_i \left(1 - \frac{\text{nature}_i}{2}\right) + W \max_{i \in P} t_i (-\text{nature}_i). \quad (2.31)$$

Then, it is not possible to visit the different cities in any order because it is always required to visit the origin point of a given client before visiting its destination point. Furthermore, the number of passengers that can be found at the same time in a same vehicle should not overcome the vehicle capacity $N_{\text{max}}$.

If we take as a convention that the $i$th client with $i \in \{1, \ldots, n_p\}$ has for origin point $i$ and for destination point $i + n_p$, the precedence relation constraints read

$$t_i \leq t_{i+n_p}, \forall i \in \{1, \ldots, n_p\}. \quad (2.32)$$

The fact that a client is dropped off at its destination point by the same vehicle as the one that picked him up at its origin point is represented by the constraints

$$\sum_{j \in P} x_{i,j,k} = \sum_{j \in P} x_{i+n_p,j,k}, \forall i \in \{1, \ldots, n_p\}, \forall k \in \{1, \ldots, n_v\} \quad (2.33)$$

Let now $R_{l,k}$ be the $l$th point visited by vehicle $k \in D$, except depots. The capacity constraints can be written

$$\sum_{l=1}^L \text{nature}_{R_{l,k}} \leq N_{\text{max}}, \forall L \in \{1, \ldots, \text{NPoints}_k\}, \forall k \in D \quad (2.34)$$

where NPoints$k$ is the number of points visited by vehicle $k$, depot excluded.

Note that the latest constraints are not written as linear programming constraints. However, since the problem will be solved by means of metaheuristics, and not via linear programming which already revealed inappropriate for the precedent problem, obtaining a complete MILP formulation is not crucial.

2.4 Adding release times

So far, we investigated only the limit case where all the clients wan to leave their origin points at the same time. Relaxing this hypothese and introducing READY_i, $\forall i \in \{1, \ldots, n_p\}$, which is a given of the problem denoting the time before which it is forbidden for any car to visit the
origin point of client $i$ because this client is simply not ready to start its journey\(^4\). The goal is no more to minimise the arrival times of the clients, but rather the laps of time between the moment at which they want to start their journey (referred as release times) and the moment at which they arrive at destination.

The objective then reads

$$
\text{Min } \sum_{i \in P} (t_i - \text{READY}_i) \frac{1 - \text{nature}_i}{2} + W \max_{i \in P} (t_i - \text{READY}_i)(-\text{nature}_i). \tag{2.35}
$$

Note that since $\sum_{i \in P} \text{READY}_i$ is a constant, the first term of the objective function can be reduced to $\sum_{i \in P} t_i \frac{1 - \text{nature}_i}{2}$, but it is important to keep in mind that the value of $W$ should be adapted if the same effect is to be obtained. The supplementary constraint is simply that vehicles cannot arrive at origin points in advance:

$$
t_i \geq \text{READY}_i, \forall i \in \{1, \ldots, n_p\}. \tag{2.36}
$$

### 2.5 Towards more fairness

Until now, the quantities that were systematically minimised for each client were totally independent of the length of the distance between the origin and the destination point of this client. The idea is to consider that a situation where a client requiring a short trip and a client requiring a long trip encounter similar waiting/arrival times is not fair. Figure 2.3a illustrates a case where the maximum relative arrival times are automatically minimised by solving the car sharing problem with a minimisation of the maximum absolute arrival time, while it is not the case in the configuration of Figure 2.3b. For this second configuration, the route $O_2, O_1, D_1, D_2$ gives arrival times of 32 and 62 for clients 1 and 2 respectively, i.e. relative arrival times of 16 and 3.1. On the other hand, for the route $O_2, D_2, O_1, D_1$, the arrival times are 52 and 20 (smaller average and maximum than with the previous route), but there is a prohibitive (more) prohibitive relative arrival time of 26 for client 21. Using the objective function

$$
\text{Min } \sum_{i \in P} t_i \frac{1 - \text{nature}_i}{2} + W_{\text{rel}} \max_{i \in P} \frac{t_i}{C_{i,i+n_p}}(-\text{nature}_i), \tag{2.37}
$$

with $W_{\text{rel}}$ chosen consistently, which minimises the maximum relative arrival times in this case, allows the system to chose more fair solutions with respect to this length of required journey when it is possible.

\(^4\)The release times are added to the ride sharing problem in this section, but they could without any problem be added to the car sharing problem.
(a) The client with a small trip demand is naturally advantaged by objective functions on absolute waiting/arrival time.

(b) The client with a small trip demand is served after the one with long trip except if the objective function takes into account relative waiting/arrival times.

Figure 2.3: Illustration of the relevance of the objective functions including relative arrival times.
Chapter 3

Methodology of resolution

In this Chapter, we first recall the principles of local search procedure and of metaheuristics that are used to solve discrete optimisation problems for which it is not possible to obtain the optimal solution within a reasonable time with an exact algorithm such as the Branch and Bound algorithm used in most of the linear solvers. We then set the focus on a Simulated Annealing metaheuristics, and after having described this method and presented its parameters, we describe a simple local search procedure suited to the problems described in Chapter 2. This local search procedure is then embedded in the Simulated Annealing procedure and measurements of the efficiency of the obtained algorithm are then performed in terms of both the execution time and the reached value of the objective function. The last section of this chapter is devoted to avenues for improving the performances of the procedure.

3.1 Local search and metaheuristics

It is well known that solving NP-hard problems as the Travelling Salesman Problem or the classical Vehicle Routing Problem is very challenging. However, such problems can be solved to optimality in a reasonable time for moderate instance sizes. For example, a simple linear problem formulation of the TSP was experimented using the Gurobi solver [2] for an instance of 50 cities, and the optimal solution was obtained within 3 seconds. On the contrary, for the TRP (Traveling Repairman Problem), which can be modelled with exactly the same formulation as the TSP except that the objective function involves more variables, 67 seconds were necessary to obtain the optimal solution for an instance of 10 cities only. For solving classical TSP or VRP for large instances or MinSum and MinMax problems even for small instances, people have to abandon exact algorithms and use heuristic and meta-heuristic approaches instead.

3.1.1 Solution space and local search.

The solution space of a problem represents the set of its (feasible) solutions. For the problems described in Chapter 2, a solution correspond to a valid set of routings for the different vehicles. It is interesting to have an idea of the size of the solution space as a function of the size of our demand-based public transit system problem. The considerations that follow are based on the car-sharing problem (one client per city or point). They could be expanded to the ride sharing problem by multiplying the number of cities or points by 2, but one should then also take into account the practical restrictions on the routes (visiting the origin point of a client before their destination and respecting the vehicles capacity).

In the car-sharing problem, let us consider an ideal situation where the clients are evenly distributed among the vehicles, i.e. where each vehicle serves an equal number $r = n_p/n_v$ of clients. First of all, let us compute the number of possible ways to distribute the different passengers among the vehicles, without considering the order in which they are served. There
Table 3.1: Values of the different factors of the number possible solutions function given by (3.1) as a function of \( n_p \) and \( n_v \) for the car-sharing problem.

<table>
<thead>
<tr>
<th>( n_p )</th>
<th>( \log_{10}(\beta) )</th>
<th>( \log_{10}(\gamma) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>0</td>
<td>13.12</td>
</tr>
<tr>
<td>20.0</td>
<td>3.0103</td>
<td>24.946</td>
</tr>
<tr>
<td>30.0</td>
<td>7.7815</td>
<td>38.983</td>
</tr>
<tr>
<td>40.0</td>
<td>13.802</td>
<td>54.471</td>
</tr>
<tr>
<td>50.0</td>
<td>20.792</td>
<td>71.043</td>
</tr>
<tr>
<td>60.0</td>
<td>28.573</td>
<td>88.48</td>
</tr>
<tr>
<td>70.0</td>
<td>37.024</td>
<td>106.64</td>
</tr>
<tr>
<td>80.0</td>
<td>46.055</td>
<td>125.41</td>
</tr>
<tr>
<td>90.0</td>
<td>55.598</td>
<td>144.73</td>
</tr>
<tr>
<td>100.0</td>
<td>65.598</td>
<td>164.53</td>
</tr>
</tbody>
</table>

are \( C^n_r \) ways of filling a first vehicle with \( r \) clients among \( n_p \). For each of these possibilities, there are \( C^{n_p-r}_r \) ways to fill a second vehicle, and so on and so forth. If for a group of clients travelling in the same vehicle, the fact that they are served by a vehicle A or a vehicle B is important (e.g. because it is considered that each vehicle starts its route from a different given depot point), then the number of possible configurations is multiplied by \( n_v! \). Then, for a given distribution of the clients among the vehicles, there are \( (r!)^{n_v} \) different permutations possible (\( n_v! \) for each vehicle). The total number of possibilities is given by

\[
n_{sol}(n_p, n_v) = n_v! (r!)^{n_v} \prod_{m=0}^{n_v-1} C^{n_p-r_m}_r . \tag{3.1}
\]

Table 3.1 shows some values of the factors that compose \( n_{sol} \) for a couple of \( n_p \) and \( n_v \) values. Naturally, the real size of the solution space is much larger, as only the cases where the clients are evenly distributed among the vehicles are considered by equation (3.1) in order to give an idea.

Most of the approximation algorithms (heuristics and metaheuristics) are based on the local search concept. The basic idea is, starting from a solution, to browse the solution space step by step, following a given strategy in the hope to converge towards the best solution possible. Naturally, only a small part of the space solution is visited. Practically, from a given solution, it is possible to go to another one if it belongs to the neighbourhood of this solution, i.e. if it is reachable by performing one single move. For a classical VRP problem, the moves proposed by Osman in [20] simply consist in either removing a city from the route of a vehicle and to insert it somewhere in the route of another (or the same) vehicle or to interchange the position of two cities in the lists of cities to be visited by the vehicles. This neighbourhood definition and some variations for the ride sharing problem are used in the implementation of our optimisation system described in the next sections.

The idea is to be able to compute the cost related to a move as fast as possible, as this operation is to be performed a large number of times in order to get a solution as reliable as possible. The aforementioned strategies consist in choosing which kind of moves are performed (e.g. browsing the whole neighbourhood and choosing the best move,...) and what is the subsequent action (e.g. updating the solution according to a move only if it improves the objectives,...). Note that the notion of move and neighbourhood is very general. For example, Shawn developed for example a large neighbourhood search procedure for the classical VRP [21]. In his procedure, a move consists in the removal of a set of cities from the current solution.
routing followed by an optimised reinsertion of these cities. Heuristics based on local search (e.g. the steepest descent) allow to reach easily a minimum in the case of a minimisation problem. However, one should keep in mind that such a minimum is likely to be only a local minimum and to correspond to an objective value much less interesting than the optimum. This is why it is often necessary to adopt a strategy allowing to escape from the local extrema.

3.1.2 Metaheuristics

Quoting the Handobook of Metaheuristics [16], metaheuristics are “solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space”. The field of metaheuristic is quite recent and has been recognized only in the 1980’s [9]. Metaheuristics have the advantage to have a very general scope and to be applicable to a wide range of discrete optimisation problems. Metaheuristics are among others based on stochastic phenomena (e.g. Simulated Annealing, Genetic algorithms) or on a memory process (e.g. Tabu Search). More recent techniques tend to mix the different aspects of these metaheuristics [16]. In this work, we chose to use the Simulated Annealing metaheuristics, but the procedures of computation of moves developed hereafter are totally suited for other metaheuristics, for example the Tabu Search.

