FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Solving the Extended Vehicle Scheduling Problem using MetaHeuristics

Jorge Alpedrinha Ramos

Master in Informatics and Computing Engineering

Supervisor: Luís Paulo Reis (Ph. D.)

18^{th} July, 2011

Solving the Extended Vehicle Scheduling Problem using MetaHeuristics

Jorge Alpedrinha Ramos

Master in Informatics and Computing Engineering

Submitted to preliminary evaluation of the committee:

Chair: António Augusto de Sousa (Ph. D.)

External Examioner: Pedro Miguel do Vale Moreira (Ph. D.)

Supervisor : Luís Paulo Gonçalves dos Reis (Ph. D.)

18^{th} July, 2011

Abstract

The Vehicle Scheduling Problem is a well-known combinatorial optimization problem that emerges in mobility and transportation sectors. The heterogeneous fleet with multiple depots extension arises in major urban public transportation companies due to different demands throughout the day and some restrictions in the use of different vehicle types. This extension introduces complexity to the problem and makes the known deterministic methods unable to solve it efficiently. This thesis describes a comprehensive model to interpret the Multiple Depot Vehicle Scheduling Problem as an Asymmetric Traveling Salesman Problem. An approach known for its application to the A-TSP, the Ant Colony based meta-heuristic, was adapted to the problem through this interpretation. The results achieved on solving problems from some of the Portuguese major public transportation companies planning databases show the usefulness of the proposed approach.

Resumo

O problema de afectação de veículos é um problema de optimização combinatória que surge nos sectores de mobilidade e transportes. A extensão com frota heterogénea e múltiplas estações de recolha surge nas maiores empresas de transportes públicos devido a diferentes demandas ao longo do dia e a restrições ao uso de diferentes tipos de viaturas. Esta extensão introduz mais complexidade ao problema tornando as abordagens deterministas conhecidas incapazes de o resolver eficazmente. Esta tese apresenta uma proposta para interpretar o problema de afectação de veículos com frota heterogénea e múltiplas estações de recolha como um problema do carteiro viajante assimétrico (A-TSP). Para resolver o problema através desta interpretação foi adaptada uma meta-heurística baseada no comportamento das colónias de formigas (Ant Colony System) conhecida por obter bons resultados para o A-TSP. Os resultados conseguidos na solução de problemas de algumas bases de dados das maiores companhias de transportes públicos em Portugal mostram a utilidade da abordagem proposta.

Acknowledgements

Before presenting my best thanks to people that enabled this work to be done in the technical field I would like to really appreciate the effort made by the people closest to me that provided the emotional bias required. The work done in this thesis wouldn't be possible without great commitment, and with such commitment it is necessary to relinquish experiences and events that mean so much to me. To all the people that were some how harmed by this priority I would like to thank for the patience and understanding that so dearly they had with me, specially my family and closest friends.

The orientation provided by my supervisor Luis Paulo Reis was, as I've been able to experience throughout this five years of graduation, incalculably valuable. The insight about the scientific community and the share of the vast knowledge allowed to reach a level of confidence that inspires great success and great work.

To all the people that created the amazing environment lived in OPT that served as motivation to keep feeling rewarded by the commitment to the cause. To Ricardo Vieira, Nelson Fernandes, João Castro, Carmo Reis, Rui Pereira, André Meneses, João Marques, José Pacheco, Sara Meireles and Catarina Alexandrino that ensured the funniest coffee breaks that were of great value to the concentration balance required, although the list is large the personality of each influenced differently these moments. I have to give a special thanks message to Renato Cardoso, sharing the path in OPT on developing the Master's thesis was a real pleasure. Finally I would like to thank to Fernando Vieira and Dulce Pedrosa for tirelessly sharing the insight they have about the problem, without it, this work wouldn't ever be successful.

Jorge Alpedrinha Ramos

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations

VSP	Vehicle Scheduling Problem
MDVSP	Multi Depot Vehicle Scheduling Problem
HFMDVSP	Heterogeneous Fleet Multi Depot Vehicle Scheduling Problem
TSP	Traveling Salesman Problem
A-TSP	Asymmetric Traveling Salesman Problem
STCP	Sociedade de Transportes Colectivos do Porto(Public Transport Society of Porto)
UTSP	Unlinked Trips Sub Problem
GIST	Gestão Integrada de Sistemas de Transporte(Integrated Management Transport Systems)
OPT	Optimização e Planeamento de Transportes (Transport Planning and Optimization)

Chapter 1

Introduction

This chapter introduces the motivation and goals of this thesis and presents the context of the problem willing to be solved.

1.1 Motivation and Goals

Public transportation systems play a major role in the lifestyle of most metropolis, and its quality is directly related with a significant improvement in life quality. Big cities have an intense flow of people specially in rush hour and with the common traffic jams an efficient public transportation system is crucial and becomes more important every day because of the continuous migration to metropolitan areas.

Optimal state search characterises the evolution in our world and every development made intends to achieve a better state. Optimisation is present in many day-to-day elements and embraces different knowledge areas. In physics the atoms seek to bound with each other in order to minimise its electrons energy, in biology species evolve to become fitter to survive, man makes its own decisions following a state of happiness and in economy companies seek to increase its income and reduce its costs. [Wei09]

Operational Research and Artificial Intelligence have studied optimisation methods. Operational Research developed deterministic mathematical methods to solve optimisation problems, while Artificial Intelligence is best known for nature inspired meta heuristics that are applied to optimisation problems. Although these are two distinct areas, more capable computer aided solutions have emerged and both fields work starts to converge, specially in the field of planning and scheduling.

The Vehicle Scheduling Problem (VSP) is important to achieve efficiency in a transport network in order to please the changes in demand throughout the day while minimising operational costs. Vehicle Scheduling is the assignment of vehicles to timetabled trips in a feasible way where every trip is satisfied by one vehicle. Efficient solutions for the

VSP intend to minimise the number of required vehicles, the empty trips and vehicle's idle time.

The Heterogeneous Fleet Multi Depot Vehicle Scheduling Problem (HFMDVSP) is an extended problem of the original VSP. This extended problem adds depot constraints that force that a vehicle starts and finishes its duty on the same garage; and it also considers that a given timetabled trip can have restrictions to the vehicles able to perform it. The heterogeneous fleet problem already without the multiple depots is considered NP-Hard [LK81].

This problem arises in public transportation companies, which usually do this kind of scheduling by human hand. When such an exhaustive task is preformed by human hand is not possible to seek the optimal solution, instead, most of the times, any feasible solutions is acceptable and some patterns intended as optimal are followed. This human scheduling leads to suboptimal schedules that won't satisfy the company goal of reducing operational costs and reduces its efficiency, where computer aided systems could provide significant improvement.

The efficiency in solving the Vehicle Scheduling Problem benefits the general population by improving the public transportations service in terms of costs and higher availability obtained through better scheduling of vehicle duties. Several companies face this problems where vehicle type constraints play a major role and the scheduling efficiency depends highly on that subject. Vehicle type restrictions can emerge for several reasons, such as difficulties in a certain vehicle to travel through a given road, different hallmarks being used on vehicles, or even rules that forbid some vehicle type to serve in a given area.

The goal is to find an approach that solves this problem, in its most complex extension, as close as possible to the optimal solution in reasonable time.

1.2 Context

The project will be developed within a company, OPT, that seeks to fulfil the arising needs in public transportation companies related to planning and scheduling. The company works with several public transportation companies in Portugal, each of them with specific needs in this task. The message that defines the company is:

> *"Towards excellent service in innovative and optimized systems for transport planning, management and public information"*

The company's activities are therefore centered on the research, design and development and marketing of software solutions for transport planning and consulting work in these areas. In this subject it has developed two similar, still differentiated by target customer, called GIST and GIST Light. This platforms offer the tools to represent the

transport network and perform the planning tasks. GIST targets at the bigger public transportation companies, and is currently used by the major companies in Portugal, while GIST Light intends to offer most of this features to smaller companies with a lighter platform that is more suitable to their reality.

Nowadays the company solutions provide an algorithm capable of solving efficiently the VSP problem, but it is acknowledged that this is not enough for some of their clients. Each trip has different requirements, some routes oblige a specific vehicle to be used, and in different times of the day some trips require more capacity than others so constraints to vehicle types are demanded as well. It is known that the some of the companies that use the system to solve the vehicle scheduling problem, often have to reschedule it after by hand, because the provided scheme does not consider this constraints.

1.3 Document Structure

Besides this introduction, this thesis contains six more chapters. In the Chapter 2 previous related work is presented and analysed. In Chapter 3 a representation of the HFMDVSP as an Asymmetric Traveling Salesman Problem is presented. A overview of the technological environment where the solution will be integrated is presented in Chapter 4 and in Chapter 5 an approach to solve this problem based on ant colony meta heuristic is proposed. In the late two Chapters, 6 and 7, the results from the developed work are presented and then analysed to reach some conclusions.

Chapter 2

Literature Review

In this chapter the current work on solving Extended Vehicle Scheduling Problems is analysed. The first extended problem analysed is the Multiple Depot Vehicle Scheduling Problem (MDVSP), which has been studied more thoroughly than its extension with Heterogeneous Fleet (HFMDVSP) that usually appears as a special case of the MDVSP problem. The few approaches for the Heterogeneous Fleet problem are analysed as well as approaches for the Asymmetric Traveling Salesman Problem (A-TSP), that has served as a testbed for many heuristics, which presents similar challenges to those from the HFMDVSP. A review over the presented state-of-the-art approaches are then presented.

2.1 Vehicle Scheduling Problem Definition

The Vehicle Scheduling Problem consists of assigning timetabled trips, that must be satisfied by one and only one vehicle, to the respective vehicle duty. To accomplish this it is required to be aware of several restrictions that are implicit, such as the time it takes for a vehicle to travel between locations and the available paths that may allow or not to travel between them.

The Vehicle Scheduling Problem has been widely studied and it has been logically modeled with different structures. Each logical representation of the problem translates into a different Linear Programming model, and different approaches take advantage of each, so slight changes in the problem representation can have significant changes in the approach to solve it.

The Network Flow Network provides a comprehensive definition for the VSP and it is presented as well as its Integer Linear Programming model proposed in [BGAB83].

Each trip $t \in T$ is characterised by its start location $s_{start\ t}$, start time $t_{start\ t}$, end location $s_{end\ t}$ and end time $t_{end\ t}$. The set of arcs AT between trips can be defined by

combining the trips information and the paths available for vehicles. The arcs connect nodes $n \in N$ that can represent the start or end of a trip. To represent the depot, where all bus duties must start and end, the special nodes $n+1'$ and $n+1''$ are used. The node $n+1'$ represents a depot exit and $n+1''$ represents the depot arrival, additionally a special arc is created from the end depot to the start depot in order to add the costs related to the use of a vehicle.