3.2 Simulated annealing

3.2.1 Principle

The simulated annealing metaheuristics is based on a stochastic process. From a given current solution, the strategy to select a move simply consists in choosing a neighbour solution randomly. If this solution is better than the current one (i.e. if the cost of the move Δ is negative), the current solution is immediately replaced by the corresponding neighbour. If Δ > 0, however, the move is accepted with a probability equal to \( \exp(-\Delta/\theta) \), where \( \theta \) is a parameter referred as the temperature. The higher the temperature, the greater the possibilities to accept bad solutions from the neighbourhood. The possibility of accepting such more costly solutions allows to explore better the solution space and to avoid getting trapped in by local extrema. The temperature is progressively decreased along the procedure to allow the convergence towards a good solution.

It is important to keep in mind that there exist no guarantee about the quality of the solution obtained by the simulated annealing as it will be used 1. It is always possible to encounter ill-conditioned problems with a behavior of the objective function that does not allow the procedure to lead easily to a satisfactory solution.

3.2.2 The cooling schedule

Our implementation is based on a very simple multiplicative cooling schedule based on two parameters: the number \( n \) of iterations of a temperature step (i.e. the number of iterations performed at a same temperature) and the factor \( f < 1 \) by which the temperature is divided at the end of each temperature step (i.e. every \( n \) iterations). Thus, after a number \( \tau \) of temperature steps, the temperature is equal to \( \theta = \theta_0 f^\tau \). The implication of the choice of a cooling schedule on the quality of the solution is not trivial and there exist a large number of possibilities. Notably, many adaptive schedules, inspired from thermodynamics and statistical physics concepts are introduced in [24]. The initial temperature \( \theta_0 \) should be chosen as a value of the same order of

\[1\] It has been demonstrated that the simulated annealing process asymptotically converges towards the optimal solution if a certain cooling schedule is applied. However the application of this schedule implies unaffordable computational costs.[24]
magnitude as the largest cost of move that could be imagined. In our case, we take

\[ \theta_i = \frac{n_p}{n_v} 10(1 + W) \text{mean}(C), \]  

(3.2)

\( \text{mean}(C) \) represent the average value of the cost matrix elements, having the order of magnitude of the delay induced by the insertion or modification of one point in a route and \( n_p/n_v \) being the order of magnitude of the number of clients impacted by such a modification. The whole is multiplied by 10 in order to reach the order of magnitude of the worst cases. The \( W \) factor (that multiplies the maximum waiting / arrival time in the objective function) is also multiplied by 10 \(^2\).

**Important note:** The value of the \( f \) parameter will systematically be fixed to 0.99 for the rest of this work. Only the \( n \) parameter will be modified for the different experiments.

### 3.2.3 Summary

The whole procedure has been implemented in the *julia* language \(^3\). The procedure starts with the generation of a random solution for which the only constraint (apart from feasibility in the ride sharing problems) is that each vehicle route serves the same number of clients, to one unit. The temperature is initialised and the corresponding value of objective function is computed. A random move is then generated. Concretely, by "generation of move", it is simply meant the generation of a few numbers describing the move. For example, specifying the move "Remove the \( p^{th} \) client served by vehicle \( v \) and place it just before the \( p^{th} \) client served by vehicle \( v' \)" only requires 5 numbers: \( p, p', v, v' \) and a number specifying the nature of the move (here a removal and an insertion). With these information, the challenging part is the computation of the corresponding cost of the move \( \Delta \). A large part of the next section of is devoted to the description of this computation which needs to be as fast as possible. Then, the current solution is potentially updated. At the end of the process, the objective of the obtained solution is recomputed from scratch and the obtained value is compared to the current objective. This verification is mostly useful for the development of the code in the framework of different problems. The whole simulated annealing procedure is represented in Figure 3.1.

### 3.3 Application to the combined MinSum and MinMax VRP

In this section, we particularise the simulated annealing to the car sharing and ride sharing problems that have been presented in Chapter 2. This requires to describe which moves can be generated and how their related cost can be computed efficiently for these two problems. The variants of the ride sharing problem that take into account the relative arrival times and the release times of the clients are also discussed at the end of the section. The different solutions to the VRP variants appearing throughout the SA procedure are always represented by a set of lists of ordered points (cities) to be visited for each vehicle (see the \( R \) matrix appearing a bit later in the text). Remember that for the car-sharing problem, each point represents a client, and that for the . In both cases, the given matrix elements \( C_{i,j} \) indicate the time cost related to a displacement from point \( i \) to point \( j \).

#### 3.3.1 Car sharing problem: local search for a combined MinSum-Minmax VRP variation

When considering the car sharing problem with all the client desiring to leave their origin point at the same time, one handles the following four elements describing the current solution:

\(^2\)Note that multiplying \( W \) by \( \frac{n_p}{n_v} \) was *a priori* not necessary. The choice of the initial temperature is however quite arbitrary

\(^3\)julia is an emerging programming language gathering the friendly syntax of scientific programming languages such as MATLAB and a fast execution as the code is compiled.
Figure 3.1: Flow chart representing the adopted simulated annealing procedure. \( \text{obj} \) represents the current value of the objective and \( \text{cpt} \) and iteration counter.

- \( R \), the routes matrix. For \( n_v = 2 \) and \( n_p = 5 \), an example of this matrix is given by
  \[
  R = \begin{pmatrix}
  6 & 2 & 3 & 4 & 6 & 0 & 0 & 0 \\
  7 & 1 & 5 & 7 & 0 & 0 & 0 & 0 
  \end{pmatrix}
  \]
  where the numbers 1 to \( n_p \) refer to client points and the numbers \( n_p + 1 \) to \( n_p + n_v \) refer to depots points (here 6 and 7).

- The times by which these points are visited: \( t_{k,i} \) is the time at which point \( R_{k,i} \) is visited. The times of visit of the depots are always set to zero. For the aforementioned \( R \) matrix, one can have
  \[
  t = \begin{pmatrix}
  0 & 2 & 6 & 23 & 0 & 0 & 0 & 0 \\
  0 & 2 & 7 & 0 & 0 & 0 & 0 & 0 
  \end{pmatrix}
  \]

- A vector with the number of clients served by each vehicle. In order to be able to keep the same notation when considering VRP variations where cities represent geographical points and not clients, this vector is called \( \text{nb\_points} \). The depots being excluded from the counting of visited points, for the mentioned \( R \) example, one has
  \[
  \text{nb\_points} = \begin{pmatrix}
  3 & 2
  \end{pmatrix}
  \]

- The maximum departure times of the clients are given by the vector
  \[
  \text{maxTimes} = \begin{pmatrix}
  23 & 7
  \end{pmatrix}
  \]

\[\text{Note that the fact that the elements of the second column of } t \text{ are different from zero indicates that we represent a problem where the depots are assumed to be prealably fixed.}\]
In order make the following equations easy to understand, the notations $M_{i,j}$ and $M[i,j]$ will be used equivalently for representing an element of any $M$ matrix. The convention taken for the labelling of the elements of a vector or a matrix uses indices starting from 1. \(^5\) Note that $R$ and $t$ should be defined with a minimum of $L = n_p + 2$ columns in order to be able to represent all the possible situations susceptible to appear along the procedure.

As mentioned in the previous section, the algorithm begins by the generation of a random solution. A random $R$-matrix is thus generated by distributing the clients randomly among all the vehicles. Initially, all the vehicles have thus a route with a same amount of points to visit to one unit and the vectors nb_points and maxTimes are initialised consequently. Then, browsing $R$ and using the information of $C$ allows to build the corresponding $t$-matrix via

$$t_{k,i} = t_{k,i-1} + C[R_{k,i-1}, R_{k,i}] \text{ for } i \geq 2 \text{ and } t_{k,1} = 0, \forall k$$

and to compute the current objective:

$$\text{obj} = \sum_{k=1}^{n_v} \sum_{i=1}^{\text{nb_points}_k+1} t_{k,i} + W \max_{k \in \{1, \ldots, n_v\}, i \in \{1, \ldots, L\}} t_{i,k}.$$ 

Then, a random move is generated. A move is characterised by its nature (a shift of one point from a place in the $R$-matrix to another or an interchange between two points (elements) of the $R$ matrix), two numbers of the vehicles involved in the moves (referred as $v_A$ and $v_B$), and the indices of the points concerned in each of these vehicles (referred as index$_A \in \{2, \ldots, \text{nb_points}_{v_A} + 1\}$ and index$_B \in \{2, \ldots, \text{nb_points}_{v_B} + 2\}$ respectively). Note that the case $v_A = v_B$ is allowed. The different moves and the computation of their related costs are developed hereafter.

**Cost of a shift from a vehicle $v_A$ to a vehicle $v_B \neq v_A$**

![Figure 3.2: Shift from a vehicle $v_A$ to a vehicle $v_B$. In this case, nb_points$_{v_A}$ = 7, nb_points$_{v_B}$ = 4, index$_A$ = 5 and index$_B$ = 4.](image)

A shift from a vehicle $v_A$ to a vehicle $v_B \neq v_A$ can be decomposed into two successive operations: removing the client $c_A = R[v_A, \text{index}_A]$ from the route of vehicle $v_A$ and inserting this client in the route of vehicle $v_B$, just before the client (or depot) $c_B = R[v_B, \text{index}_B]$. The procedure is illustrated in Figure 3.2.

Let $\Delta_A$ and $\Delta_B$ denote the costs induced by the two steps of this process. In the case of a classical VRP, one would easily obtain the cost of the move as $\Delta_{\text{VRP}} = \delta_A + \delta_B$, where $\delta_A$ and $\delta_B$ are given by

$$\delta_A = a - b - c \text{ and } \delta_B = d + e - f. \quad (3.3)$$

\(^5\)Convention used in the Julia language.
In equations (3.3), \( a, b, c, d, e \) and \( f \) refer to the costs indicated in Figure 3.2 with among others \( a = C[R[v_A, \text{index}_A - 1], R[v_A, \text{index}_A + 1]] \) and \( e = C[R[v_A, \text{index}_A - 1], R[v_B, \text{index}_B]][\). However, given the objective function of the combined MinSum MinMax problem, the modification of the times at which the clients are served after the shift should be taken into account, and finally, one obtains

\[
\Delta_A = \frac{(\text{nb}_{\text{points}}_{v_A} - \text{index}_A + 1)}{\text{number of clients impacted by the removal}} \cdot \delta_A - \frac{t(v_A, \text{index}_A)}{\text{waiting time of the removed client}}
\]

(3.4)

and

\[
\Delta_B = \frac{(\text{nb}_{\text{points}}_{v_B} - \text{index}_B + 2)}{\text{number of clients impacted by the insertion}} \cdot \delta_B + \frac{t(v_B, \text{index}_B)}{\text{waiting time of the inserted client}} + C[R[v_B, \text{index}_B - 1], R[v_A, \text{index}_A]].
\]

(3.5)

These \( \Delta_A \) and \( \Delta_B \) values do not take into account the potential variation of the maximum visiting time. Defining \( \text{newMax}_A = \max_{v_A} + \delta_A \) and \( \text{newMax}_B = \max_{v_B} + \delta_B \), if \( \text{newMax} \) is the maximum value among \( \max_{v_k}, k \in \{1, \ldots, n_v\} \) \{\( v_A, v_B \}\}, one can write

\[
\Delta = \Delta_a + \Delta_b + W(\text{newMax} - \max_{k \in \{1, \ldots, n_v\}} \max_{v_k}).
\]

(3.6)

Cost of a shift inside a vehicle route

When \( v_A = v_B \), it can be trivially found that if \( \text{index}_A = \text{index}_B \) or if \( \text{index}_A = \text{index}_B - 1 \), there is no move to be observed. In the other cases, the cost for removing the client from their initial position is computed the same way as previously with equation (3.4). Computing \( \delta_b \) the same way as previously, \( \Delta_B \) can be obtained by multiplying \( \delta_B \) by the number of clients impacted by the insertion of the client, which is equal to \( \text{nb}_{\text{points}}_{v_B} - \text{index}_B + 1 \) if \( \text{index}_B < \text{index}_A \) and to \( \text{nb}_{\text{points}}_{v_B} - \text{index}_B + 2 \) if \( \text{index}_B > \text{index}_A + 1 \). The former case is illustrated in Figure 3.3.