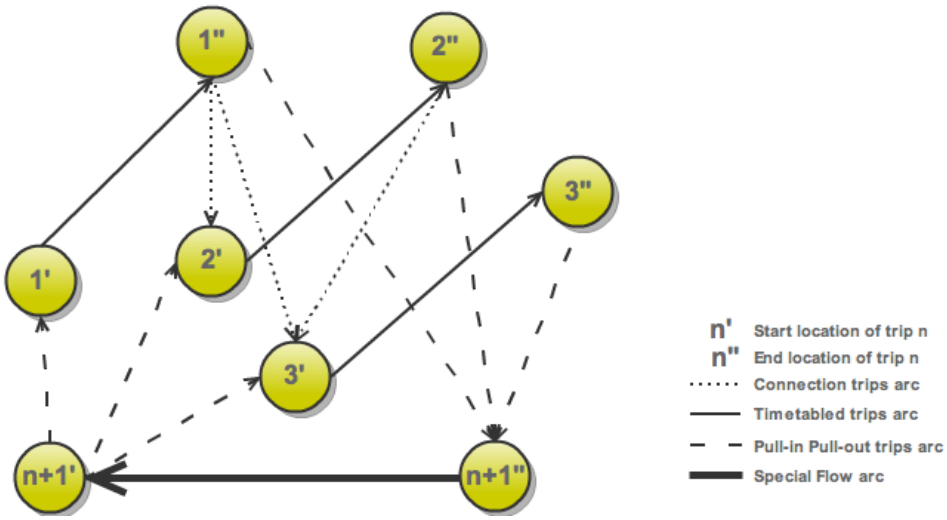

Figure 2.1: Network Flow Model for three trips

To define which nodes are connected by the arcs in $a_{ij} \in A$ a compatibility operator must be established and it represents whether or not after trip i the bus can also serve j if the time to travel between the locations isn't longer than the time between the end of i and the start of j. Trips are compatible if one of two premises is true as presented in Eq. 2.1. The links that connect the start and end of a trip are represented by $AT \subset A$.

$$compatible_{ij} = \begin{cases} t_{end\ i} < t_{start\ j} & s_{end\ i} = s_{start\ j} \\ t_{end\ i} + d_{ij} < t_{start\ j} & s_{end\ i} \neq s_{start\ j} \end{cases} \tag{2.1}$$

The problem is presented as a integer linear programming problem in Eq. 2.2 where c_{ij} are the costs of connecting between i and j and x_{ij} is a boolean variable that defines whether or not that connection belongs to the model.

$$\begin{array}{lll} \min & \sum_{(i,j)\in A} c_{ij}x_{ij} & \\ \text{s.t.} & \sum_{i:(i,j)\in A} x_{ij} - \sum_{i:(j,i)\in A} x_{ij} = 0 & \forall n \in N \\ & 1 \leq x_{ij} \leq 1 & \forall (i,j) \in AT \\ & x_{ij} \geq 0 & x_{ij} \in \mathbb{N} \end{array} \tag{2.2}$$

2.2 Multi-Depot Vehicle Scheduling Problem

The Multi-Depot Vehicle Scheduling Problem is an extension of the VSP in which vehicles are distributed in several depots. This problem has two strands one forces that the vehicles start and end in the same vehicle and the other doesn't, this causes significant changes while solving the problem, even though in terms of logical representation they remain very similar.

Throughout the extensive search made in the past decades on this topic, different model approaches were proposed. In [BK09] Bunte and Kliewer make an overview on this models that is presented next.

2.2.1 Modelling The Problem

The models are divided in three groups:

- Single-Commodity Models
- Multi-Commodity Models
- Set Partitioning Models

2.2.1.1 Single-Commodity Models

Single Commodity Models are graphs where a single node represents a trip or a depot. Two Single-Commodity models have been presented.

The **Singe-Commodity with Sub-tour Breaking Constraints** presented in [CDFT89] uses a node to represent each vehicle per depot and arcs from these nodes to every trip, modelled as a transportation problem with additional sub-tour breaking constraints which forbid every path with more than one vehicle, as seen in Fig. 2.2.

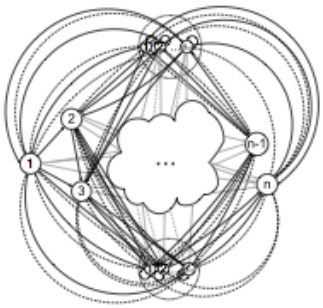

Figure 2.2: Single-Commodity Model with Sub-tour Breaking Constraints

In [MP92] Mesquita and Paixão propose the **Singe-Commodity Model with Assignment Variables** that provides a more comprehensive network structure that represents each depot as a node as well as arcs connecting them to each trip, and an additional group

of variables that assign each trip to a specific depot. This model reduces significantly the number of constraints as well as the number of variables in comparison to the other Single-Commodity Model, represented in Fig. 2.5.

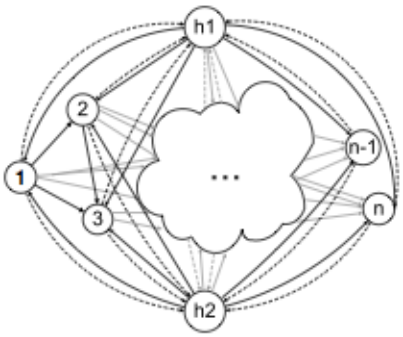

Figure 2.3: Single-Commodity Model with Assignment Variables

2.2.1.2 Multi-Commodity Models

Multi-Commodity formulations generate an independent subnetwork for each depot. The group of subnetworks model the problem as a whole. Each subnetwork is an extension of the Network Flow Model presented in Fig. 2.1 even though a few changes have to be made. The multiple networks concept is common in the further presented models, the Connection-Based Network and the Time-Space Network, but they present different subnetwork structure.

Connection-Based Network

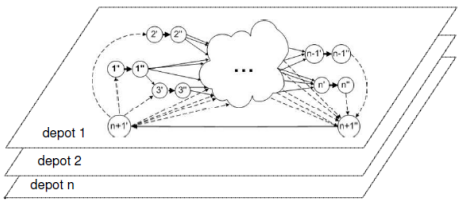

Figure 2.4: Connection Based Network

The subnetworks are modelled as a Network Flow Model, and constraints must be added in order to guarantee that each trip is served only in one of the subnetworks (2.3), and a constraint to ensure that no more than the available vehicles from a depot are used (2.4). Consider T the timetabled trips and $AT^t \subseteq AT \subset A$ where $t \in T$ as the set of arcs related to trip t in all subnetworks; H is the set of all depots available and d^h represents

the number of vehicles in depot $h \in H$ and A^h is the set of circulation flow arcs of depot h.

$$\sum_{(i,j)\in AT^t} x_{ij} = 1 \quad \forall t \in T \tag{2.3}$$

$$\sum_{(i,j)\in A^h} x_{ij} \leq d^h \quad \forall h \in H \tag{2.4}$$

Time-Space Network

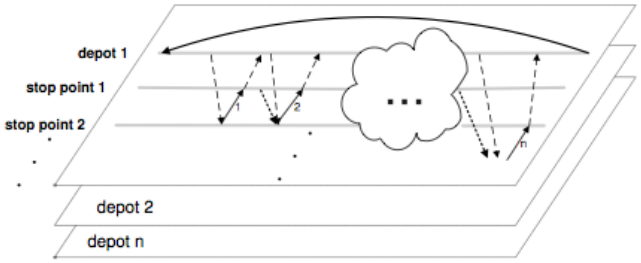

Figure 2.5: Time-Space Network

Time-Space Network, presented by Kliewer, Mellouli and Suhl ([KMS02] and [KMS06]), uses a different subnetwork structure. The main idea is to group connections between one trip and compatible trips based on Eq. 2.5. This is possible through a time oriented network where each station is represented by a time line with several arcs to and from it, representing the timetabled trips or connection trips.

$$(i \propto j) \wedge (j \propto k) \rightarrow (i \propto k) \tag{2.5}$$

2.2.1.3 Set Partitioning Models

In [HMS06] the Set Partitioning Model is presented as a Dantzig-Wolfe decomposition to the multi-commodity model. A set of every possible path in the multi graph is represented by Ω , which allows to model the problem without many constraints (Eq. 2.6) even though it increases the number of variables. In this model a_{jp} represents the dual variable related to the use of trip j in the path p, while k_d represents the number of available vehicles in depot d.

$$\begin{array}{lll} \min & \sum_{d\in D}\sum_{p\in\Omega_d} c_p x_p & \\ \text{s.t.} & \sum_{d\in D}\sum_{p\in\Omega_d} a_{jp} x_p = 1 & \forall j \in T \\ & \sum_{p\in\Omega_d} x_p \leq k_d & \forall d \in D \\ & x_p \in \{0,1\} & p \in \Omega \end{array} \tag{2.6}$$

2.2.2 Approaches

Several heuristic approaches have been studied since the 1970s to solve MDVSP instances. These approaches can be divided in search heuristic solution approaches ([BCG87], [BGAB83], [BRK78], [DFT93]) and deterministic heuristic approaches ([CDFT89], [HMS06], [KMS06], [LÖ97], [LÖ98] , [RS94] and [MP92]).

An important comparison between heuristic and meta heuristic based methods is made by Pepin et al. in [PDHH06] where both methods advantages and disadvantages are presented . Five methods are presented and analysed :

- Heuristic MIP Approach
- Lagrangian heuristic
- Heuristic column Generation
- Large Neighbourhood Search
- Tabu Search

Heuristic MIP Approach

The heuristic MIP approach here presented relies on a Time-Space Network Model, Fig. 2.2.1.2, using a branch-and-cut method, provided by the CPLEX MIP solver. The algorithm stops as soon as it finds an integer feasible solution.

Lagrangian heuristic

The Lagrangian heuristic proposed, solves the simplified VSP associated problem, by relaxing flow constraints on the connection based model and omitting availability constraints adding Lagrangian multipliers associated to each trip node. This provides feasible paths for a solution lacking the association to a depot, which can be solved as a transportation problem with an Hungarian algorithm. The solution for the (VSP) relaxed subproblem provides a lower bound while the solution provided through the transportation problem provides an upper bound. After this the Lagrangian multipliers are updated as well as parameters used to ensure the algorithm convergence.

Heuristic column Generation

Uses the Set Partitioning Model which implies a high number of variables, given the amount of possible paths. To become feasible it is divided in a master problem, considering a smaller set of possible paths, and a subproblem for each depot. Iteratively the master problem is solved and from its solution dual variables for the reduced cost subproblems are calculated which are then solved in order to find better paths for the master problem. The heuristic solves this problem iteratively until optimality is achieved.

Large Neighbourhood Search

Large Neighbourhood Search uses a column generation heuristic for improving a set of scheduled trips r in a given initial solution. The algorithm is an informed search heuristic that at each step selects a few scheduled trips (r) and reassigns them in an optimal way by solving the reduced cost problem associated.

Three different heuristics are used to choose which scheduled trips to optimise that allow to explore thoroughly the initial solution. The heuristics are called *Random Schedules*, *Less Frequent Schedules* and *Closest Schedules*. The first acts as the name suggests, by selecting trips randomly, the second selects the trips that have not been selected in the last iterations of the algorithm ensuring that the whole schedule is reconsidered, as of the last attempts to select one trip and after that select the closest trips that could possibly be connected to it.

Tabu Search

Tabu Search takes advantage of the neighbourhood definition of a given state. The neighbourhood of a given solution are the states that can be achieved by using a search operator. In this approach two search operators were defined, move and swap. Move switches the vehicle that will perform one trip, while swap makes a switch between two trips i and i' assigned to two bus duties j and j' respectively the operation consists of assigning i to j' and i' to j. A tabu is something that is avoided, so in tabu search some operations shall be avoided, this tabu list is created as search operations are preformed. With this tabu list one avoids to make cyclic changes that will not produce an improvement, but after several iterations, that operation may gain new meaning and significance, so it will be removed from the tabu list, allowing exploitation operators to be preferred, but not losing the capacity to reconsider operations.

Lagrangian relaxations and Column Generation

Löbel presents an approach in [L99̈] capable of solving up to 25000 trips. The method initially defines a lower bound using Lagrangian relaxations and an upper bound using an opening heuristic. The LP relaxation is solved to optimality using column generation, using Lagrangean pricing and standard reduced cost, and column elimination scheme. If the new upper bound found is close enough to the lower bound, the algorithm stops. This combination of heuristics is the result of a wide study made by Löbel and presents very good results.

2.2.3 Results

The comparison provided in [PDHH06] indicates that Heuristic Column Generation explained before provides the best solutions when there is enough computational time and stability is required, while Large Neighbourhood Search presents the best trade-off between time required and solution quality. While the other heuristics have a worse behaviour. The Heuristic MIP Approach provides good quality solutions, but requires too much time to find a solution; Tabu Search is able to improve the initial solution very fast in the beginning but reaches a limit from where it takes too much time to improve and the Lagrangian Heuristic performs closely to Large Neighbourhood Search with a slight decrease in solutions quality.

2.3 Heterogeneous Fleet Multi-Depot Vehicle Scheduling Problem

In comparison to the MDVSP, the Heterogeneous Fleet extension introduces restrictions to the vehicle types that can be used to satisfy a given trip.

The must used solution found to this restriction is to model depots as pairs of physical depots / vehicle types. The subnetwork related to a given depot only considers the trips that can be satisfied by that kind of vehicle.

2.3.1 Approaches

In [KMS06] Kliewer et al. propose a fix-and-optimize heuristic for solving MDVSP with vehicle-type restrictions. This heuristic relaxes the main problem into sub problems for each physical depot. Optimal solution for each sub problem is found and the paths that appear more often in sub problem solutions are considered to be optimal and are fixed, reducing the problem into solving the remaining trips.