**Cost of an interchange between two different vehicles**

An interchange move simply consists in switching two points (except depots) from the \( R \) matrix. An interchange move can thus either happen for two clients served by different vehicles or two clients served by the same vehicle.

An interchange between two clients from vehicles \( v_A \) and \( v_B \neq v_A \) is illustrated in Figure 3.4. Here, the difference in the waiting times of the clients served after the changes are given by

\[
\delta_A = a + b - c - d \quad \text{and} \quad \delta_B = e + f - g - h,
\]

(3.7)

where \( a = C[R[v_A, \text{index}_A - 1], R[v_B, \text{index}_B]] \), etc.
Figure 3.4: Interchange between the vehicles $v_A$ and $v_B \neq v_A$. In this case, \nb_points_{v_A} = \nb_points_{v_A} = 5$, index$_A = 4$ and index$_B = 3$.

With a reasoning similar to previously, one gets

$$
\Delta_A = \delta_A (\nb_points_{v_A} - \text{index}_A + 1) - t[v_A, \text{index}_A] + t[v_A, \text{index}_A - 1] + a \tag{3.8}
$$

and

$$
\Delta_B = \delta_B (\nb_points_{v_B} - \text{index}_B + 1) - t[v_B, \text{index}_B] + t[v_B, \text{index}_B - 1] + e. \tag{3.9}
$$

Cost of an interchange inside a vehicle

Figure 3.5: Interchange inside the vehicle $v_A$. $\nb_points_{v_A} = 8$, index$_A = 3$ and index$_B = 7$.

The advantage of equations (3.8) and (3.9) is that they are still valid when $v_A = v_B$, in the case where $|\text{index}_A - \text{index}_B| > 1$. As shown in the example presented in Figure 3.5, $\delta_A$ and $\delta_b$ can be computed with equations (3.7) and the number of clients impacted by both the routes changes is given by $\nb_points_{v_A} - \text{index}_A$ or $B + 1$

In the case where the clients to be interchanged are neighbour, i.e. if $\text{index}_B = \text{index}_B + 1$, one can simply compute

$$
\delta_{AB} = -C[R[v_A, \text{index}_A - 1], R[v_A, \text{index}_A]] - C[R[v_A, \text{index}_A], R[v_A, \text{index}_B]] \\
- C[R[v_A, \text{index}_B], R[v_A, \text{index}_B + 1]] + C[R[v_A, \text{index}_A - 1], R[v_A, \text{index}_B]] \\
+ C[R[v_A, \text{index}_B], R[v_A, \text{index}_A]] + C[R[v_A, \text{index}_A], R[v_A, \text{index}_B + 1]] \tag{3.10}
$$

which impacts $\nb_points_{v_A} - \text{index}_B + 1$ clients.

The treatment of the maximum times happens exactly as in the case of a shift move (equation 3.6) to obtain $\Delta$.

If any of the moves presented so far is accepted, (i.e. if $\Delta < 0$ or $\exp(-\Delta/\theta) > \lambda$, $\lambda$ being a random parameter with an uniform distribution between 0 and 1), the appropriate lines of $R$ and $t$ are uploaded, as well as $\nb_points$ and maxTimes.
3.3.2 Ride sharing problem with maximum arrival times

When considering the possibility to put several clients together in a same vehicle, it is necessary to consider, in the VRP variant problem, a number of cities equal to twice the number of clients (one origin and one destination point by client). In this problem, the clients are still numbered from 1 to \( n_p \), but the number of non-depots points is equal to \( 2n_p \). The origin points of a client \( c \) is labelled point \( c \), and his destination point is labelled \( N_p + c \). The depot of the \( k^{th} \) vehicle is a point that is labelled \( 2n_p + k \). However, in order to make the reading of the routes easier, the destination points of client \( c \) will be written as \(-c\). An example of a \( R \) matrix for this problem with \( n_p = 5 \) and \( n_v = 2 \) is given by

\[
R = \begin{pmatrix}
11 & +1 & +4 & -1 & +2 & -4 & -2 & 11 & 0 & 0 \\
12 & +3 & +5 & -3 & -5 & 12 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

The \( t \) matrix plays the same role as in the previous problem, as well as maxTimes and \( \text{nb\_points} \), with \( \text{nb\_points}_k \) representing twice the number of clients on the route of vehicle \( k \). To solve the problem, it is necessary to introduce some new matrices. Their example values are consistent with the routes presented in the latest \( R \) matrix.

- The Out matrix that shows at which points clients are getting out of a vehicle. It will be useful when computing the objective variations since only the times by which the clients get out the vehicle are taken into account in the objective function.

\[
\text{Out} = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

- The INDEX matrix, which contains the information about the positions in the \( R \) matrix where the origin points (col 1) and the destination points of a client can be found (without specifying the car):

\[
\text{INDEX} = \begin{pmatrix}
2 & 4 \\
5 & 7 \\
2 & 4 \\
3 & 6 \\
3 & 5
\end{pmatrix}
\]

- The population matrix: \( \text{Pop}[k, i] \) denotes the number of clients on board of vehicle \( k \) when it leaves the \( i^{th} \) point of its route. It is used in order to check if the proposed moves are feasible.

\[
\text{Pop} = \begin{pmatrix}
0 & 1 & 2 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

The value of the objective function can be computed as

\[
\text{obj} = \sum_{k=1}^{n_v} \sum_{i=1}^{\text{nb\_points}_k+1} t_{k,i} \text{Out}_{k,i} + W \max_{k \in \{1, \ldots, n_p\}} t[k, \text{nb\_points}_k + 1]
\]

A simulated annealing iteration begins with the choice of a feasible move, i.e. a move that does not lead to a situation where at a given moment, the number of passengers inside a vehicle exceeds the capacity \( N_{\text{max}} \). For this problem, moving one client corresponds to moving two points in the routes of the vehicles: the origin and the destination point of the client. As for the previous problem, the specification of a move consists in its nature (shift or interchange), the two implicated vehicles \( v_A \) and \( v_B \) and the positions of the concerned clients in the vehicles list of point. For interchanges and for the removal of a client from a vehicle, it is sufficient to
chose a random number in \( \{2, ..., \text{nb\_points}_k + 1\} \) for the considered vehicle \( k \). If this number corresponds to an origin (resp. a destination point), the position of the destination (resp. origin) point can be retrieved from the INDEX matrix. The origin and destination indices are referred as \( \text{index}_{v_A, \text{in}} \) and \( \text{index}_{v_A, \text{out}} \) respectively. In the case of an interchange \(^6\), a client should be chosen the same way in the second vehicle. In the case of a shift, in the vehicle where the client will be inserted, it is necessary to chose two random numbers: \( \text{index}_{v_B, \text{in}} \) and \( \text{index}_{v_B, \text{out}} \), which correspond respectively to the positions in vehicle \( v_B \) before which the origin point and the destination point of the client coming from vehicle \( v_A \) is to be inserted. These two value should not lead to a feasible move, they are chosen again until obtaining a valid combination. Once a feasible move is determined, the corresponding move cost \( \Delta \) can be computed.

**Cost of a shift**

As for the former problem, one computes successively the cost due to the removal of the client from the vehicle \( v_A \) and the cost of the insertion of the client in vehicle \( v_B \). In the current version of the software, when \( v_A = v_B \), a copy of the rows of the \( v_A \) line of \( R, t, \text{Pop} \) and \( \text{Out} \) is done. The removal of the client is performed on these copies, so that if the computation of the cost due to the reinsertion of the client can then be performed with the same formula as when \( v_A \neq v_B \). In this case, \( \text{index}_{v_B, \text{in}} \) and \( \text{index}_{v_B, \text{out}} \) are chosen in \( \{1, ..., \text{nb\_points}_v + 2\} \). \(^7\)

Figure 3.6 shows an example of a shift move. One can compute \( \delta_{A, \text{in}} = a - b - c, \delta_{A, \text{out}} = d - e - f, \delta_{B, \text{in}} = g + h - i \) and \( \delta_{B, \text{out}} = j + k - l \). For example, \( a \) is computed as \( a = C[\text{SIN}(R[v_A, \text{index}_{v_A, \text{in}} - 1]), \text{SIN}(R[v_A, \text{index}_{v_A, \text{in}} - 1])] \), where \( \forall z < 0, \text{SIN}(z) = -z + \text{np} \) and \( \forall z \geq 0, \text{SIN}(z) = z \). Note that \( \text{SIN} \) stands for \text{Shift If Negative}. The number of arrival times impacted by these different changes of route (not to be confused with the number of impacted points) are computed using the Out matrix, which allow to obtain the variations of the objective function (maximum arrival time considerations excluded) in both vehicles as \(^8\)

\(^6\)For the seek of simplicity, the interchanges within a given vehicle route are forbidden. They can however still be performed by a series of shift moves. This reduction of the neighbourhood is expected to have only a negligible impact on the solution.

\(^7\)Note that this approach avoids complicated computations but is susceptible to slow down the process.

\(^8\)The '-1' in equation (3.15) allows not to take the removed destination point into account.
\[
\Delta_A = \delta_{A,in} \left( \sum_{i=\text{index}_{v_A,in}+1}^{\text{nb\_points}_{v_A}+1} \text{Out}[v_A,i] - 1 \right) + \\
\delta_{A,out} \left( \sum_{i=\text{index}_{v_A,in}+1}^{\text{nb\_points}_{v_A}+1} \text{Out}[v_A,i] \right)
\]

Number of arrival times impacted by the removal of the client origin point

\[
\delta_{A,out} \left( \sum_{i=\text{index}_{v_A,in}+1}^{\text{nb\_points}_{v_A}+1} \text{Out}[v_A,i] \right)
\]

(3.15)

Number of arrival times impacted by the removal of the client destination point

and similarly \(^9\)

\[
\Delta_B = \delta_{B,in} \left( \sum_{i=\text{index}_{v_B,in}+1}^{\text{nb\_points}_{v_B}+1} \text{Out}[v_B,i] + 1 \right) + \\
\delta_{B,out} \left( \sum_{i=\text{index}_{v_B,out}+1}^{\text{nb\_points}_{v_B}+1} \text{Out}[v_B,i] \right)
\]

Number of arrival times impacted by the insertion of the client origin point

\[
\delta_{B,out} \left( \sum_{i=\text{index}_{v_B,out}+1}^{\text{nb\_points}_{v_B}+1} \text{Out}[v_B,i] \right)
\]

(3.16)

Number of arrival times impacted by the insertion of the client destination point

The maximum times can be handled in a way similar to the former problem, since the last point before going back always corresponds to the destination point of the last client.