In Löbel thesis work [L̈97] he presents a solution for solving different extended VSP instances, and first approaches the HFMDVSP using a Connection-Based Network Model with a depot for each vehicle type in each garage. The approach used in this thesis is the early work of Löbel into defining the previously presented approach.

The work developed in OPT by S. Neves in [NG09] presents two heuristic approaches, Firsts and Clusters. Firsts separates the remaining trips to assign into two groups at each step, one with every trip that can't be preceded by any other and the other group with the remaining. At each iteration the connection between the first and second group problem is solved, and the process is repeated for the second group, until all trips are assigned to a duty. Clusters solves a minimum weight matching problem transforming a pair of nodes into a new node (cluster) iteratively until no matching is possible and all trips are clustered. Each cluster represents a bus duty that will be serviced by the cheapest vehicle that can satisfy every trip in that cluster. The two heuristics assign depot and vehicle

type to each duty after the trip sequences are defined which doesn't really consider the optimisation problem in relation to vehicle types.

2.3.2 Results

The fix-and-optimize solution proposed by Kliewer et al. provided a solution for the practical problems it intended to solve, even though it performed better than heuristic MIP approaches, the time to achieve solution was high, but it was only tested to solve problems with more than 2000 trips.

The heuristic tested by Löbel presents results similar to the ones of Kliewer, but the cases considered are slightly different. This heuristic has been able to solve problems with over 3000 trips, but the time spent is relatively high. A good term of comparison between both methods computational time is impossible to be established.

The algorithms Firsts and Clusters were tested against a lower bound defined by the optimal solution for the similar VSP calculated through the heuristic proposed in [PB87]. The results reveal that even for smaller cases considered (maximum 400 trips) it provided a solution containing almost the double of the vehicles needed in the VSP solution, which can not be considered a good result.

2.4 Asymmetric Traveling Salesman Problem Approaches

The Asymmetric Traveling Salesman Problem is a NP-Hard combinatorial problem that has been studied for several years as a special case of the original Traveling Salesman Problem. The main difference between the original Traveling Salesman Problem and it's asymmetric version is that the distance between two cities is different in different directions (i.e. A to B 10Km and B to A 12Km).

This problem approaches can provide a good starting point to apply to the problem being solved, the Extended Vehicle Scheduling Problem. Some methodologies are presented and analysed in terms of quality and possibility to be adapted to this matter.

2.4.1 Approaches

2.4.1.1 Genetic Algorithm

In the First International Contest on Evolutionary Optimisation [BDL+97] several methods were presented in a contest for solving several instances of the A-TSP, this instances are available in [Rei08].

The winner of this contest was an approach based on a genetic algorithm proposed by Freisleben and Merz [FM96] that starts its search from a population of individuals generated using a nearest neighbour heuristic. The individuals represent feasible solutions,

and at each step two of them are selected for crossover. The crossover procedure starts by deleting all edges from the first parent that are not contained in the second and then reconnects the segments. The reconnection is performed using a greedy heuristic based on a nearest neighbour choice. The new individuals are brought to the local optimum through a 3-opt heuristic, and a new population is generated after the application of a mutation operation that randomly removes and reconnects some edges in the tour.

2.4.1.2 Ant Colony System

The Ant Colony System was initially proposed by Dorigo [Dor92] and later proposed by him and Gambardella [DL97] to solve the A-TSP. It is a nature inspired meta heuristic that emulates the behaviour of ants in the pursuit for the shortest path to reach food. It can be interpreted as a GRASP heuristic that at each iteration creates a given number of paths, by means of artificial ants that follow preferentially connections with high concentrations of pheromones. The pheromone concentration on a given connection is updated with consideration for the final evaluation of paths containing it, this is done in two phases local update that favours exploration and global update that favours exploitation. This method is capable of identifying the key connections to find a good solution.

2.4.2 Results

In the paper that presented the Ant Colony System as a solution for the Asymmetric Traveling Salesman Problem, a comparison between the two presented methods using the data instances from the International Contest on Evolutionary Optimisation is presented. The results, which are presented in Tab. 2.1 suggest that both methods could find solutions very close to the known optimum, although the Ant Colony System did better in some cases.

	Optimum	**ACS avg**	**ACS Error%**	**GA avg**	**GA Error %**
p43 43 cities	2810	2810	0,00%	2810	0,00%
ry48p 48 cities	14422	14422	0,00%	14440	0,12%
ft70 70 cities	38673	38679,8	0,02%	38683,8	0,03%
kro124p 100 cities	36230	36230	0,00%	36253,3	0,01%
ftv170 170 cities	2755	2755	0,00%	2766,1	0,40%

Table 2.1: Comparison between the Ant Colony System and ICEO winner Genetic Algorithm

2.5 Conclusions

Real world instances of the Multi-Depot Vehicle Scheduling Problem with up to 7000 tasks have been solved to an optimal state, and some solutions for bigger problems with

25000 tasks have been solved with acceptable solutions for this environment [L̈99]. Even though this such problem instances have only been solved because of their special structure that eases the solution, while there are some less structured cases that are harder to solve. Carpaneto et al. [CDFT89] proposed some randomly generated instances of the problem, that have been used by many researchers, and instances from this set with only up to 800 tasks have been solved to optimality [HMS06].

The solution found in [NG09] doesn't consider the costs associated to using a specific vehicle in the process of creating bus duties, which can lead to a situation where a more expensive vehicles are used in order to satisfy the least expensive path set. This conclusion represents that the distance between HFMDVSP and VSP is not as linear as solving multiple VSP problems, but creates a more complex network.

Combining mathematical programming with meta heuristic has proven to be a valuable approach in order to achieve a trade-off between solution quality and computational time [PDHH06]. This trade-off can be observed as well in approaches for the A-TSP problem, for which approaches such as Ant Colony System or Genetic Algorithms present good solutions in reduced time.

The approaches used for the A-TSP represent a good starting point for new applications to the HFMDVSP since this can be interpreted as a special A-TSP, this analogy is better presented in Section 3.4.

Chapter 3

Problem Representation

Before introducing approaches to solve the problem it is important to thoroughly introduce all the concepts of the problem, in a manner that allows to understand fully how the information is related and its role in solving the problem. The basic input data as well as the optimization goals and the restrictions of problem are presented and finally the output format is shown allowing to understand the process between identifying the problem and presenting a solution to it. A Linear Programming Model is also presented to formally define the problem.

The comparison between the Vehicle Scheduling Problem and the Asymmetric Traveling Salesman problem was proposed in [RRP]. This comparison is hereafter explained further to introduce how approaches known for its application to the A-TSP can be applied to Vehicle Schedule Problem extension.

The decision support system supports more functionalities than full instance scheduling, enabling the users to explore the algorithm with partial solutions defined.

3.1 Problem Definition

The Heterogeneous Fleet Multi Depot Vehicle Scheduling Problem is defined by the timetabled trips, the empty paths available, their duration , the stations, depots, its restrictions, and the objective function. This concepts will be presented thoroughly to provide a better understanding.

3.1.1 Timetabled Trips

The timetabled trips T (i.e. Tab. 3.1) must be covered exactly once, each trip t_i is defined by the start and end station $s_i, e_i \in St$, and start st_i and end time et_i.

$i \in T$	st_i	$s_i \in S$	et_i	$e_i \in S$
Trip 1	08:10	A	08:20	B
Trip 2	08:25	B	08:35	C
Trip 3	08:45	C	08:55	A

Table 3.1: Scheduled Trips

3.1.2 Empty Paths

The empty paths P (i.e. Tab. 3.2) define which stations $a, b \in St$ may be connected by an empty trip $et \in P$ and the time that it takes $t_{a,b}$. The stations St are the locations where trips can start and end, and some of them are depots $D \subset St$ which are special stations where vehicles can stay for longer periods such as stay overnight.

$ep \in P$	$a \in St$	$b \in St$	$t_{a,b}$
A to Dep	A	Depot	5
Dep to A	Depot	A	5
Dep to C	Depot	C	5

Table 3.2: Empty Paths

3.1.3 Problem Restrictions

The objective in solving this problem is to find an assignment such that:

- every trip is satisfied by exactly one vehicle
- every vehicle duty starts and ends in a depot
- each vehicle performs a feasible sequence of trips
- the overall costs are minimised

The first topic is clear, no predetermined trip can be left out of the assignment, so it must have a vehicle to perform it.

The second topic implies that every vehicle has a depot where it starts the day and must end in the same depot, this must be the station where the first trip starts and the last trip ends for the respective duty.

The third topic introduces the term feasible into the assignment. A feasible assignment implies feasible sequences of trips for every vehicle, and a feasible sequence of trips is nothing more than one that a vehicle can actually perform. To clearly define what sequences a vehicle can perform the **trip compatibility** (Section 3.1.3.2) must be defined, so that a feasible trip sequence is one where every pair of consequent trips are compatible.

The fourth topic is related with the costs of a given assignment that are intended to minimise. The cost structure is composed by tangible and intangible costs that are presented further (Section 3.1.4).

3.1.3.1 Vehicle Type Restrictions

Each trip $i \in T$ can have denied vehicle types $De_i \subset VT$ and/or mandatory vehicle types $Man_i \subset VT \setminus De_i$, meaning that the feasible vehicle types for the trip are defined by $VT_i = (VT \setminus De_i) \cap Man_i$. An example of this restrictions is presented in Tab. 3.3.

$i \in T$	De_i	Man_i	VT_i
Trip 1	Mini-Bus,Articulated	$\emptyset$	Bus
Trip 2	Mini-Bus	Bus	Bus
Trip 3	$\emptyset$	Bus	Bus

Table 3.3: Vehicle Type Restrictions Definition Sample

3.1.3.2 Trip Compatibility

Trip compatibility between two consequent trips means that they can be satisfied by the same vehicle in sequence. The compatibility $\alpha_{i,j}$ between two trips $i, j \in T$ can be formally defined as in Eq. 3.1.

$$\alpha_{i,j} = \begin{cases} et_i <= st_j & if\ e_i = s_j\ \&\ VT_i \cap VT_j != \emptyset \\ et_i + dist_{i,j} < st_j & if\ e_i \neq s_j\ \&\ VT_i \cap VT_j != \emptyset \end{cases} \tag{3.1}$$

3.1.3.3 Time between trips

The time required $dist_{i,j}$ to connect two consequent trips $i, j, et_i < st_i$ may not be equal to the time between the two stations e_i and s_j, because there are restrictions for the time a vehicle can remain idled at a given station. A given station has a maximum stop time, that determines how long can a vehicle stay idled in it.

In cases which time between the trips is too long, it is possible to make an interruption to the vehicle duty in a given depot if its maximum stop time allows it and then return to work. This is called a rest link and it also has a minimum stop time so that it can be considered.

$$dist_{i,j} = \begin{cases} t_{i,j} & if\ st_j - et_i - t_{i,j} < maxStop_j + maxStop_i \\ dist_{i,d} + dist_{d,j} & if\ minS_d < st_j - et_i - dist_{i,d} + dist_{d,j} < maxS_d\ \&\ d \in D \\ \infty & otherwise \end{cases} \tag{3.2}$$

3.1.3.4 Depot Restriction

Since every vehicle duty must start and end in the same depot, additional constraints are required in terms of trip compatibility.

Two consequent trips can be compatible, but they may not be reachable and reach to the same depots. To understand this further assume that there are only two trips to assign which are consequent, the first trip goes from A to B and the second trip from B to C and all stations are depots. According to the definition of compatibility this two trips would be compatible, the catch is that there is no empty path between C and A, and this would imply that if this two trips are assigned to the same vehicle it will start in A and end in C, which does not meet the restriction that every vehicle must start and end in the same vehicle. Although if there was a third trip that went from C to A after the second trip, it would become feasible.

This example explains the complexity introduced by multiple depots which demands that trip compatibility takes this in consideration. So a trip is available for a given depot, only if it has a connection from that depot or a compatible precedent trip does and if it has a connection to the depot or a compatible consequent trip that does. This brings even another problem, if trip i has a compatible precedent trip j that has a connection from depot d and a compatible consequent trip k that has a connection to the depot d, but all other precedent and consequent trips are not available for that depot, it cannot have the trip i as precedent trip assigned if the consequent trip is not k.

This implies that trip compatibility is different for each depot, which makes a switch to the α operator defined in Eq. 3.2, becoming α^d where $d \in D$. From this definition it is concluded that to create an assignment it must be defined previously for each vehicle what is the depot in which it will start and end so that trip compatibility can be determined.

3.1.4 Costs Structure

Solving the Vehicle Scheduling Problem implies not only finding one feasible solution for the problem but finding the best solution. The quality of a solution is measured by the costs that it has for the company applying it. The objective function is to minimize the sum of the costs of connections belonging to the solution.

The major factor in the costs structure must be the number of vehicles necessary to fulfil it specially because of this assets intrinsic value and the additional crew necessities. Although this is a major factor there are others that must considered as well, such as the time the vehicles spend idled and performing additional trips without passengers.

3.1.4.1 Vehicle Duty Costs

The most important factor of an assignment is the number of vehicle duties required to perform the trips, so it is important to define how important it is in terms of global optimality. To achieve this a parameter that represents the costs of using an extra vehicle is defined and is let to the final user to determine how important it is for the solution quality. This parameter is then multiplied by the previously defined Vehicle Type Factor to determine the cost of using a specific vehicle type for a new vehicle duty, this favours the use of the least expensive vehicle when it is possible.

3.1.4.2 Connections Costs

The compatibility operator between two trips $\alpha^d_{i,j} = 1$ has a related cost $c^d_{i,j}$, that represents how much assigning i and j consequently costs to the final evaluation of the solution. Different factors change the value of this cost, this factors are:

- **Empty Trip Penalty** The empty trip penalty *ePen* is a factor that is multiplied by the time that the vehicle spends performing trips without passengers.
- **Idled Time Penalty** The idled time penalty *idPen* is a factor that penalises the time the vehicle has to remain still in a given terminus.
- **Idled Time at Depot Penalty** $idDepPen$ The same penalisation as the previous one, but applied to depot stations only.
- **Vehicle Type Factor** $VTFac_{vt}$ Is a factor that translates the difference between the operational costs of different vehicles.

A connection between two trips is defined by the empty trips required to perform the connection and the time it has to remain idled in any given station during the connection period. Determining the costs of a given connection is only possible after calculating three variables: the idled time in a depot station $iDepTime^d_{i,j}$, the time idled in a non-depot station $iTime^d_{i,j}$ and the time on empty trips $eTime^d_{i,j}$, after doing so the $c^d_{i,j}$ can be obtained by Equation 3.3.

$$c^d_{i,j} = ePen * eTime^d_{i,j} * VTFac_{vt} + idPen * iTime^d_{i,j} + idDepPen * iDepTime^d_{i,j} \quad (3.3)$$

3.2 Solution Representation

An assignment for the HFMDVSP is a set of vehicle duties each representing a different vehicle and the trips it will preform. This trips include the timetabled trips and auxiliary empty trips to connect between stations so that the vehicle can satisfy the next trip, making

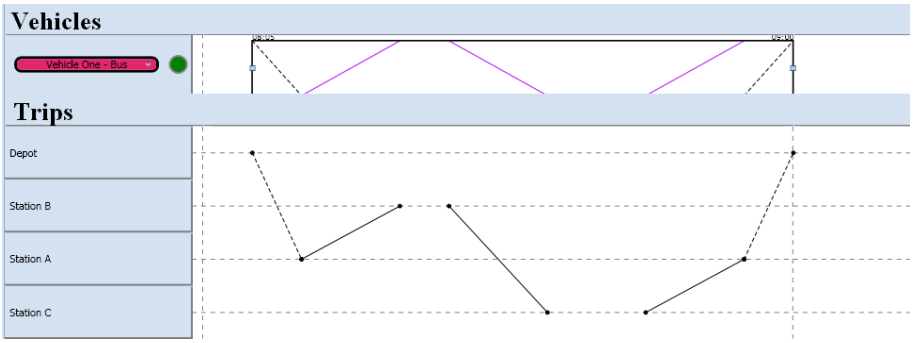

Figure 3.1: A graphic representation of a sample problem

the vehicle duty feasible. Each vehicle duty has its respective vehicle type and starts and ends in a depot station.

In this graphic representation the three predetermined timetabled trips presented in Table **??** are represented by the continuous lines, and the dashed lines represent empty trips that are created to allow the vehicle duty to be feasible. In this case the empty trips link the Depot to the first trip, and the last trip back to the Depot.

3.3 Linear Programming Model

The problem representation was based on a Connection-Based Model (Section 2.2.1.2) using pairs depot/vehicle type to represent each layer. To build each layer's network the previously presented restrictions and compatibility operators are used and each node represents either the start or end of a trip, or the exit or arrival of a given depot. This model can be represented by the mathematical method in Eq. 3.4. In this model N is the set of all nodes and c_{ij} represents the costs of an arc between node i, j that belong to the set of arcs A where all possible connections between nodes are present and x_{ij} is a boolean variable that defines whether or not a given arc belongs to the solution. The subsets $AT \subset A$ contains all arcs that connect with a node of any trip, while $AT^t \subseteq AT$ represents the connections that are related to trip $t \in T$. In the same logic $A^h \subset A$ represents all the connections related to a depot node $h \in H$.

$$
\begin{array}{lll}
\min & \sum_{(i,j)\in A} \; c_{ij}x_{ij} & \\
\text{s.t.} & \sum_{i:(i,j)\in A} \; x_{ij} \; - \; \sum_{i:(i,j)\in A} \; x_{ij} = 0 & \forall j \in N \\
& \sum_{(i,j)\in AT^t} \; x_{ij} = 1 & \forall t \in T \\
& \sum_{(i,j)\in A^h} \; x_{ij} \leq d_h & \forall h \in H \\
& x_{ij} \in \{0,1\} & \forall (i,j) \in A \setminus \bigcup_{h\in H} A^h \\
& x_{ij} \geq 0 \; \textit{and integer} & \forall (i,j) \in \bigcup_{h\in H} A^h
\end{array}
\tag{3.4}
$$

The first equation constraints the problem so that the number of incoming arcs on a node is the same as the outgoing arcs, the second ensures that in the nodes related to trips the number of incoming and outgoing arcs is equal to one, while the third guarantees the vehicle availability where the number of outgoing arcs from a given depot is less than or equal to the number of vehicles available. The last two equations guarantee that the arcs related to any trip node are used only once, if used, while arcs from depot nodes can be used one or more times.

3.4 Traveling Salesman Problem Comparison

The Traveling Salesman Problem is a widely studied subject, it is a well-known combinatorial optimisation problem. It is the problem of assigning a route of a traveling salesman that must go through a number of cities with different distances between each other. An instance of the problem with n cities is defined by a distance matrix $M[n][n]$, as the one defined in Tab. 3.4 and the solution consists of an optimal Hamiltonian tour with the shortest possible length.

	A	**B**
A	∞	300
B	300	∞

Table 3.4: Traveling Salesman Problem Distance Matrix

The symmetry of a Traveling Salesman Problem exists if the distance from and to a city C is the same in every case, and otherwise it is considered an Asymmetric Traveling Salesman Problem.

3.4.1 A-TSP Approach

With the Connection Based Network model, the problem of finding the optimal solution for the assignment can be interpreted as a special case of the Asymmetric Traveling Salesman Problem. The main difference is that in the original A-TSP the traveling salesman only visits each city once, returning to the original city and in this problem the depot is a special node that can be visited more than once, and the links in the model have an infinite cost in the opposite direction.

The problem defined by the empty trips in Tab. 3.1 and the empty paths in Tab. 3.2 can be represented as an A-TSP graph in Fig. 5.2 or as a cost matrix as seen in Tab. 3.5. The links between nodes have a special structure due to the time restrictions that does not allow all nodes to be connected between each other. The connections that are not feasible appear represented with the ∞ symbol in the cost matrix.

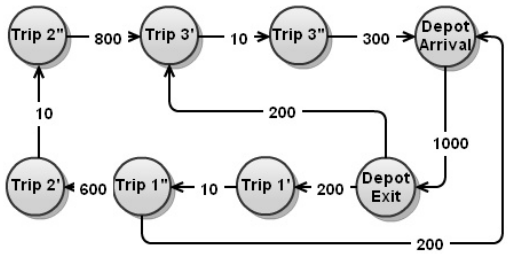

Figure 3.2: An A-TSP Graph sample

From/To	Depot E.	Depot A.	Trip 1'	Trip 1"	Trip 2'	Trip 2"	Trip 3'	Trip 3"
Depot E.	∞	∞	200	∞	∞	∞	200	∞
Depot A.	1000	∞	∞	∞	∞	∞	∞	∞
Trip 1'	∞	∞	∞	10	∞	∞	∞	∞
Trip 1"	∞	200	∞	∞	600	∞	∞	∞
Trip 2'	∞	∞	∞	∞	∞	10	∞	∞
Trip 2"	∞	∞	∞	∞	∞	∞	800	∞
Trip 3'	∞	∞	∞	∞	∞	∞	∞	10
Trip 3"	∞	300	∞	∞	∞	∞	∞	∞

Table 3.5: One layer cost matrix

Another peculiarity about this case is that the cost of the connections between trips is different whether the vehicle type being used, and given the depot restrictions (defined in Section 3.1.3.4) the connection may not exist for a given depot. The Connection Based Network Model uses a multi layer network to represent the different depots connections that in the case of heterogeneous fleet can be used to represent pairs depot/vehicle type. By doing so every layer can be represented as a costs matrix.

Multiple depots and heterogeneous fleet extension was not considered in the cost matrix of Tab. 3.5, but it changes this scheme into a more complex cost matrix as in Tab. 3.6. In the cost matrix each connection has a cost for each pair depot/vehicle type that is represented in a small matrix as well, this matrix relation is presented in Tab. 3.7. Connections between the start and end of trips are not represented in this matrix for ease of comprehension. Interpreting this model as an A-TSP there are two major draw backs that must be overcome : the multiple visits to depots, and the multi layered costs matrix.

This special version of the A-TSP brings more complexity to the problem because it is not only about improving the path distance, but also to reduce the depot visits, which can be overcome by using a special connection from the depot arrival to the depot departure, which will be the only connection available from the depot arrival. With this tweak a depot visit will represent a significant increase in the path cost, which still allows the A-TSP approaches to work in this case.

From/To	Depot A.		Trip 1'		Trip 2'		Trip 3'	
Depot E.	∞	∞	200	∞	∞	∞	200	∞
	1000	∞	∞	∞	∞	∞	∞	∞
Trip 1"	∞	∞	∞	10	∞	∞	∞	∞
	∞	200	∞	∞	600	∞	∞	∞
Trip 2"	∞	∞	∞	∞	∞	10	∞	∞
	∞	∞	∞	∞	∞	∞	800	∞
Trip 3"	∞	∞	∞	∞	∞	∞	∞	10
	∞	300	∞	∞	∞	∞	∞	∞

Table 3.6: Multi layer cost matrix

Depot A Vehicle 1	**Depot B Vehicle 1**
Depot A Vehicle 2	**Depot B Vehicle 2**

Table 3.7: Depot / Vehicle Type Pairs

3.5 Partial Solutions

Public Transportation companies are constantly subject to changing rules in terms of operational limitations, and so it is important that they can interact with the decision support system as much as possible, not only establishing the parameters, but also by being able to use it partially. This can be done in three different ways meeting different requirements which will be presented further.

3.5.1 Completing Partial Solutions

Completing partial solutions allows the user to define a small sequence of trips that shall be performed in that given way, and still use the system to assign the remaining trips, making the partial solutions become feasible ones, as can be observed in Fig. 3.3. This may be necessary when there is a sporadic restriction for a given driver, or a specific path that may not be available for some time, and this can be overcome without the necessity of creating new schedules to overcome that constraint.

3.5.2 Merging Partial Solutions

Defining partial solutions may imply creating more than a partial vehicle duty and in some cases it may desirable to merge some of them if possible, and in other cases it may not be so desirable.