Note that for all the configurations met, if \(\text{index}_{v_A,out} = \text{index}_{v_A/B,in} + 1\) for a removal or an exchange, or if \(\text{index}_{v_A/B,out} = \text{index}_{v_A/B,in}\) for an insertion, one consider only a local cost \(\delta_{A/B,in\_out}\) impacting the points from \(\text{index}_{v_A/B,in/out} + 1\) to \(\text{nb\_points}_{v_A/B} - 1\).

**Cost of an interchange**

In the case of an interchange, the computation of \(\delta_{A,in}, \delta_{A,in}, \delta_{B,in}\) and \(\delta_{B,in}\) can be done by reasoning as in the previous cases, and the number of impacted arrival times can be computed by browsing each time the Out matrix between indices \(\text{index}_{A/B,in/out} + 1\) to \(\text{nb\_points} + 1\), no '+1' or '-1' needing to be added in any case.

**3.3.3 Ride sharing with maximum relative arrival time**

With \(\text{trip}_i\) denoting the time required to travel directly from the origin to the destination point of client \(i\), the objective can be computed as

\[
\text{obj} = \sum_{k=1}^{n_a} \sum_{i=1}^{\text{nb\_points}_k} t_{k,i} \text{Out}_{k,i} + W_{rel} \max_{k \in \{1,\ldots,n_p\}, i \in \{1,\ldots,L\}} \left( \frac{t[k,i]}{\text{trip}_k[i]} - \text{Out}[k,i] \right).
\]

(3.17)

The main drawback of this objective function is that this time, when performing a move, the cost relative the variation of the considered maximum for the route of a given vehicle is no more equal to \(\delta_{v_{A/B,in}} + \delta_{v_{A/B,out}}\), which as the case for the absolute arrival times. The moves considered are exactly the same as in the previous problem. A simple version implementation was constructed was implemented by computing an update of the solution whatever the cost of the move and then re-computing this cost from scratch in order to obtain \(\Delta\). This high-cost procedure leads to long computation times, and there are possibilities of ameliorations to be explored.

\(^9\)The '+1' in equation (3.16) allows to take the inserted destination point into account.
3.3.4 Multi-passengers vehicles with release times

The current version of the software allows to solve the ride sharing problem with the following MinSum objective function:

\[ \sum_{i=n_p+1}^{2n_p} (t_i - \text{READY}_i), \]  

(3.18)

defining the release time \( \text{READY}_i \) as the time at which client \( i \in \{1, ..., n_p\} \) is ready to leave their origin point. Instead of considering that the cases where a vehicle arrives at the origin point of a client who is not ready yet are infeasible, we simply assign a cost to the sum of such advance times. This approach allows to reuse most of the functionalities developed for solving the ride sharing problem, and it also allows to study the potential modifications on the behaviour and the performances of the ride sharing system when modifying the weight \( W_{adv} \) attributed to such advance times in the objective function (see Chapter 4). The problems can thus be translated in the minimisation of

\[ \sum_{i=n_p+1}^{2n_p} t_i + W_{adv} \sum_{i=n_p+1}^{2n_p} \text{ADV}_i H(\text{ADV}_i) \]  

(3.19)

with \( \text{ADV}_i = \text{READY}_i - t_i, \ H(x) = 1 \) if \( x \geq 0 \) and \( H(x) = 0 \) if \( x < 0 \). A solution could be for example given by

\[
R = \begin{pmatrix}
11 & +1 & +4 & -1 & +2 & -4 & -2 & 11 & 0 & 0 \\
12 & +3 & +5 & -3 & -5 & 12 & 0 & 0 & 0 & 0 \\
\end{pmatrix},
\]

\[
t = \begin{pmatrix}
0 & 3 & 10 & 14 & 20 & 23 & 25 & 0 & 0 & 0 \\
0 & 2 & 8 & 12 & 13 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix},
\]

\[
\text{ADV} = \begin{pmatrix}
2 & 5 & -10 & 5 & 2 \\
\end{pmatrix}.
\]

The solution is feasible in the sense of the release times set as constraints in the cases where all the elements of the ADV vector are negative. The used algorithm is identical to the ride sharing problem with absolute times, except that in order to compute the cost \( \Delta \) associated to a move, one should add to \( \Delta_A \) and \( \Delta_B \) a penalty of \( W_{adv} \) times the total difference of ‘positive advances’. The idea is to sum up the individual penalties for all the clients in a vehicle route that follow a removal, insertion or interchange. The penalty is computed for each client \( i \) on the basis on the change of departure time \( \delta_{A/B,\text{in}} \) or \( \delta_{A/B,\text{in}} + \delta_{A/B,\text{out}} \), the value of \( W_{adv} \), the initial and the new value of \( \text{ADV}_i \). The operation performed with these two values being dependant of their sign in order to compute the variation of the product \( W_{adv} \text{ADV}_i H(\text{ADV}_i) \).

3.4 Performances of the method

In order to use the algorithms described in the previous section in a wise way, it is important to know, given the problem parameters, how long the program should run to obtain a good quality solution. This amounts to know on one hand how long the iterations during the simulated annealing procedure last, and on the other hand, how many of these iterations are necessary to reach solutions considered as good.

3.4.1 Computation times

We now give a quantification of the rapidity with which the \( \Delta \) evaluation and the potential update of the data are performed as a function of the size of the problem. Therefore, we run \(^{10}\)

\(^{10}\)The executions of the code are performed on the THALES machine which has the following specifications: CPU: Intel Core i7-990X at 3.47GHz, 6 cores, 12 threads; RAM: DDR3-1333 SDRAM (24Gb). All the
the simulated annealing procedure for different values of $n_p$ and $n_v$, but fixing the initial and final temperatures as well as the $n$ parameter. We take $\theta_i = 512.715$, $\theta_f = 1$ and $n = 10$, which amounts to 13100 iterations if $f = 0.99$\textsuperscript{,11} Note that the times measured in this section for low $n$ values only give an idea of the computing times for longer executions. It is indeed not exact to state that the execution time is proportional to $n$. As a matter of fact, the solutions with $n$ being small could for example soon get stuck in an area where the topology is such that the proportion of accepted solutions is lower, and as one can expect (and as it will be shown later), the computation cost of an SA iteration when a new solution is accepted is higher than when it is not.

**Car sharing**

For the car-sharing problem, the two parameters that determine the size of the problem are $n_p$ and $n_v$. Figure 3.7a shows the time computation time required for given values of $n_v$ as a function of $n_p$. For a small number of vehicles ($n_v = 5$), the computation time seems to grow monotonously and linearly with the number of clients. This is however not always the case as shown by the red points in Figure 3.7a where $n_v = 100$ and where one observes a decrease of the computation time with the augmentation of $n_p$ for values of $n_p$ up to 200. This behaviour is to be compared with the proportion of accepted moves presented in Figure 3.8. This comparison allows to clearly see that the iterations with update are much more time consuming than the others. The fact that the significant decrease of the fraction of accepted moves stops around $n_p = 200$ (that is $n_p = 2n_v$) is a consequence of the fact that the code was run with the option of free departure points for the vehicles (and thus that the first client of each vehicle is served at time $t = 0$). Below 200 clients, a certain number of vehicles carry only one client. Interchanges between these vehicles (and shifts towards remaining unused vehicles) are automatically accepted ($\Delta = 0$). When adding more clients, the number of possibilities of such moves is reduced, which renders the process faster. When overcoming 200 clients, cases of vehicles serving one single client become really rare, and adding more clients does not decrease the probability of the aforementioned free moves any more. The elongation of the iterations due to the size of the problem dominates then (positive slope).

![Figure 3.7: Computations times as a function of the size of the problem for the car sharing problem for $n = 10$.](image)

In order to assess the computation times due to the management of the data along the computation times mentioned in this section include the preliminary generation of a random solution and the computation of the objective value of this solution.

\textsuperscript{11}The very poor quality of the obtained solution is not a problem, as the solutions are not studied here.
simulated annealing process but

If the number of iterations is multiplied by 10, i.e. if we use the simulatad annealing parameter \( n = 100 \) instead of \( n = 10 \), it is interesting to notice that the computation times are not exactly multiplied by 10. An example is shown with \( n_v = 5 \) in Figure 3.9 to be compared with Figure Figure 3.7 a. The multiplication factor is as a matter of fact smaller than 10, which reflects the presence of the non negligible fixed computation cost (initialisation,...).

Ride sharing

The same measurements are performed for the ride sharing problem. For this problem, the capacity of the vehicles can be modified as well. The results are presented in Figure 3.10, where the influence of \( N_{\text{max}} \) is clearly noticeable. The variations of computation time due to this parameter originate in the fact that only feasible moves (i.e. move that do not leave to more than \( N_{\text{max}} \) people at the same time in a vehicle) are accepted as candidates for the \( \Delta \) computation. In situations with a large number of clients an a small number of vehicles, infeasible moves are more likely to be generated, leading the program to try several moves before finding a feasible one, which increases the computation time.

For the ride sharing problem with minimisation of the maximum relative arrival times, the
computation times are in general much greater than for the previous problem. As it has been mentioned earlier, in this version of the program, it is necessary to systematically create an update of the solution and recompute the objective from scratch. The red points of Figure 3.11 are to be compared with the ones of Figure 3.10a.

This leads to the conclusion that a fast computation of $\Delta$ handling as few data as possible and as (as performed for the other problems), has a great impact on the computation time!

![Figure 3.11: Ride sharing problem, minimizing the relative arrival times, $N_{\text{max}} = 5, n_v = 100$.](image)

The naïve implementation of this problem leads to extremely high computation times.

### 3.4.2 Quality of the solutions

In this section, we study at the quality of solutions as a function of the $n$ parameter. A solution is said to be of good quality when the related value of the objective function is close to the assumed lower bound and when it is reliable, i.e. it does not vary much from a an execution to another (good reproducibility of the experiments). Figure 3.12a shows the evolution of the objective value of the current solution at the end of each temperature step for different values of $n$. It can be observed that the greater the value of $n$, the better the solution. On Figure 3.12b, which is a magnification of the right part of figure 3.12a, it can be seen that going from $n = 1000$ to $n = 10000$ still allows to obtain an improvement of the objective with an order of magnitude of $10\%$.

Tables 3.2 to 3.5 present statistics on the objective values obtained for different instances of the car sharing problem and for different values of the parameter $n$. The different statistics (average, minimum, maximum, standard deviation and amplitude, i.e. maximum - minimum) correspond to the obtained values of the objective function. For each instance, these values have been divided by the lowest objective function value obtained during the experiments. The $A$ parameter is defined such that $W = A \frac{m}{n}$. Note that in the Tables 3.4 and 3.5, the data for $n = 10000$ are presented for information only but the length of the experiments did not allow to obtain significant samples.

Generally, one observes that both the average, the standard deviation of the objective decrease when $n$ is increased. Comparing Tables 3.2 and 3.3, which both represents small instances of the problem, one can notice that the results are less good when $A = 1$ than when $A = 0$. This can be due to the fact that in the former case, the relative variations of the objective from one solution to another are greater than in the second case. Another reason can be the behavior of the objective function across the solution space which could be less adapted to simulated annealing solutions for a large value of $W$. In the case of $A = 0$, the decrease of the average objective becomes negligible beyond $n = 100$, which is not the case for $A = 1$.