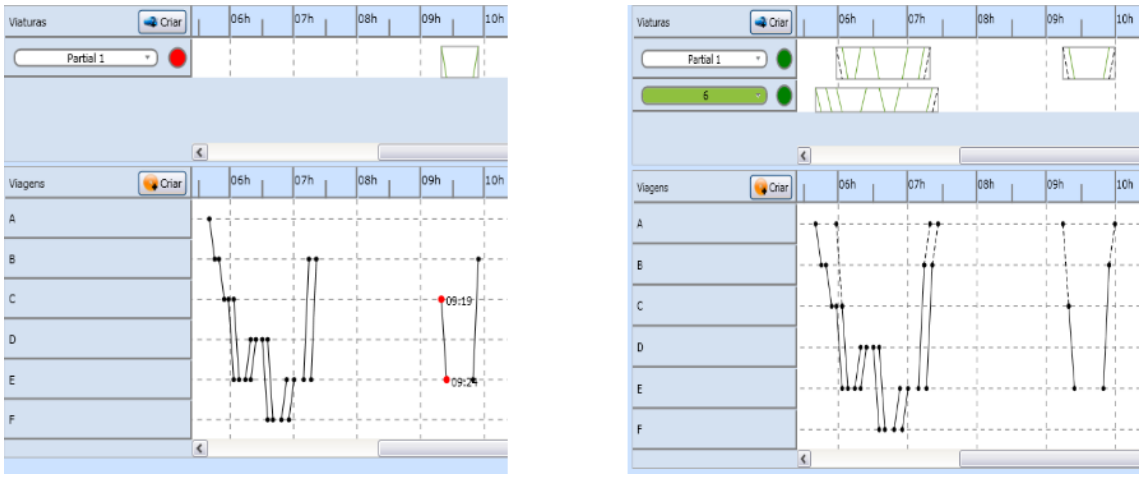

Figure 3.3: Partial Solutions: on the left a partial solution created through the GIST interface and on the right the solution provided by the algorithm using the partial solution

This allows the user to create restrictions of small blocks of trips that must be performed in a given way, not considering whether or not some of them will connect some how by merging vehicle duties and completing them, as seen on Fig. 3.4.

When it is not desirable to merge partial vehicle duties, the user may define that the partial solutions may not be connected, and can create disjunction restrictions, such as ensuring that two trips (or trip blocks) are not performed by the same vehicle.

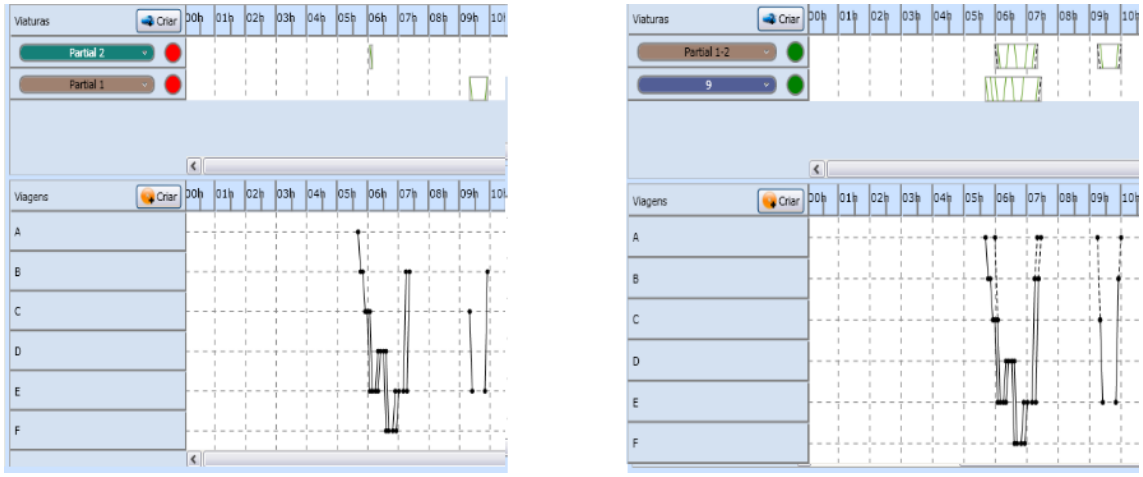

Figure 3.4: Merging Partial Solutions: on the left two partial solutions created through the GIST interface and on the right the solution provided by the algorithm using the merged partial solutions

3.5.3 Locked Partial Solutions

Another requirement may be to let a given vehicle duty as it is, and not changing it, which introduces the concept of locked partial solutions. The algorithm can still perform an optimal solution for the remaining trips but not considering the already defined duties.

This use can be observed in Fig. 3.5 where the duties *Locked 1* and *1* could be a single duty, but it was defined by the user that *Locked 1* was locked.

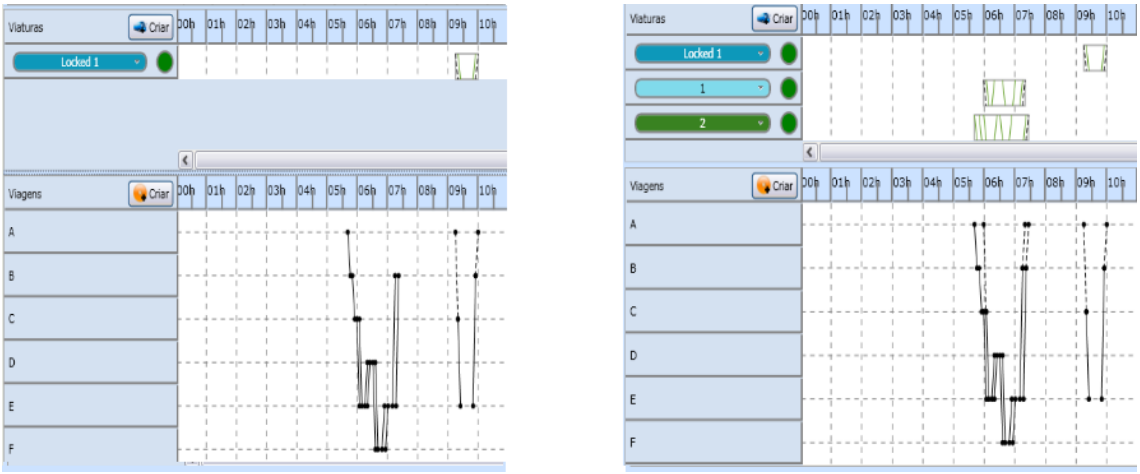

Figure 3.5: Locked Partial Solutions: on the left a partial solution created through the GIST interface and on the right the solution provided by the algorithm without changing the locked partial

3.6 Conlcusions

Modeling the Extended Vehicle Scheduling Problem plays a significant role in efficiently solving it. Selecting the Connection Based Model with multiple layers provides a good way to handle the multiple depot and heterogeneous fleet extension.

The comparison with the Asymmetric Traveling Salesman Problem enables known approaches to this problem to be adapted to the HFMDVSP. Despite the differences that exist in pratical terms, new light can be shed into solving this efficiently and being this a NP-Hard combinatorial problem even solutions that are more time efficient with slight loss in solution quality.

The public transportation business is always changing. New regulations emerge constantly and some flexibility is mandatory in this kind of decision support system. Partial solutions allow the user to interact more with the algorithm behind it, by using it taking in consideration some manually defined constraints that the system doesn't, by definition, support.

Chapter 4

Implementation Details

The solution was created to incorporate the systems OPT offers its customers, that aim to ease the planning operation of public transportation systems networks and schedules. This systems are further presented as well as architectural issues that had to be overcome in order to create a transversal solution for this matter.

4.1 Context

The company that serves as environment for the development of this thesis has been serving transportation companies for more than 20 years now and since the beginning has provided decision support systems that are transverse to the needs of their clients. The first management system, GIST 1, was developed more than 20 years ago, and since then has kept its name but has suffered a drastic evolution. Their systems is currently used in near 60% of the total public transportation sector in Portugal.

> " GIST's roots can be traced back to several independent research and development efforts, starting about 15 years ago both in the universities and in the main Portuguese transport companies." [eCdS00]

Smaller companies have different networks and challenges in this task and, recently, the need arose to create a new branch of GIST that would overcome this needs. This version is called GIST Light and is still in its embryonic form.

4.1.1 GIST 3.0

GIST 3.0 is the latest evolution from the original GIST, and is still in development. It consists, at the moment, of two different models: Network 3.0 and Planning 3.0. This platform is the result of years of experience in developing public transportation integrated systems for organising and planning tasks. This experience is the result of a close relation with R & D departments, in universities and public transportation companies themselves,

and a constant feedback on the quality of the provided solutions. This allowed this evolution to be aware of the needs of their costumers and to provide excelence in the functionalities that are offered. GIST platform is currently being used in the major Portuguese public transportation companies.

4.1.1.1 Network 3.0

Representing the network where the transportation system operates is the initial step of the planning process which starts by defining all points that play significant roles in the system operation. In GIST this process takes place in the Network environment, seen on Fig. 4.1.

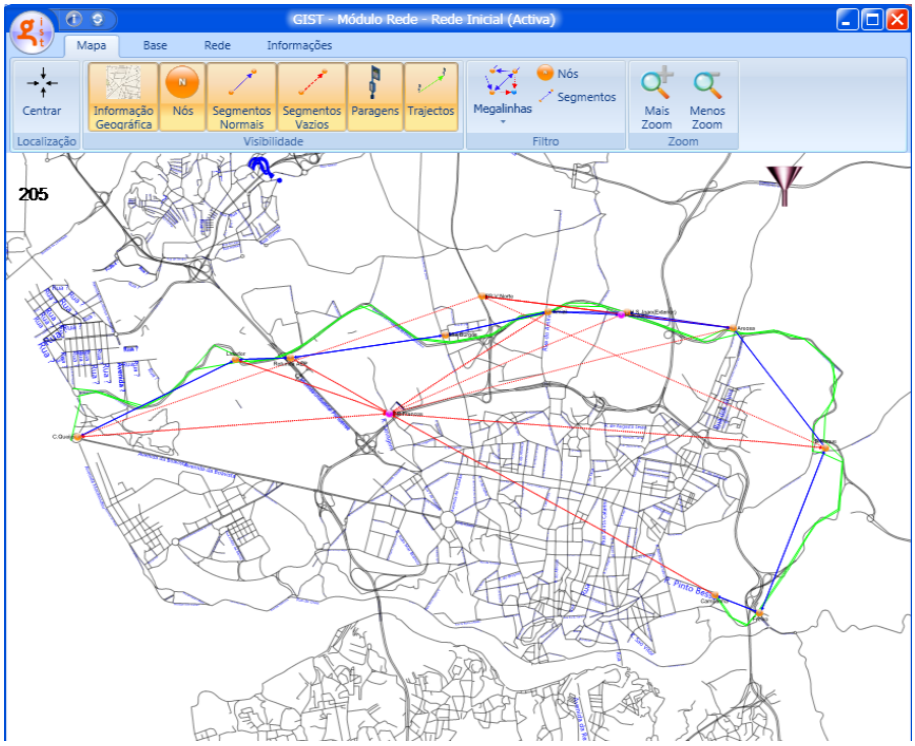

Figure 4.1: Network 3.0 Overview

The most basic entity of the network are the nodes, which represent, for instance, terminus, relief points (i.e. points where the drivers can be replaced) and important crossroads or depots. Once this locations are identified, segments that link them need to be defined.

Once the network has been settled, one can define paths that are ordered sets of segments along which trips are going to take place. A set of paths can form a route - the trips of a route are grouped together, forming a timetable that is viewed by the public as a unit. In this module, we also define the gist-lines, i.e., sets of routes and paths, probably with common segments, for which schedules are defined simultaneously.

4.1.1.2 Planning 3.0

Once the network is defined and demand is analysed it is possible to establish the trips to offer and define the public timetables. This is not possible without considering at the same time the operational tasks of scheduling vehicles and crew.

The Planning module, seen on Fig. 4.2,allows the schedule creation either for trips, vehicles and crew. It facilitates the production of public timetables and the assignment of vehicles to the trips that is supported by optimisation algorithms and by intuitive graphical representations of solutions.

The basic units of the Planning module are the gist-lines, which can be used in different contexts of demand, such as day periods, day types, and year season.