Concerning the average values of the objective function, one can notice, comparing Tables 3.3, 3.4 and 3.5 that the same relative difference between the average and the reference minimum
value, it is necessary to increase \( n \). From Table 3.3 to Table 3.4, the size of the problem is roughly multiplied by 10, and the \( n \) value to reach a relative average around 1.07 (1.077 for Table 3.3 and Table 3.4) has to be multiplied by 10 when passing from the former instance to the second. Besides, for a relative average around 1.3, it is necessary to multiply \( n \) by 50 for passing from the instance presented in Table 3.4 to the instance presented in Table 3.5 which is 8 times larger. This shows that the required number of SA iterations grows faster much faster than linearly with the size of the problem. By contrast, the relative amplitude and standard deviations for a given \( n \) value do not seem to depend on the size of the problem. It is worth to keep in mind that for practical applications which involve approximation and uncertainty on several parameters (e.g. time for going from one place to another which can depend on the environmental conditions), obtaining a solution with objective less than 1% close to the best reachable objective value is probably not crucial.

![Graph](image1.png)

(a) Global view.  
(b) End of process

Figure 3.12: Evolution of the current objective along the simulated annealing procedure. Figure obtained for the car sharing problem \((n_p = 200, n_v = 50, W = 50)\) with \( \theta_i = 512715 \), \( \theta_f = 1 \) and \( f = 0.99 \).

### 3.5 Avenues of improvement

We present in this section the different ways of improvement of the method. Some are quite straightforward and technical (fast computation of \( \Delta \),...), while others, based on more theoretical considerations, require further investigations and experimentation (clusters...). We distinguish the improvements of the simulated annealing method itself from the pre-processing techniques designed to reduce the amount of work to be performed by the simulated annealing procedure.

<table>
<thead>
<tr>
<th>( n )</th>
<th>10</th>
<th>25</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1.1017</td>
<td>1.0493</td>
<td>1.0215</td>
<td>1.0099</td>
<td>1.0023</td>
</tr>
<tr>
<td>Minimum</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Maximum</td>
<td>1.49</td>
<td>1.26</td>
<td>1.08</td>
<td>1.08</td>
<td>1.01</td>
</tr>
<tr>
<td>Standard dev.</td>
<td>0.0883</td>
<td>0.0516</td>
<td>0.0268</td>
<td>0.0180</td>
<td>0.0042</td>
</tr>
<tr>
<td>Amplitude</td>
<td>0.4924</td>
<td>0.2624</td>
<td>0.0857</td>
<td>0.0808</td>
<td>0.0104</td>
</tr>
<tr>
<td>Sample size</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 3.2: \( n_p = 30, n_v = 10, A = 0 \)
<table>
<thead>
<tr>
<th>$n$</th>
<th>10</th>
<th>25</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1.127</td>
<td>1.093</td>
<td><strong>1.077</strong></td>
<td>1.072</td>
<td>1.021</td>
</tr>
<tr>
<td>Minimum</td>
<td>1.018</td>
<td>1.018</td>
<td>1.0</td>
<td>1.003</td>
<td>1.0</td>
</tr>
<tr>
<td>Maximum</td>
<td>1.462</td>
<td>1.36</td>
<td>1.137</td>
<td>1.137</td>
<td>1.086</td>
</tr>
<tr>
<td>Standard dev.</td>
<td>0.083</td>
<td>0.045</td>
<td>0.03</td>
<td>0.031</td>
<td>0.034</td>
</tr>
<tr>
<td>Amplitude</td>
<td>0.443</td>
<td>0.341</td>
<td>0.137</td>
<td>0.133</td>
<td>0.086</td>
</tr>
<tr>
<td>Sample size</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 3.3: $n_p = 30$, $n_v = 10$, $A = 1$

<table>
<thead>
<tr>
<th>$n$</th>
<th>10</th>
<th>25</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1.639</td>
<td><strong>1.342</strong></td>
<td>1.166</td>
<td>1.068</td>
<td>1.026</td>
</tr>
<tr>
<td>Minimum</td>
<td>1.389</td>
<td>1.218</td>
<td>1.08</td>
<td>1.0</td>
<td>1.014</td>
</tr>
<tr>
<td>Maximum</td>
<td>1.922</td>
<td>1.519</td>
<td>1.301</td>
<td>1.152</td>
<td>1.037</td>
</tr>
<tr>
<td>Standard dev.</td>
<td>0.107</td>
<td>0.061</td>
<td>0.048</td>
<td>0.045</td>
<td>0.016</td>
</tr>
<tr>
<td>Amplitude</td>
<td>0.533</td>
<td>0.301</td>
<td>0.22</td>
<td>0.152</td>
<td>0.022</td>
</tr>
<tr>
<td>Sample size</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.4: $n_p = 100$, $n_v = 20$, $A = 1$

<table>
<thead>
<tr>
<th>$n$</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>5.397</td>
<td>2.36</td>
<td><strong>1.354</strong></td>
<td>1.0</td>
</tr>
<tr>
<td>Minimum</td>
<td>4.895</td>
<td>2.232</td>
<td>1.308</td>
<td>1.0</td>
</tr>
<tr>
<td>Maximum</td>
<td>5.803</td>
<td>2.488</td>
<td>1.426</td>
<td>1.0</td>
</tr>
<tr>
<td>Standard dev.</td>
<td>0.183</td>
<td>0.082</td>
<td>0.054</td>
<td>undefined</td>
</tr>
<tr>
<td>Amplitude</td>
<td>0.908</td>
<td>0.256</td>
<td>0.118</td>
<td>0</td>
</tr>
<tr>
<td>Sample size</td>
<td>50</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.5: $n_p = 800$, $n_v = 160$, $A = 1$
3.5.1 Ameliorations for the simulated annealing process

\section*{\textbf{\underline{\Delta} computation in the relative arrival times problem}}

Among the different implementations proposed to solve the different problems, the one corresponding to the ride sharing problem with relative arrival times could be obviously significantly improved as far as the \textbf{\underline{\Delta}} computation is concerned. The client encountering the greatest relative arrival time among the clients served by a given vehicle is not necessarily the one who arrives at destination in last position. Furthermore, a modification of a part of the route via a shift or an interchange move has a different impact on the relative arrival time of each concerned client. This impact is easily computed for all the concerned clients despite it requires for each advanced/delayed arrival to retrieve the length of the origin-destination trip of the client and to divide the \( \delta \) value by these value for each of them. A possibility is to handle a vector containing all the relative arrival times corresponding to the current solution and to look if the proposed move induces a change in the maximum value of this vector. It would also be interesting to consider the minimisation of the mean relative arrival times. To sum up, it is possible to enhance the efficiency of this implementation, but the number of computations that would need to be performed does not allow obtaining a resolution as fast as for the car sharing problem with absolute waiting times.

\section*{Release times problem: possibility for the vehicles to start their route at different moments}

As it will be discussed in Chapter 4, the release time constraints, if strictly applied, can lead to an under-utilisation of the fleet. Because starting all the routes at \( t = 0 \) automatically leads to the fact that some people are served too early, some vehicles simply do not leave their depot in the obtained solutions. A solution can be to allow them to start at different times, which would be variables of the problem. In order to handle them with the discrete optimisation tools used so far, thes starting times should be discretised, allowing for example departures every minute. In order to obtain a very realistic model, it would be necessary to allow the vehicles to stop for a given moment at some points. One could therefore add for each point a break laps of time variable, but this renders the model very complicated to solve with the tools presented so far.

\section*{Reducing the size of the solution space}

As mentioned in the beginning of this chapter, given an evenly distribution of the clients among the vehicles (in the case of the car sharing problem), the possibilities of interchanging the position of the clients inside the vehicle routes among for a multiplication factor \( \left( \frac{n_c}{n_v} \right)^{n_v} \) in the computation of the number of possible solutions. The very large values of this factor (cf 3.1) encourages the consideration of a simulated annealing procedure for which shift moves would only consist in removing one client from a chosen vehicle and to insert it at the optimal place in another vehicle. Each time a vehicle undergoes a move, the its route would be automatically re-optimised. In the case where only the sum of the waiting (car sharing) or arrival times (ride sharing) are to be minimised, the routes can simply be optimised according to the TRP. The times required to solve TRP from scratch with a simulated annealing for a sequential service of the clients is about 0.2 seconds for \( n = 10 \). The performances as a function of \( n \) are shown in Tables 3.7 and 3.6. However it may be not necessary to solve a TRP from scratch when inserting, removing or changing one point of a route. If the maximum waiting/arrival times is taken into account, the procedure would be more complicated as the global objective function cannot trivially be transposed to each single vehicle. An idea could be to alternate sequences of iteration of the simulated annealing procedure as presented in section 3.3 and sequences of the reduce space with an optimisation process inside the vehicles that uses a modified objective
function (e.g. the combined MinSum-MinMax problem with a weight factor equal to \( \frac{\text{nb_points}}{n_p} \) \( W \) if vehicle \( k \) is concerned).

The idea it to be able to reduce significantly the number of required SA iteration while making them more expensive in term of computation time in the hope to get net computation time benefits.

### 3.5.2 Pre-processing of the data (clustering)

In the hope to get better results and/or to reduce the required computation time, the idea of solving several small problems instead one single large problem involving \( n_p \) clients and \( n_v \) vehicles can be considered. Such an approach is essential to solve very large instances of the problems. It allows to perform multi-thread resolutions or even to reduce the sequential computation time as the number of iterations necessary to obtain a good solution increases more than linearly with the problem size. The idea is to use common sense in order to restrict the solutions browsed of the simulated annealing procedure to a subset of solutions that one would intuitively consider as good. The disadvantage of such a restriction is that it potentially remove the possibility to reach very good solution which are not intuitive at all and that would thus not have been included in the aforementioned subset. It is thus expected that experimentation takes an important place when studying this approach.

Concretely, the idea is to divide the set clients in several groups or *clusters*, and to attribute a given set of vehicles to each cluster. In other words, clients of different clusters cannot use the same vehicles, and the optimisation problem can then be solved for each cluster with the simulated annealing method independently, and thus possibly in parallel, which can lead to very high computation time reduction. A complete study of the efficiency of these clustering methods would imply on one hand the choice of the strategy for grouping the clients and on the other hand the study of best distribution of the available vehicles among the different groups.

<table>
<thead>
<tr>
<th>( n )</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1.336</td>
<td>1.169</td>
<td>1.162</td>
<td>1.137</td>
<td>1.087</td>
</tr>
<tr>
<td>Minimum</td>
<td>1.081</td>
<td>1.029</td>
<td>1.014</td>
<td>1.0</td>
<td>1.013</td>
</tr>
<tr>
<td>Maximum</td>
<td>2.299</td>
<td>1.644</td>
<td>1.416</td>
<td>1.318</td>
<td>1.183</td>
</tr>
<tr>
<td>Standard dev.</td>
<td>0.198</td>
<td>0.099</td>
<td>0.076</td>
<td>0.076</td>
<td>0.046</td>
</tr>
<tr>
<td>Amplitude</td>
<td>1.218</td>
<td>0.615</td>
<td>0.402</td>
<td>0.318</td>
<td>0.17</td>
</tr>
<tr>
<td>Sample size</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 3.6: TRP with 30 cities. For \( n = 10 \), this problem is solved in 0.2 seconds

<table>
<thead>
<tr>
<th>( n )</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1.078</td>
<td>1.044</td>
<td>1.029</td>
<td>1.025</td>
<td>1.015</td>
</tr>
<tr>
<td>Minimum</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Maximum</td>
<td>1.138</td>
<td>1.137</td>
<td>1.1</td>
<td>1.1</td>
<td>1.072</td>
</tr>
<tr>
<td>Standard dev.</td>
<td>0.04</td>
<td>0.038</td>
<td>0.026</td>
<td>0.022</td>
<td>0.017</td>
</tr>
<tr>
<td>Amplitude</td>
<td>0.138</td>
<td>0.137</td>
<td>0.1</td>
<td>0.1</td>
<td>0.072</td>
</tr>
<tr>
<td>Sample size</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 3.7: TRP with 10 cities
Definition of the distance between clients

It is essential to define a distance between each possible pair of clients, the idea being that two clients separated by a too large distance cannot travel together and should be put in different clusters. This distance is not necessarily an Euclidian distance geographical points, but rather an abstract distance that can be defined following different strategies. Then, the clusters can be determined by solving the Facility Location Problem. If we choose to determine \( n_c \) clusters, this problem amounts to choose \( n_c \) centroïds among the set of clients (the centroïds correspond to the facilities) and to attribute each client to one of the centroïds. The objective is to minimise the sum of the distance of each client to their corresponding centroïd, taking into account constraints about the minimum size of the clusters.