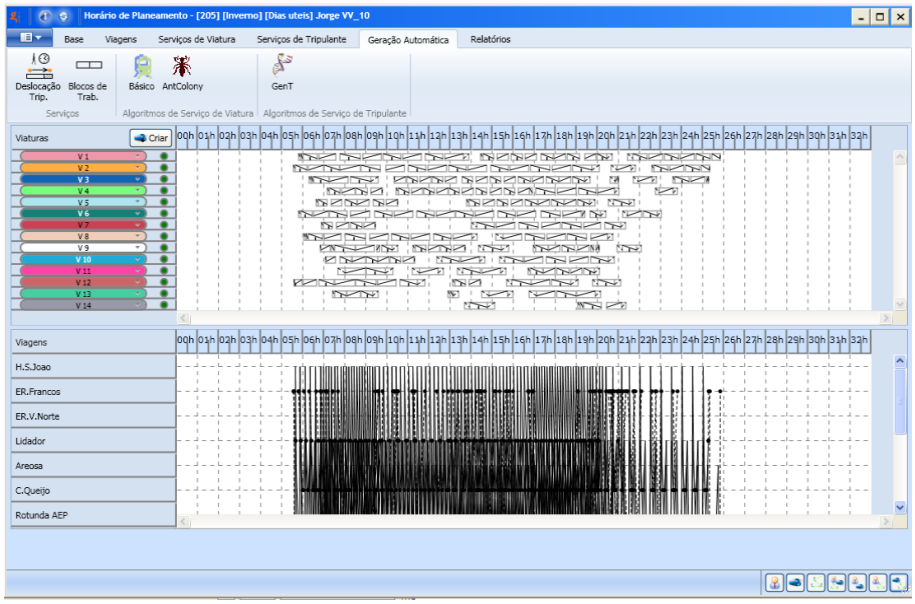

Figure 4.2: Planning 3.0 Overview

4.1.2 GIST Light

GIST Light as the name suggests is a lightweight platform similar to GIST, that aims to satisfy the smaller public transportation companies needs. This platform gathers the main characteristics of Network and Planning modules from GIST into a single platform that allows to perform planning operations with the same intuitive interface and basic concepts (Fig. 4.3). It offers great flexibility on the operational planning functionalities, such as defining the transport network the trips to offer and the scheduling to support that offer.

The lightweight version uses a lightweight data base, using XML files to store the network and scheduling information. The network consists of the same basic concepts,

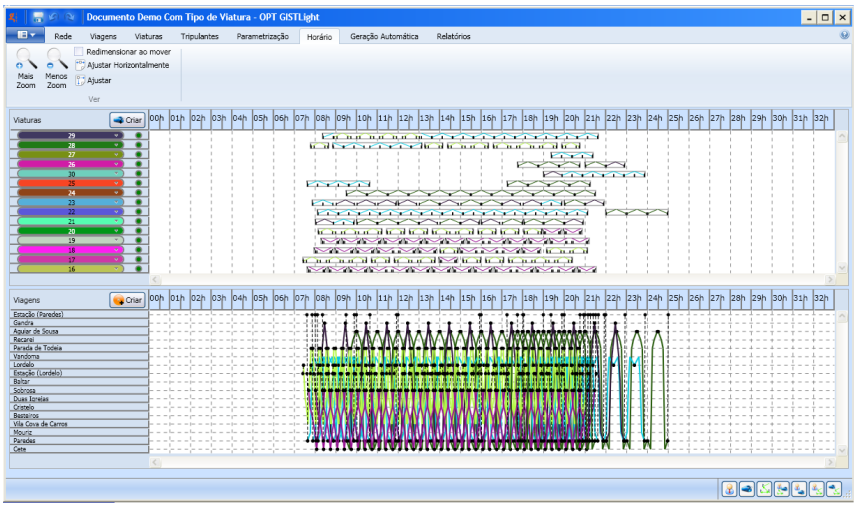

Figure 4.3: GIST Light Overview

such as Nodes, Segments, Paths and Lines. Since smaller companies usually perform the whole network planning tasks as a single problem, the gist-line concept is not applicable, so instead of having several gist-lines, gist light supports a single gist-line per file. The planning domain has simplifications since the requirements are slightly simpler, i.e. the year season concept is not applicable to this platform.

4.2 Addin Framework

Both systems have core modules, such as the ones that support the creation of networks and schedules, but also have several addins that allow to perform additional tasks. This addins are connected with the core module through the addin framework that supports different kinds of addins, i.e. report addins, algorithm addins, data exportation. This architecture allows the platforms to evolve with time without changing their heart and to be adaptable to future requirements without changing main functionalities.

The layered approach that constitutes the decision support is schematically defined in Fig. 4.4 which clearly separates different addins providing a sandbox environment for creating new functionalities for the core platforms in a transparent manner, that can be used by both systems.

The decision support algorithms are connected to the core through an adapter addin, that creates a transparency layer between the domain used in the platform and the domain of the algorithm. This algorithm adapter consists of three main modules: algorithm adapter, domain converter and user interface which role in the system is represented in Fig. 4.5.

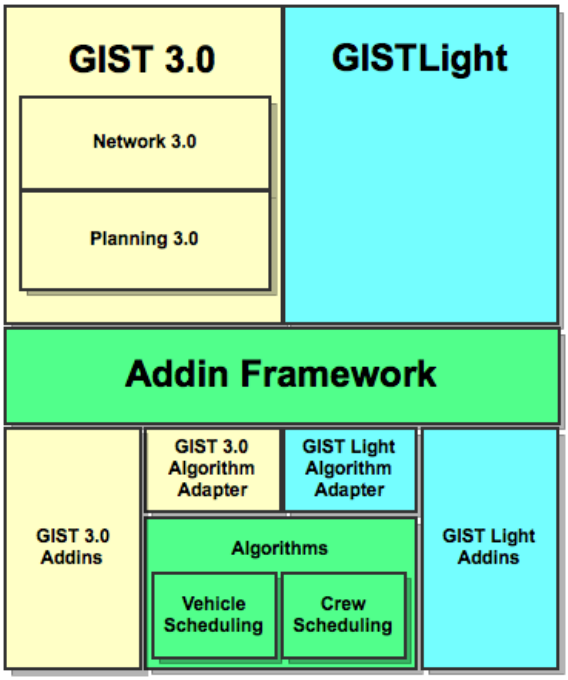

Figure 4.4: System Layers Overview

The user interface enables the end user to establish the parameter settings and run the algorithm. The Algorithm Adapter receives data from the User Interface and the information system and calls the Domain Converter to transform the data into a language that the algorithm understands and pass it to the algorithm. When the algorithm returns a solution to the adapter, it once again calls the converter to transform the data into the system data format completing the task.

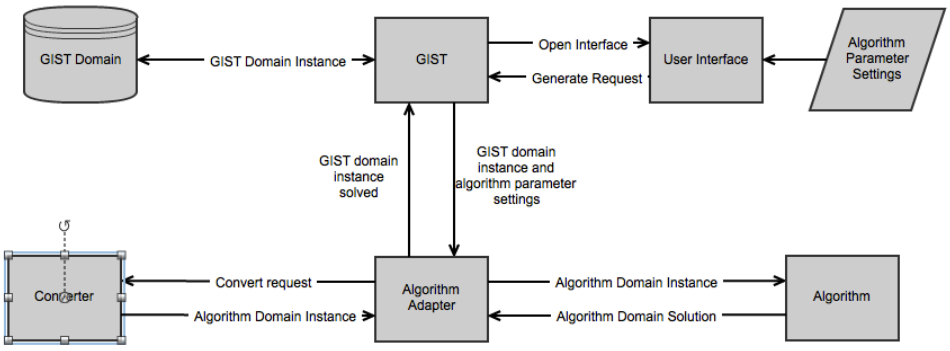

Figure 4.5: Algorithm Adapter Flow Chart

This Addin Framework intends to open this sort of tasks to be further explored in Academic enviroment, opening the GIST platform to exploration without losing the privacy

of the proprietary software. This can enable more sinergies between the company and universities and R&D departments with a simple sandbox environment that allows hands on approaches to be tested and developed.

4.3 Multiple Vehicle Type Scheduling Domain

This domain defines the information recognized by the algorithm, this plays a major role in implementation because it supports in a transparent manner the domains from both GIST Light and GIST 3.0, that have slight differences. The domain model is represented in Fig. 4.6

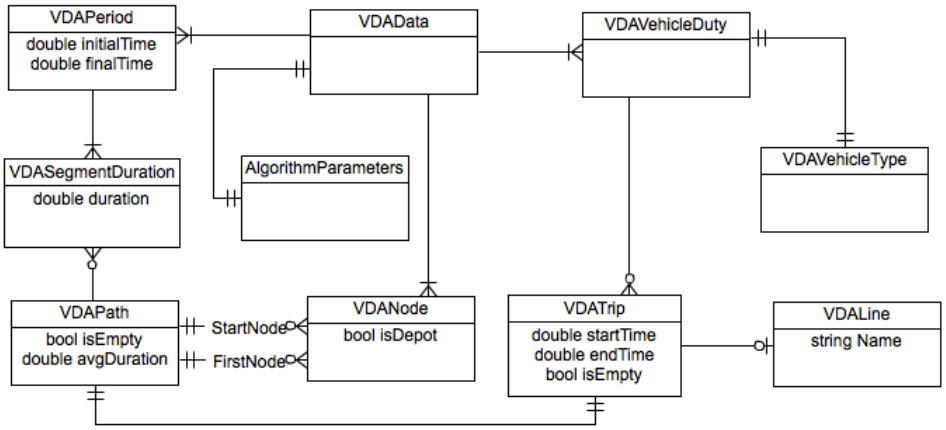

Figure 4.6: Domain Data Model

The most significant difference between GIST Light and GIST 3.0 is that in GIST Light a path always has the same duration and does not consider traffic constraints, while in GIST 3.0 a Period structure that considers different traffic conditions throughout a day is provided and varies with day type and year season.

A Period is a time interval and how the traffic conditions affect the speed on a given network which is represented as a speed percentile that when multiplied by the average duration of a given segment gives the time it takes to perform the segment at that time period. Periods must be defined for the whole day, which is set to thirty two hours enabling that trips ending at 23h59 can connect with trips starting at 00h01, so that durations can be calculated at any given starting time.

To create a transparency layer in this subject, every path has an average duration, that in case there are no periods, it is used as default duration. In case there are day periods defined, the value must be calculated considering the time the trip is set to start or end. This is specially important when creating empty trips required to link trips or depots.

This domain model serves as the unit of communication between the algorithms layer and the core module, the VDAData object is the input and output of the algorithm, that is interpreted in the Algorithm Adapter that changes the core module data with the solution.

4.4 Conclusions

The company where the system will be developed has a significant experience with the public transportation sector, and provides management solutions for the operational tasks they perform. Currently they offer two different systems with different targets, GIST aims major companies with big infrastructures and vast networks, while GISTLight is more compliant with smaller companies that operate in smaller transportation networks.

Both systems have an architecture that allows addins to extend the core functionalities, and a specific addin for vehicle scheduling algorithms that allow transparency in developing algorithms. This structure allows a single algorithm to be created for both systems. The transparency is created by the adapters that translate the core domain to the algorithm domain, and the opposite as well.

Chapter 5

Ant Colony Meta-Heuristic

Throughout the years several nature inspired heuristics have emerged. This heuristics are specially applied in optimisation problems, where some natural abilities are exploited to be simulated by algorithms capable of solving real life problems by analogy.

Presented by the work developed by Marc Dorigo [Dor92] Ant Colony Meta-Heuristics have been used to solve different optimisation problems based on the unique ability of ants to cooperate in the search for food by exploring the pheromone information. One of the most known applications of this algorithm is to the A-TSP problem. Hereafter an adaptation of this application to the HFMDVSP is presented.

5.1 Ant Colony Behaviour

The ants search for food starts with an ant wandering and finding food, while laying down a pheromone trail. Other ant also wandering eventually finds the pheromone trail and is likely to follow it and reinforce the pheromone trail. This biological feature by itself and the fact that pheromone trails evaporate with time help this task. The longer the path to food, higher is the the probability of evaporation before another ant follow that entire path, the opposite occurs with short paths, because ants will be more likely to follow the trail and reinforce it.

The Fig. 5.1 above represents this behaviour, in image 1 the first ant finds randomly a path to food and leaves the pheromone trail so the next ant can follow it with a higher probability than any other path. Given the low concentration of pheromones on all paths initially, the ants will eventually travel through the whole search space, as seen on 2, leaving pheromone trails with a concentration related with the distance to food. In the end as it is shown in 3, the ants converge to the shortest path,leaving the shortest path with much more pheromones than any other path.

It can be seen as a GRASP cooperative method, since it uses a greedy random heuristic by preferentially selecting connections with a higher pheromone concentration. Initially

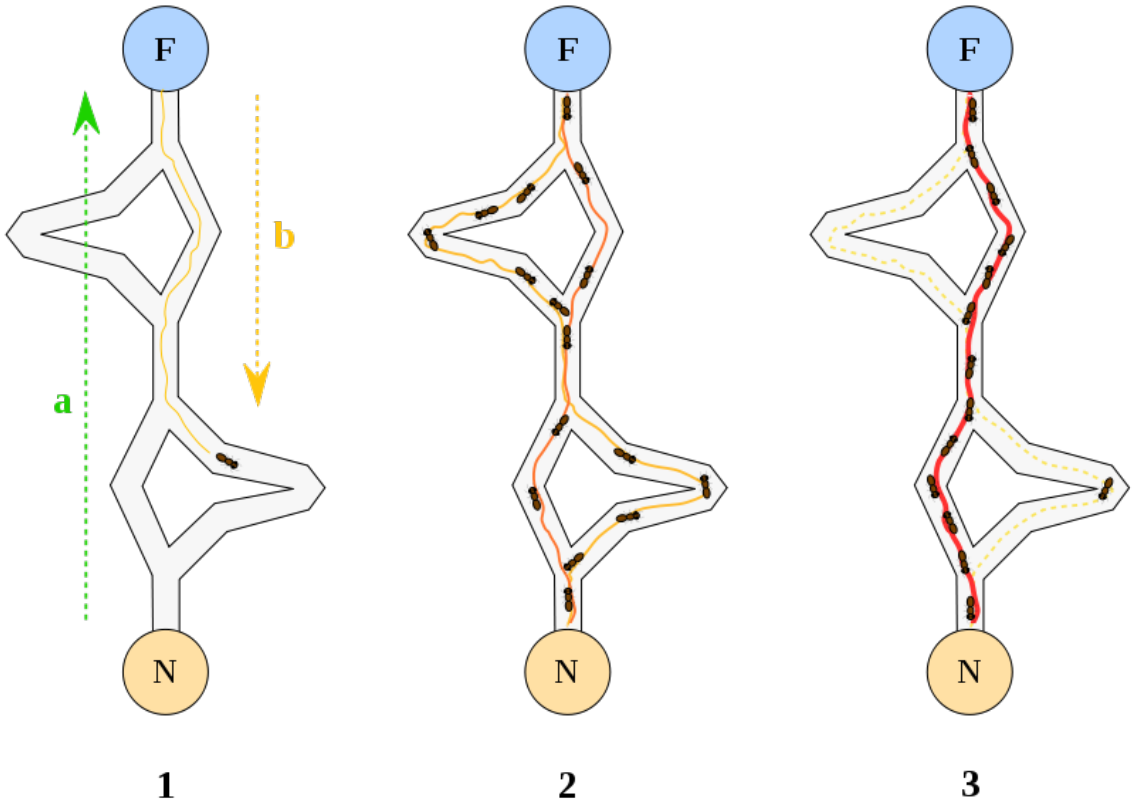

Figure 5.1: Ant's search for food behaviour representation

the pheromone concentration is based only on the cost of that connection and a calculated factor that balances the number of paths and the distance on each path, promoting exploration. As artificial ants start to find feasible paths, it adapts the connections pheromones based on the total cost of solutions that contain that link. The update process is done in two phases: the first is the local update and the second is the global update.

The local update rule (Eq. 5.1) promotes the exploration of different paths, reducing the probability of following the same path repeatedly. This update is done after every artificial ant finds its path, and has a low impact in the pheromone concentration. In the formulas presented pheromone concentration is represented as $ph(e)$ and L_{gb} represents the balanced weight of this link in the global best solution cost. The factor α is the local update rule factor that defines how this rule affects the system, as the factor ρ is the factor for global update rule.

$$ph(e) = (1-\alpha) * ph(e) + \alpha * \Delta Lo(e) \tag{5.1}$$

$$\Delta Lo(e) = \tau_0 \tag{5.2}$$

The global update, Eq. 5.3, is done after a number of artificial ants find their path. Those paths are then evaluated and the best path is selected, the links in the best solution are updated with a high impact on pheromone concentration.

$$ph(e) = (1-\rho) * ph(e) + \rho * \Delta G(e) \tag{5.3}$$

$$\Delta G(e) = \begin{cases} (L_{gb})^{-1} & if\, e \in globalBest \\ 0 & otherwise \end{cases} \tag{5.4}$$

The link selection, Algorithm 1, considers the pheromone density of each link and the costs related to it to calculate the probability of selection of that link.

Algorithm 1 Link Selection Algorithm

```
prob ⇐ RandomNumber(0,1)
if prob < ExploitationProbability then
  max ⇐ 0
  for all linkinlist do
    if link.Weight > max then
      max = link.Weight
      bestLink = link
    end if
  end for
  return bestLink
else
  totalCosts ⇐ 0
  for i = 0 → list.Size do
    tempCosts ⇐ link.Weight
    totalCosts+ = tempCosts
    probGap[i] = totalCosts
  end for
  probPick ⇐ RandomNumber(0,1) × totalCosts
  for i = 0 → list.Size do
    if pick ≤ probGap[i] then
      return list[i]
    end if
  end for
end if
```

In order to promote exploitation over exploration, the best link is used with a given probability ,Eq. 5.5, called the exploitation factor, and otherwise the previously presented probabilities are used in order to select the used link ,Eq. 5.6. This selection is formally presented bellow, where σ is a random value, σ_0 is an exploitation factor and S is a random variable selected according to the probability distribution given in the second equation.

$$s = \begin{cases} max\ ph(e) * \eta(e)^{\beta} & if\sigma < \sigma_0 \\ S & otherwise \end{cases} \tag{5.5}$$

$$p(e) = \begin{cases} \frac{ph(e)*\eta(e)^{\beta}}{\Sigma ph(e_i)*\eta(e_i)^{\beta}} & e, e_i \in CurrentEdges \\ 0 & otherwise \end{cases} \tag{5.6}$$

To define the initial pheromone concentration a static value is used, and then an artificial ant is set to determine a feasible path, and it's solution cost is used to redefine the pheromone concentration (Eq. 5.7). The definition of this initial value is specially important because if there is a huge gap between the initial concentration and the first value used to update the pheromone concentration it can make those links almost mandatory or irrelevant, so the closer this initial solution is to an optimal solution higher is the causality of that update. This causality improves the learning capacity of the algorithm.

$$p(e) = \frac{1}{firstSolution.Costs} \tag{5.7}$$

The cost of connecting the current node to the depot node only considers the trip required to do so, and no waiting time, which makes this links very likely to be selected, because other links usually have higher costs. In order to avoid the selection of links to depot from any trip, it was considered the cost of using another vehicle in the cost of those links.

5.2 Ant Colony Meta-Heuristic Approach

In this section an introduction to the Ant Colony Algorithm and its application to the HFMDVSP is presented. This algorithm is inspired in the proposal from Dorigo and Gambardella [DL97] on applying the Ant Colony System to the TSP and A-TSP and considers the special situation of the A-TSP presented in Section 3.4. A solution for the main problem found on this application of the algorithm, ensuring that the paths followed by ants are feasible, is presented. Some considerations on the algorithm basic functions such as the link selection algorithm are presented, and some specific situations that emerge in this application are also analysed.

The Ant Colony Algorithm 2 presented is an iterative algorithm that simulates the trajectory followed by a number of ants (colonySize) leaving pheromone trails that are used to measure the value of the connections. At each iteration it is verified if the best solution found is better than the previous best and when the algorithm ran for a given number of iterations (maxIter) without finding an improvement stops. The pheromone trails are updated with two previously mentioned rules, the local update and global update rule.

Algorithm 2 Ant Colony System Algorithm

```
bestEvaluation ⇐ ∞
initPheromone ⇐ CalculateInitialPheromone()
for all nodeinNodes do
  for all linkinnode.Links do
    link.Pheromone ⇐ initPheromone
  end for
end for
while iterations < maxIterations do
  while AntsNotFinished() do
    for i = 1 → colonySize do
      link ⇐ ants[i].SolveNext()
      LocalUpdate(link)
    end for
  end while
  colonyBestAnt ⇐ ComputeBest(ants)
  GlobalUpdate(colonyBestAnt)
  if colonyBestAnt.Evaluation() < bestEvaluation then
    bestAnt ⇐ colonyBestAnt
    bestEvaluation = bestAnt.Evaluation()
  else
    iterations++;
  end if
end while
return bestAnt.Solution()
```

5.2.1 Feasible Solutions

One of the main issues in this approach was to find feasible solutions for the problem. The problem arises mainly due to the fact that, in some problems, nodes are not heavily connected, and the decision tree of building a path may contain leaves that do not correspond to feasible solutions (Fig. 5.2) .

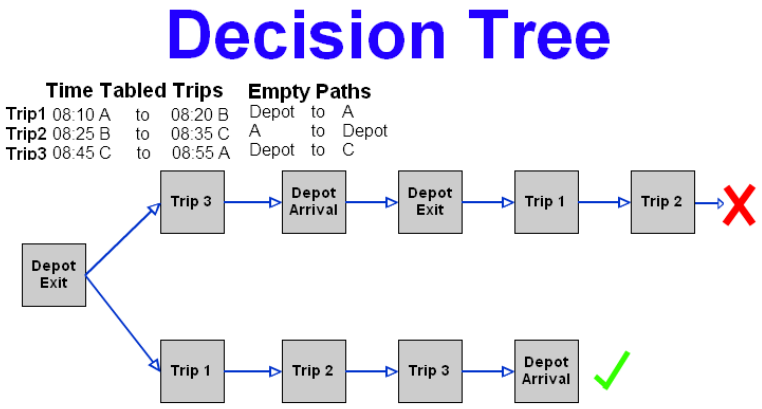

Figure 5.2: Example of a path construction tree with two paths, one feasible and one unfeasible

This matter is really important because it avoids useless path creation of unfeasible solutions, making the exploration of possible solutions more effective. some preliminary studies indicated that without an effort to avoid unfeasible solutions the heuristic of searching for the least expensive connection would generate mostly unfeasible solutions. The analysis of this studies concluded that this was mainly due to the fact that it would hard for the algorithm to understand early that some connections would lead to unfeasible solutions and would mostly lead to after diminish the value of pheromones for a several connections it would find a small neighbourhood of feasible solutions that would rarely end up to be the best solutions. This analysis lead to create a method that would eliminate unfeasible paths but capable of exploring all feasible solutions.

5.2.1.1 Unlinked Trips Sub problem

If a trip can be connected to a depot, it will always have an available connection, because depots can be visited more than once, except if the trip is set to another depot, and then this link will no longer be available. Filtering the graph by assigning the nodes that are not connected from or to all depots that they can be assigned is a relaxed problem that ensures that all trips can be connected. This relaxed sub problem allows to avoid unfeasible solutions that can represent a high percentage of the reachable solutions, enhancing the performance of any search heuristic. On the other hand this problem may become very

complex, that is the case presented in Fig. 5.3 which has a ratio of 85% unlinked nodes over the total nodes.