Similar trips strategy

The most basic idea to create clusters is to group together clients that need to perform a similar trip. In this vision, two clients are considered as close to each other if both their starting points and destination points are respectively close to each other. One can thus define the distance between clients \( i \) and \( j \) with the arithmetic mean:

\[
D_{ij} = \frac{1}{2} \left( C_{i\text{or},j\text{or}} + C_{i\text{dest},j\text{dest}} \right).
\]

By defining this distance, it is expected that the proposed solutions will include routes constituted of a series of pick up in an area and, after a trip with a full vehicle, a series of clients drops. An example of clustering realised with this strategy for an artificially generated set of data modelling a morning peak hour situation is shown in Figure 3.13.

![Figure 3.13: Example of clustering for an artificially generated set of data modelling a morning peak hour with 100 clients with the similar trip strategy. The number of cluster is 5 and they are constrained to a minimum size of 5 clients. Each color represents a cluster. The stars represent origin points and the diamonds represent destination points.](image)

Successive trips strategy

Taking into account the fact that it is also possible for the vehicles to take new clients after dropping others, the situation where the starting points of some clients are situated near the destination point of another one who will use the same vehicle is interesting. One can define the distance between two clients as

\[
D_{ij} = \min \left( \frac{1}{2} \left( C_{i\text{or},j\text{or}} + C_{i\text{dest},j\text{dest}} \right), C_{i\text{dest},j\text{or}}, C_{i\text{or},j\text{dest}} \right).
\]
Demand corridors

By contrast with the previous ones, this definition of distance is based on geometric characteristics. Recall that in the framework of this work, it has been decided to work only with distances between points that could without loss of generality be replaced real displacement times in by taking speed limitations and the road network geometry into account. The following approach, however, is expected to work only in the framework of simple Euclidian distances.

In order to put in evidence the possibility to pick up or drop a client along a journey, we consider that two clients are also close to each other when the smallest distance between the starting or destination point of one of these clients and the line segment describing the demand journey of the other is small and both segments are oriented in a similar way (both clients travel in a similar direction). The idea is to group together clients in corridors. The encapsulation of demand into corridors is a concept that is also used for the design of classical public transport networks [14].

A graphical illustration of the three presented strategies is presented in Figure 3.14.

![Figure 3.14: Illustration of the parameters taken into account in the different method for defining the distance between two client characterised by their origin-destination vectors (black arrows). a) Similar journey strategy, b) Successive trips strategy, c) Corridors strategy. The colors represent the distances / angle taken into account.](image-url)

Further investigations

The tests performed so far on a set of real data in the framework of a very general and heterogeneous demand in a whole region did not lead neither to an amelioration of the results nor a to a decrease of the computation times. However, in dense population areas, it is not excluded that the method would present good results. The aforementioned tests have been performed by distributing the vehicles evenly among the clusters but the link between the sum of the distances between each clients of a cluster and the corresponding centroïd (or the average distance between the clients of a cluster) and the ideal size of the fleet to attribute to this cluster is still to be investigated.
Chapter 4

Results analysis

The methods mentioned in Chapter 3 are now used to analyse qualitatively and quantitatively the solutions proposed by the optimisation process that has been presented in Chapter 1. The solver is able to solve the following problem: we consider a fleet of public vehicles serving a set of clients, and each client requires one vehicle to achieve a trip from their origin point to their destination point. The vehicles all start their tour at the same time, either from a given a point referred as a depot, or from the origin point of a client considered as optimal by the process. In the latter case, the starting point is said to be free. The vehicles do not have the possibility to stop until the end of their tour and they all ride at the same speed. In order to fix the ideas, the unique speed of 60 kilometres per hour will be considered. One can thus refer to a same vehicle displacement by saying it is 10 km long or that it lasts 10 minutes. Finally, we distinguish car sharing problems from ride sharing problems. In the former case, a vehicle can transport only one client at once and it serves the different clients sequentially while, in the second case, it is possible to carry several clients in a same vehicle provided the capacity of the vehicle is not exceeded. In the car sharing problem, the journey of a client is identical to the one they would have followed with a personal car. On the other hand, they are likely to make detours in a ride sharing situation.

4.1 Solved problems definitions and notations

In practice, four different particular cases of the problem were implemented in the software:

1. The car sharing problem with simultaneous demand where a linear combination of the sum (and thus the average) of the waiting times and the maximum waiting time is minimised.

2. The ride sharing problem with simultaneous demand where a linear combination of the sum (and thus the average) of the waiting times and the maximum arrival time is minimised.

3. The ride sharing problem with simultaneous demand where a linear combination of the sum (and thus the average) of the arrival times and the maximum relative arrival time is minimised.

4. The ride sharing problem with time-distributed demand. Only the average arrival time is minimised, which is equivalent to minimize the average laps of time between the desired departure time of a client and (referred as the release time) their arrival time.

The different parameters that can be chosen for solving the problems for a given number \( n_p \) of trip demands specified by an origin point, a destination point and for a wished starting time (only in Problem 4) are

\[ 1 \text{Recall that the relative arrival time of a client refers to their arrival time divided by the length of their trip.} \]
The number $n_v$ of available vehicles;
- The capacity $N_{\text{max}}$ of the vehicles (except for Problem 1);
- The value of the MinMax weight factor ($W$ for Problems 1 and 2, $W_{\text{rel}}$ for Problem 3);
- The starting points (depots) of the vehicles. They can be either chosen randomly among the origin points of the clients, or left free, i.e. as a part of the problem to be solved.

The variables that can be observed to analyse a routing are for $i \in \{1, ..., n_p\}$ and $j \in \{1, ..., n_v\}$
- The waiting time $w_i$ of client $i$;
- The arrival time $a_i$ of client $i$ at their destination;
- The length of the journey $l_i$ of client $i$;
- The distance travelled by the vehicles $T_k$;
- The number of clients served by a vehicle $\text{nb\_clients}_k$;
- The passenger load of each vehicle along time, i.e. the number of clients sitting this vehicle $\text{Pop}_k(t)$;
- The time weighted average passenger load of the vehicles $\mu(\text{Pop}_k) = \frac{1}{T_k} \int_0^{T_k} \text{Pop}_k(t)dt$;
- The cumulated riding times of all the vehicles $\sum_{k=1}^{n_v} T_k$, and the economy factor $E = \frac{\sum_{k=1}^{n_v} T_k}{\sum_{i=1}^{n_p} \text{trip}_i}$;
- The global average vehicle passenger load: $M(\text{Pop}) = \sum_{k=1}^{n_v} T_k \mu(\text{Pop}_k)$;
- The advance $\text{ADV}_i$ with which client $i$ is served in Problem 4. A feasible solution corresponds to negative advances only. If $\text{ADV}_i \leq 0$, $w_i = -\text{ADV}_i$.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Name</th>
<th>Simultaneity of the demand</th>
<th>Objective function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>Car sharing</td>
<td>Yes</td>
<td>$\sum_i w_i + W \max_i w_i$</td>
</tr>
<tr>
<td>Problem 2</td>
<td>Ride sharing</td>
<td>Yes</td>
<td>$\sum_i a_i + W \max_i a_i$</td>
</tr>
<tr>
<td>Problem 3</td>
<td>Ride sharing</td>
<td>Yes</td>
<td>$\sum_i a_i + W_{\text{rel}} \max_i \text{trip}_i$</td>
</tr>
<tr>
<td>Problem 4</td>
<td>Ride sharing</td>
<td>No</td>
<td>$\sum_i a_i + W_{\text{adv}} \text{ADV}_i$</td>
</tr>
</tbody>
</table>

Table 4.1: Summary of the problems solved by the current software. $i$ always in the set of clients.

A summary of the different problems with their corresponding weighted sum-based objective function is presented in Table 4.1. Recall that in Problem 4, the goal is to minimise $\sum_i a_i - \text{READY}_i$ where $\text{READY}_i$ is the time by which client $i$ wants to leave its origin point, which is equivalent to minimise $\sum_i a_i$ because $\sum_i \text{READY}_i$ is a constant. The term in $W_{\text{adv}} \text{ADV}_i$ in the table allows to avoid cases where vehicles pick up clients too early.

The rest of this chapter will consist in the analyse of several aspects of the solutions proposed by the software for the different problems. The set of data used (i.e. the set of origin-destination pairs) will be in some cases a real and totally general demand covering a whole region containing several cities [10] and in some other cases artificially generated random data in order to put some phenomena in evidence. When talking about the real data set for a given number of clients $n_p$, it means the first $n_p$ clients of the list.
4.2 Influence of the weight factors \( W \)

In this section, we study the influence of the \( W \) parameter appearing in the objective functions of the different problems on the solutions. The goal is to assess the ability of such parametric objective functions to reflect the degree of importance attributed to the minimisation of the maximum waiting/arrival times chosen by the operator.

4.2.1 Evolution of the maximum waiting time with \( W \) (Problem 1)

It is expected that when increasing the \( W \) parameter, the maximum waiting time (in the framework of Problem 1) gets lowered to the detriment of the average waiting times. We define the parameter \( A \) such that \( W = AW_{\text{ref}} \), where \( W_{\text{ref}} \) is a reference value of the parameter. It is chosen to be equal to \( n_p/2 \). A reason for this choice is that in the objective function, the sum term can be seen as being equal to \( n_p \) times an average characteristic time \( t^* \), while the maximum time can be considered to be of the order of magnitude of \( 2t^*n_p \). When \( W = W_{\text{ref}} \), both terms are supposed to be roughly equal. Note that \( t_{\text{max}} = 2t^*n_p \) corresponds to an idealised case where it is possible to obtain a very good fairness (i.e., to have a low difference between the average and the maximum waiting times). In practice, the value of \( A \) corresponding to putting the average waiting time and the maximum waiting time on an equal foot (call it \( A_{\text{eq}} \)) is most of the time smaller than 1.

![Figure 4.1: Maximum waiting times for different values of \( A \) obtained with the set of real data with \( n_p = 100 \) and \( n_v = 20 \). The blue points correspond to the 5 sample results and the red ones to their average for each value of \( A \). The starting point of the vehicles is left as free.](image)

Figure 4.1 shows the different values of the maximum times obtained along the experiments. Figure 4.1a shows that even for very small values of \( A \), the maximum time (each time averaged over the five obtained values) decreases quickly towards a value (about 50 min) before stabilising. Then, in Figure 4.1b (for which the \( A \) scale is logarithmic), one sees that they decrease much more slowly before stabilising for large values of \( A \).

\(^2\)The values presented are \( A = 0.2, 0.5, 2, 20 \) and 200.
The strong decrease of the maximum times when going from $A = 0$ to $A = 0.02$ can be explained by the fact that there exist solutions for which the maximum waiting is significantly reduced with respect to the ones obtained with $A = 0$ without significantly increasing the average waiting time. A weak value of $A$ is thus sufficient to make these solution appear. This phenomenon is to be put in relation with the large discrepancies in the set of trip demands (see Figure 4.2). According to Figure 4.1b, the largest decrease of the waiting times (apart for the aforementioned one) happens for $A$ values of orders of magnitude of $0.1$ and $1$, which is consistent with the previous considerations.

![Figure 4.2: Histogram of the length of the trip demands (origin-destination of the 100 clients of the real data set).](image)

So far, only values averaged over the different experiments were discussed. However one can see that for each mentioned value of $A$ (except 0), there is a large deviation between the different experiments (amplitude up to 18% in the data presented in Figure 4.1). This makes very difficult to obtain a reliable control of the priority attributed to the minimisation of the maximum time. The reason are the inability of the used metaheuristic procedure to converge towards a unique solution and the fact that even for similar values of the objective function, it is possible to have different types of solution with different repartition of the objective contribution of the two terms. This situation happens for moderate $A$ values. In order to avoid this phenomenon, one could use the tools of multi-objective optimisation, for which there exist suitable simulated annealing algorithms [22].

Knowing that there is always uncertainty about the priority effectively attributed to the minimisation of the maximum waiting time, one has to settle with empirical rules of good practices: ut $A = 0$ to totally neglect the maximum waiting times, take $A \simeq A_{eq}$ to attribute them a moderate priority and take $A >> A_{eq}$ to neglect the average waiting times.

All these considerations can be extended to Problems 2 and 3 with adapted definitions of $A$. For Problem 3, taking $W_{rel} = \frac{1}{2} \sum_{i=1}^{n_p} \text{trip}_i$ is enough to reach the saturation of the maximum relative arrival time (around 4 against 24 for $W_{rel} = 0$ with the same instance of the problem, for ride sharing with 5-place vehicles).

### 4.2.2 Penalty factor of Problem 4

In our model of the ride sharing problem with release time (Problem 4), the vehicles do not have the possibility to stop in order to wait for a client to be ready. If a large penalty coefficient $W_{adv}$ is chosen, this leads the optimisation process to a solution where some available vehicles do not leave their depot in order to avoid serving any client in advance. In order to avoid this situation, it is convenient to choose a reduced value of $W_{ADV}$, which allows a reasonable advance for a few clients when it is beneficial for the system. For the real data set with $n_p = 100$ and
$n_v = 20$, the histogram of the advance times are shown respectively in Figures 4.3a and 4.3b for $W_{adv} = 10$ and $W_{adv} = 100$ respectively. Using $W_{adv} = 50$ represents a good compromise (all the vehicles are used and the advance times are kept reasonable). Note that allowing advance times will lead to average arrival times\(^3\) lower than what would be observed if the vehicles had the possibility to stop.

Figure 4.3: Histograms of the advance times $ADV_i$. These results were obtained with the real origin-destination data with release times evenly distributed on a 30-minute lap of time, with $n_p = 100$ and $n_v = 20$. Negative advance times correspond to waiting times.

4.2.3 Quality of service as a function of the distance from the center

As mentioned in the Chapter 1, the main problem when using a system of shared vehicles is that the clients situated far from the city centres are most of the time disadvantaged. In this subsection, we aim at quantifying the correlation between the waiting/arrival times and the distance that separate the clients from a city center. In order to put this relation in evidence, we decided to focus on a demand corresponding to one single artificial large city in a morning peak situation, i.e. with a strong tendency of the clients to require trips from the outskirts towards the center. Such an artificial demand can be generated by creating random origin and destination points with polar coordinates $(r, \theta)$, where $\theta$ is a random variable uniformly distributed in $[0, 2\pi]$ and where $r$ is the absolute value of a random variable that follows a normal law centred in 0 and of which the standard deviation can be chosen. These standard deviations are denoted by $\sigma_{\text{origin}}$ and $\sigma_{\text{destination}}$. In this section, we focus on the case where $\sigma_{\text{origin}} = 50$ km and $\sigma_{\text{destination}} = 20$ km. The corresponding situation is represented in Figure 4.4.

Figure 4.4: Artificial demand for a schematic city during morning peak with $\sigma_{\text{origin}} = 50$ km and $\sigma_{\text{destination}} = 20$ km.

\(^3\)by arrival times it is meant $a_i - \text{READY}_i$ in the framework of Problem 4
Figure 4.5: Simultaneous view of the origin points distances from the center and the waiting
times for the artificial demand with $n_p = 100$ and $n_v = 20$. Obtained with $n = 10000$.

Problem 1 was solved for this set of demands for different values of $A$. In the graphs
presented in Figure 4.5, each point represents one client. The results shown in Figure 4.5a
reflect a correlation between both variables when giving priority to the minimisation of average
waiting times. The Spearman correlation coefficient is equal to 0.51 for $A = 0.01$. When
using a large $W$ value, one can notice that the correlation nearly disappears, as the correlation
coefficient decreases to 0.27 in Figure 4.5b for $A = 2$. One can clearly notice that in Figure 4.5b,
the waiting times for the people with an origin point at less than 50 kilometres from the center
have their waiting time increased (nearly doubled for some of them) with respect to Figure 4.5a.
The experiment was conducted for different values of $A$, five times for each value. The average
Spearman correlation coefficients for each value of $A$ are presented in Figure 4.6.

Figure 4.6: Evolution of the Spearman coefficient of correlation between the waiting time and
distance from origin point to the city center as a function of the $A$ parameter. For each value
of $A$, the coefficient is averaged on 5 experiments conducted with $n = 10000$.

About the influence of weight coefficients appearing in the weighted sum objective functions,
one can conclude that if the effect is unequivocal for extreme values, a refined control of the
importance of the different terms of the objective functions appears to be much less evident.
Empirical good practice rules can be used, and the values of the weight coefficient that lead to
an equivalence between the two terms are strongly depending of the data of the problem (e.g.
large discrepancies in the trips demand for problem 3).
4.3 Influence of the demand density on the quality of the service. Modelling with problems 2 and 4.

4.3.1 Influence of the density of the demand

In this section, we work on artificial data and build a demand corresponding to a city with the same characteristic sizes as the one presented in the previous section (and represented in Figure 4.4): \( \sigma_{\text{origin}} = 50 \text{km} \) and \( \sigma_{\text{destination}} = 20 \text{ km} \). It is thus once again a morning peak situation. The idea is to generate three different sets of demand with different numbers of trips requirement (we take \( n_p = 50, 100 \) and 200) while keeping the \( n_p/n_a \) ratio constant (\( n_v = 10, 20 \) and 40 respectively). For all the experiments of this section, the starting point of the vehicles is let as free. This instance was solved using three different approaches: Problem 2 with \( W = 0 \) and \( W = 100 \) and Problem 4 for \( w_{\text{adv}} = 50 \). For this last problem, the release are times randomly generated and spread over 30 minutes. It is supposed that the capacity of the vehicles was of 5 passengers. However, it has been noticed that when performing these experiments with a very large value of \( N_{\text{max}} \), which makes the computations faster (see Chapter 3), the number of passengers hardly ever exceeds 4 5. The results are presented in Tables 4.2, 4.3 and 4.4. Observing these three tables, one can see that the average waiting times and arrival times are systematically decreased when the density of the demand increases. This reflects the fact that with a larger demand density, if the available resources are increased accordingly, there are locally more possibilities of routing which allows to find better solution. The fact that the maximum arrival time does not decrease systematically when the density increases when solving Problem 2 with \( W = 100 \) (Problem 2 being the only one among the three where maximum arrival time appears in the objective function) comes from the differences in the longest required trips among the three instances. As matter of fact, when one looks at the maximum relative arrival times for this problem, one obtains 2.16 min, 2.41 min and 2.62 min for \( n_p = 200, 100 \) and 50 respectively. Note that the mean length of the required trips for these three instances is respectively of 42, 46 and 48 minutes.

4.3.2 Other considerations

Besides the study of the impacts of the density, it is visible that solving Problem 2 with \( W = 100 \) is the approach that leads to the most economical routing. This can be understood intuitively since minimising the maximum arrival time means that the maximum length of the routes is also minimised. Note that when solving Problem 4, the economy factor can get larger because of the detours that vehicles need to do in order to avoid taking people not ready to go. To put this phenomenon in evidence the an instance of the problem was run with \( n_p = 100 \) and \( n_v = 20 \) but with release times spread over 80 minutes. The economy factor increases then to 0.94, i.e. this situation is nearly even bad as if each client used their own vehicle. Typical trajectories of the vehicles in such situations are presented in Figure 4.7a, to be compared with Figure 4.7b, which are trajectories obtained with Problem 2 (\( W = 100 \)) and where the vehicles exhibit a tendency to go from the outskirts to the center in a more straightforward way. Note that the vehicle represented by a red curve in Figure 4.7b is charged to perform an inner peripheral tour.

The temporal aspect of vehicles tours and the journeys of the corresponding clients can also be represented graphically via a set of time lines. In Figure 4.8, time lines for the 5 clients using the vehicle represented by a green curve in Figure 4.7a are represented. They allow to visualise the laps of time the clients would have spend in their private vehicles if not using the public transport system and time by which they begin and end their journey in the public vehicle.

\[ \sum \alpha_{3 \text{km}} T_k \]

Among all these experiments, the largest fraction of the total riding time \( \sum_{k=1}^{n_v} T_k \) spent with a number of passengers larger than 5 is of 2.5 %. It is believed that imposing the capacity constraint would not change the results in a significant way. The computations were thus performed with \( N_{\text{max}} = 50 \).
(a) Solving Problem 4 with $W = 50$ and release times evenly distributed over 80 min.

(b) Solving Problem 2 with $W = 100$ (simultaneous demand).

Figure 4.7: Some typical trajectories of vehicles for the case of the artificial city in peak hours with $\sigma_{\text{origin}} = 50 \text{km}$, $\sigma_{\text{destination}} = 20 \text{ km}$, $n_p = 100$ and $n_v = 20$. Each colour corresponds to one vehicle. The circles represent the (free) starting points, the stars represent the origin points of the clients and the diamonds their destination points. The constraints on the release prevent the vehicle to use straightforward routes to go to the center.

<table>
<thead>
<tr>
<th>$n_p$</th>
<th>200</th>
<th>100</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean arrival times (min)</td>
<td>74.0</td>
<td>84.9</td>
<td>98.1</td>
</tr>
<tr>
<td>Max. arrival times (min)</td>
<td>252</td>
<td>236</td>
<td>291</td>
</tr>
<tr>
<td>Occupation rate M(Pop)</td>
<td>2.13</td>
<td>2.00</td>
<td>1.67</td>
</tr>
<tr>
<td>Economy factor</td>
<td>0.57</td>
<td>0.63</td>
<td>0.76</td>
</tr>
</tbody>
</table>

Table 4.2: Results for Problem 2 ($W = 0$)

<table>
<thead>
<tr>
<th>$n_p$</th>
<th>200</th>
<th>100</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean arrival times (min)</td>
<td>75.6</td>
<td>86.2</td>
<td>106.2</td>
</tr>
<tr>
<td>Max. arrival times (min)</td>
<td>156</td>
<td>161.1</td>
<td>159.0</td>
</tr>
<tr>
<td>Occupation rate M(Pop)</td>
<td>2.37</td>
<td>2.27</td>
<td>2.19</td>
</tr>
<tr>
<td>Economy factor</td>
<td>0.52</td>
<td>0.57</td>
<td>0.63</td>
</tr>
</tbody>
</table>

Table 4.3: Results for Problem 2 ($W = 100$)

<table>
<thead>
<tr>
<th>$n_p$</th>
<th>200</th>
<th>100</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean arrival times (min)</td>
<td>75.3</td>
<td>82.9</td>
<td>104.2</td>
</tr>
<tr>
<td>Max. arrival times (min)</td>
<td>180</td>
<td>176.2</td>
<td>176.0</td>
</tr>
<tr>
<td>Occupation rate M(Pop)</td>
<td>1.95</td>
<td>2.13</td>
<td>2.09</td>
</tr>
<tr>
<td>Economy factor</td>
<td>0.65</td>
<td>0.61</td>
<td>0.69</td>
</tr>
</tbody>
</table>

Table 4.4: Results for Problem 4 ($W_{\text{adv}} = 50$) with release times evenly distributed over 30 minutes.
Figure 4.8: Time lines for the clients served by the vehicle with trajectory represented in green in Figure 4.7a. Client 1 is served slightly before their release time. Client 4 and 5 leave their origin points by a time at which they would already be arrived at destination with their private car.

4.4 Quality of the service as a function of the available fleet

The question of the practical application of the car sharing or ride sharing system was up to now not addressed. In this section, we try to determine, in a given situation, the size of the fleet required in order to obtain reasonable waiting and arrival times. The usefulness of such a system in areas that are subject to traffic congestion is also discussed.

The problem is run on an artificial set of demands representing a city similar the one used so far, but with smaller characteristic dimensions ($\sigma_{\text{origin}} = 25$ km and $\sigma_{\text{destination}} = 10$ km) for $n_p = 100$, and several sizes of fleet are tried. The obtained results are shown in Tables 4.5, 4.6 and 4.7. The departure position of the vehicles is free. All the situations are approached with the same three problems as in the previous section (Problem 2 with $W = 0$ and 100 and Problem 4 with $W_{\text{adv}} = 50$.) The average length of the required trips is 20 km. The experiments are performed with infinite capacities, but the results in the hypothesis of 5-places vehicles are valid for all the experiments (the number of passenger in the vehicles do not exceed 5 for more than 3 % of the times), except for $n_v = 10$, where the number of passengers exceeds 5 during up to 14 % of the time depending from the approach used.

<table>
<thead>
<tr>
<th>$n_v$</th>
<th>30.0</th>
<th>25.0</th>
<th>20.0</th>
<th>15.0</th>
<th>10.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean waiting times times (min)</td>
<td>6.72</td>
<td>9.61</td>
<td>12.9</td>
<td>18.5</td>
<td>29.47</td>
</tr>
<tr>
<td>Max. waiting times (min)</td>
<td>27.42</td>
<td>37.9</td>
<td>71.8</td>
<td>79.1</td>
<td>118.45</td>
</tr>
<tr>
<td>Mean arrival times (min)</td>
<td>29.5</td>
<td>33.1</td>
<td>37.1</td>
<td>45.04</td>
<td>58.5</td>
</tr>
<tr>
<td>Max. arrival times (min)</td>
<td>65.9</td>
<td>75.2</td>
<td>116.5</td>
<td>127.1</td>
<td>161.04</td>
</tr>
<tr>
<td>Occupation rate $M(\text{Pop})$</td>
<td>1.81</td>
<td>1.91</td>
<td>1.9</td>
<td>2.2</td>
<td>2.25</td>
</tr>
<tr>
<td>Economy factor $E$</td>
<td>0.62</td>
<td>0.61</td>
<td>0.63</td>
<td>0.6</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Table 4.5: Impact of the size of the fleet on the service for a morning peak hour configuration for a randomly generated demand corresponding to a city with dimensions $\sigma_{\text{origin}} = 25$ km and $\sigma_{\text{destination}} = 10$ km. Results for Problem 2 ($W = 0$)
#### Table 4.6: Impact of the size of the fleet on the service for a morning peak hour configuration for a randomly generated demand corresponding to a city with dimensions $\sigma_{\text{origin}} = 25$ km and $\sigma_{\text{destination}} = 10$ km. Results for Problem 2 ($W = 100$).

<table>
<thead>
<tr>
<th>$n_v$</th>
<th>30.0</th>
<th>25.0</th>
<th>20.0</th>
<th>15.0</th>
<th>10.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean waiting times (min)</td>
<td>7.69</td>
<td>9.58</td>
<td>14.34</td>
<td>19.03</td>
<td>32.5</td>
</tr>
<tr>
<td>Max. waiting times (min)</td>
<td>36.5</td>
<td>32.94</td>
<td>46.27</td>
<td>72.01</td>
<td>78.6</td>
</tr>
<tr>
<td>Mean arrival times (min)</td>
<td>30.72</td>
<td>33.37</td>
<td>39.28</td>
<td>46.32</td>
<td>66.2</td>
</tr>
<tr>
<td>Max. arrival times (min)</td>
<td>49.16</td>
<td>58.9</td>
<td>62.95</td>
<td>86.62</td>
<td>105.37</td>
</tr>
<tr>
<td>Occupation rate $M(\text{Pop})$</td>
<td>1.75</td>
<td>1.95</td>
<td>2.13</td>
<td>2.43</td>
<td>3.29</td>
</tr>
<tr>
<td>Occupation rate $M(\text{Pop})$</td>
<td>0.65</td>
<td>0.6</td>
<td>0.58</td>
<td>0.56</td>
<td>0.51</td>
</tr>
</tbody>
</table>

A general tendency that appears is that on average, when increasing $n_a$, the fraction of the arrival time of a client that is devoted to waiting is reduced and does not exceed one quarter for $n = 30$. Even 30 vehicles, the average relative arrival time (i.e. the factor by which the total length of the trip is multiplied when using the system with respect to the use of a private car) is only reduced to 1.67,1.84 and 1.61 for Problem 2 with $W = 0$, Problem 2 with $W = 100$ and Problem 4 with $W_{\text{adv}} = 50$ respectively. These quantities are not acceptable for such a large $n_p/n_v$ ratio. The use of this system becomes however advantageous when one consider its generalised use on road axis subject to congestion. As a matter of fact, for Problem 2 with $W = 0$ for example, the economy factor is of 0.62. In other words, the total travelled number of vehicle kilometres during the observed peak hour period using the system is reduced to 62 % (and even less if we consider the density of demand effect) of what it would have been in the case where all the client would use their private car. As in this peak hour situation, all the clients aim at travelling at the same time, it is safe to say that the number of vehicles on the roads at each moment would be reduced to about 62 %.

Usually, the capacity of a road axe is of 30 vehicles per minute per traffic lane.\(^5\). According to the congestion function developed by the Bureau of Public Routes (BPR), the time $T$ required by a vehicle to cross this axe as a function of the traffic volume $v$ (in vehicles per minutes) attempting to cross this arc is given by [11]

$$T(v) = T_0 \left(1 + 0.15 \left(\frac{v}{c}\right)^4\right), \quad (4.1)$$

\(^{5}\) As a matter of fact, at any speed, the security laps of time between two successive cars should be of 2 seconds [4]
with $c$ the capacity of the axe (in vehicles per minute) and $T_0$ the time required to cross the axe in conditions of normal situation. Taking $c = 30$ vehicles per minute and for an axe for which $T_0 = 1$ min, one obtains the curve presented in Figure 4.9. According to this function, if $v \approx 50$ vehicles per minute, reducing the volume by 60 % would multiply the speed of the traffic by a factor 2. In these conditions, there is a gain to use of the demand responsive transit system, but one should keep in mind that this is possible only if a significant part of the population abandon their private car in favour of this system.

![Figure 4.9: Saturation of a road axe according to the BPR function compared to the infinite capacity hypothesis used in the model. The traffic volume in normal conditions is of 30 vehicles per minute.](image-url)
Chapter 5

Conclusions and perspectives

The objective of this work was to develop an optimisation process in order to determine the routes of a fleet of public vehicles responding to a set of a priori specified demands for individual trips. These public vehicles are supposed to be not driven by the client (common taxis or self-driving cars able to travel without passengers). Both the cases of car sharing (vehicles able to transport one client) and ride sharing (vehicles transporting several clients at the same time) were envisaged. The optimisation was based on the minimisation of the waiting times of the clients with the concern to obtain a routing of the vehicles both fast and fair. Variants of the problem with a minimisation of the maximum factor by which a client multiplies the length of their personal-vehicle-based trip when using the system and with accounting of various release times were also envisaged.

We showed that these problems can be modelled as Vehicle Routing Problems for which the objective function, which classically represents sum of the costs for the operator, is replaced by a linear combination of the sum of the waiting / arrival times and their maximum.

Solving these problems even for small instances requires the use of heuristic or metaheuristic methods. The simulated annealing metaheuristics was used in the framework of a local search procedure for a small neighbourhood. The method appears to give solutions for which the objective function converges towards a given value when the number of simulated annealing iterations increases and to solve better the problems in the limit case where only the average waiting/arrival time is to be minimised. A precise control of the priority accorded to the minimisation of the maximum waiting / arrival time (very high or absent priority).

Applying the optimisation process to sets of trips demand, it was possible to put in evidence the fact that when according a high priority to the maximum waiting time, the discrepancies between the quality of the service delivered for the clients from the outskirts of a city with respect to the ones of the center can be reduced. The efficiency of the system also appears to increase with density of population for a fixed number of clients/number of vehicles ratio. Finally, in order to be beneficial for the clients, the system should be used massively on road axis subject to congestion in order lower the traffic volume and to increase the commercial speed. This would compensate the time spent waiting for a vehicle and devoted to detours. This naturally implies the use of vehicles capable of transporting several clients at the same time, car sharing being only useful to decrease the number of vehicles but would logically increase the traffic volume.

This introductory work on the optimisation of a demand-responsive transit system paves the way for further research. On a modelling point of view, it is interesting to explore the possibilities offered by multi-objective optimisation to obtain a better control of the impact of the priority attributed to the minimisation of maximum times.

On the methodological point of view, beyond the ways for improvement of the method sugested in Chapter 3, some more sophisticated method could be envisaged. Laporte [12], for example, developed a powerful simulated annealing-based method for solving the CCVRP (VRP minimising the sum of the visit times) which could potentially be extended to the problems solved
in this work. This method is based on an extension of the large neighbourhood search developed by Shawn in [21] and the simulated annealing procedure is enhanced with a system of scoring for the moves.

A real time application of the optimisation process could be envisaged. It means that demands appearing during the service could be treated. One could for example solve the problem in a way similar to the one proposed in Chapter 3 but allowing only for small changes in the routes in order to satisfy the new demands. As a matter of fact, some clients would already be sitting inside the vehicles and the client that would have been waiting for a longer time should not be disadvantaged. The addition of such constraints implies that the solution space could be considerably reduced with respect to the problem studied in this work, making fast real time computation possible.
Bibliography