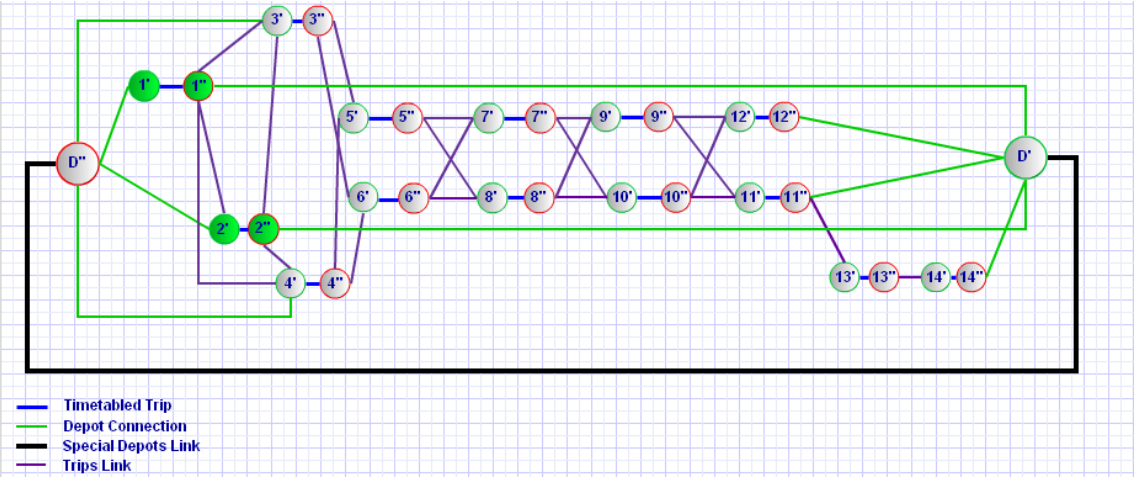

Figure 5.3: A CBN layer sample focusing on the unlinked trips sub problem

5.2.1.2 Formulation

The UTSP problem consists of finding sub tours S from linked nodes to linked nodes in a way that all unlinked nodes are assigned to one of them and performed in a feasible way. After defining the sub tours, the unlinked nodes connections are reduced to the ones from its sub tour, so that every path becomes a feasible one.

Let U be the group of nodes that don't have connection from or to all depot nodes, considered the unlinked nodes and the group C_{n1} be the available connections of node $n1 \in N$ and $D_{n1} \subseteq C_{n1}$ be the set of connections with depots. Some nodes from the unlinked nodes $U_c \in U$ may not require to be associated to a sub tour if they have connections with a depot, but not all of them, although it must be ensured that they can only be assigned to a depot that they can connect to, removing all possible connections to other depots. For ease of comprehension this will be treated as a special sub tour that only considers the start and end node of a given trip. The sub tours may not have a defined vehicle type, but they must have a group of vehicles that can perform it in order to remove from the main model the connections that would connect this sub tour to any other depot.

The first constraint is that every unlinked node must be related to one and only one sub tour that defines the mandatory links for that node.

$$\forall\, n \in U\ \exists\, s \in S \rightarrow n \in s \tag{5.8}$$

The second constraint is that the first node $n1$ of a sub tour must be connected from at least one depot, and also the last node $n2$ must be connected to at least one depot that also belongs to the connected depots of the first node.

$$\begin{aligned} D_{n1} &\neq \emptyset \\ D_{n1} &\neq \emptyset \\ D_{n1} \cap D_{n2} &\neq \emptyset \end{aligned} \tag{5.9}$$

The third constraint ensures that all the connections C^{s1} of a sub tour $s1 \in S$ have the same feasible vehicle type group. Let FV_c with $c \in C$ be the group of feasible vehicle types for any connection, that considers the group Vt_i and Vt_j that define which vehicle types can perform the connected trips i, j, this constraint can be represented as in Eq. 5.10.

$$\forall\, s \in S \,\forall\, c1, c2 \in C^s \rightarrow FV_{c1} \cap FV_{c2} \neq \emptyset \tag{5.10}$$

5.2.1.3 Approach

To solve the Unlinked Trips Sub Problem an algorithm for assignment was established based on the ant colony behaviour as well. The main difference is that in this case the algorithm needs to have the ability of return in a decision if it leads to an impossible solution and the connections are not selected in sequence.

The connections are selected using the same selection algorithm from the Ant Colony System, but the next node to assign is selected through a first fail heuristic that considers the number of connections available for each node as well as the number of connections to depots it has, so that the first point of possible failure is considered first. Since a node can be connected with a depot, but not all of them, the nodes without any connection to any depot are considered in first place and when considering nodes with depot connections the connections with other nodes are considered first, and only if none of them lead to feasible solutions the hypothesis of that node being connected to the available depots is considered.

To avoid stepping into sub tours that would lead to unfeasible solutions for the UTSP a method was created to validate at each step of the algorithm, or every new connection selection, if the remaining unlinked nodes can still be assigned in a feasible way. If this method returns that it is not possible to assign the remaining nodes, it steps back until the previous decision that offers unexplored alternatives.

5.2.2 Vehicle Selection

The selection of the next vehicle type in the solution construction occurs when a given vehicle duty reaches any depot arrival, so there are special connections between depot arrivals and depot exits that have a special cost structure that is considered by the algorithm

to select the depot/vehicle type of the next vehicle duty to create. This connection can be used more than once, so the update on pheromones happens more than once per solution, so the dominant factor is actually the cost of the connection and there is a serious interest in give a higher probability to select least expensive vehicles whenever it is possible, but also to allow that the expensive ones to be selected even when they are not mandatory. This equilibrium is not easy to achieve, and depends highly on the user and how he understands the real difference between using one vehicle type over another one which is defined by the previously presented **Vehicle Type Factor**. A vehicle type may become mandatory if any other vehicle type offers no connections, this can happen whether by the fact that trips only have that determined vehicle as a feasible vehicle type, or by means of the mandatory links defined in the UTSP problem that oblige to use only that vehicle type. This can occur also during solution construction when all the pairs depot/vehicle of a given vehicle type run out of connections with trips.

5.2.3 Connections with and from Depots

The construction of a vehicle duty starts by selecting the first trip to be performed after the vehicle leave the depot, and this is a special case because the implicit cost of performing a connection between the depot and a trip is only related with the empty trip, if required, from the depot location to the start of the trip. The issue that this matter arises is that this way the algorithm wouldn't consider whether the trip it was selecting was the last trip to start or the first, and in the case of being the last trip, it wouldn't be possible to connect it with any other trip, creating a vehicle that would only serve that last trip. The same matter happens when we're considering connecting to another trip or connecting to a depot, because the connection cost to a depot can be inferior to connecting to another trip, specially if trips are sparse in time, and the algorithm would be blind to the fact that it is rather likely to find an expensive solution by selecting the depot over any trip.

To avoid this blindness of the algorithm in this cases special cost structures had to be defined for links from a depot and to a depot.

In the case of selecting the first trip to perform, it is preferable to select trips that start early enhancing the chances that it can connect with other trips that start later, so to achieve a good balance in selecting the first trip the difference between the start of the trip and the initial time (00h00) was considered as idled time of the vehicle for that specific connection, which is only considered in the selection of connections and not on the evaluation of the solution.

Avoiding the early end of a vehicle duty by connecting with the depot over selecting another trip the cost of using another vehicle is considered in the connections to the depot, even though for fitness evaluation of a solution this is only considered in the connections between depot arrivals and depot exits. This way it becomes more likely to finish a vehicle

duty only when there are not any other choice, or the available options are not really that interesting, even though there is always a probability of doing so.

5.3 Parameterisation

The decision support system has several parameters that must be defined, some of them are defined in code, and the user can not change them, while others can be adapted by the end user given the problem instance being solved.

5.3.1 User Parametrisation

The parameters that can be defined by the user are divided in four groups: *Costs and Penalties* , *Create Empty Trips* , *Partial Solutions*, *Advanced Options*

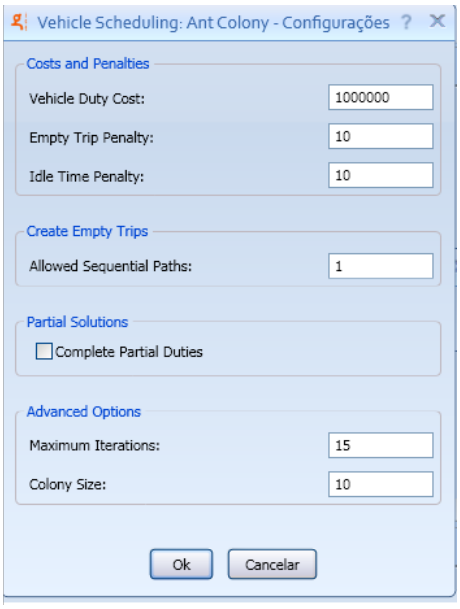

Figure 5.4: Ant Colony Parameterisation Interface

The *Costs and Penalties* parameters influence the objective function calculation. The first parameter defines the cost of using a generic vehicle duty, that usually has an high value in order to achieve a solution with the least possible vehicles, but can be adapted to favour the trips sequence quality over vehicle number. The second and third parameter were presented in Section 3.1.4 and represent the weight of the idle time and the time performing empty trips have in the evaluation function.

The *Create Empty Trips* sub menu only has one parameter that defines how many paths can be used in sequence to satisfy a single connection, the way the user interacts with the system may change, and he can define paths for every possible path, or just some short connections that he wants to be used in sequence by the algorithm. This parameter could

not be required but it helps the user to adapt his way of using the system to the use of the algorithm.

The *Partial Solutions* group includes the parameter that defines whether or not a partial solution can be altered. This was further discussed in Section 3.5.

The *Advanced Options* are related with the termination criteria and the morphology of the ant colony. The first parameter is the *Maximum Iterations* that defines how many iterations can the algorithm run without finding a better solution before stopping. The second parameter is the *Colony Size* that defines the number of artificial ants that search for a path at each iteration. This advanced options group can become valuable specially when the cases being treated are to small or too big and the default settings may not by the most appropriate.

5.4 Conclusions

The Ant Colony Metaheuristic simulates the behaviour of ants when searching for food and the pheromone trails and has been applied to several combinatorial problems.

One of the most known applications of this heuristic is to the Asymmetric Traveling Salesman Problem. This heuristic promises to be a good approach to the Heterogeneous Fleet Multiple Depot Vehicle Scheduling Problem ,from the comparison previously made in Section 3.4.1 with the A-TSP.

This adaptation presents several problems that must be overcome. One of them is the existance of unfeasible paths, that originally are not considered in the A-TSP, which brings new complexity to the problem. This problem can be overcome by grouping the trips that can not be linked with depots, since in the worst case scenario, after doing so, all trips will be able to be linked from and to the depot, allowing the flow constraints to be satisfied. The other problem to overcome comes with the fact that some nodes can be visited more than once, the depots. This problem exists mostly because of the special structure that it's connections have, since vehicles they don't have a time associated and vehicles can come and go at any time with the same considered cost. To overcome this a special costs structure must be used.

Some parameters of the algorithm must be handed to the user to define, this parameterisation interface must be easily understood by the user and flexible enough to explore it.

Chapter 6

Results

The testing platform was the environment used by many public transportation companies in Portugal to manage their network and planning data, GIST. The database used for the performance tests, in Section 6.1 and 6.2, was a copy of the one that a major company operating in Porto uses in their daily planning operations. This platform already provides a planning support system that allows the Vehicle Scheduling Problems to be solved automatically, even though it does not support heterogeneous fleets and vehicle type constraints. This method is based on a quasi-assignment algorithm, developed by Paixão and Branco [PB87]. It was used as a benchmark for situations in which vehicle type constraints are not defined. These tests allowed to verify the solutions quality, and efficiency.

The number of trips and the amount of unlinked nodes in a given instance play a major role in the algorithm performance. In order to understand how they affect the results the dataset is divided into categories that characterise the number of trips and the percentage of unlinked nodes overall. This dataset characterisation is presented in Table 6.1

	Unlinked Nodes < 30 %		**Unlinked Nodes > 30 %**	
Trips Number	Instances	Average Trips	Instances	Average Trips
T > 150	4	180,6	0	0
100 < T < 150	9	129,33	2	103,5
50 < T < 100	20	78,35	2	71
20 < T < 50	7	17,54	4	34
T < 20	10	11,25	4	10

Table 6.1: Dataset characterisation on instance number and average trips per instance

The dataset used for the tests in Section 6.3 consider vehicle type restrictions with a heterogeneous fleet and multiple depots, this data was a reproduction of the data from a company operation in Lisbon, which is hereafter characterised.

The tests were made in a computer equipped with 4GB DDR2 ram memory, and an Intel i3-540 processor running Windows XP Professional Edition with Service Pack 3.

6.1 Time Performance

The time the algorithm takes to provide a solution plays an important role, since the previous algorithm provided almost instant solutions, and the end users are used to that level of performance. The time measured considers only the time spent by the algorithm while searching for solutions, and was analysed by the previously defined categories of trip number and unlinked nodes percentage.

	Unlinked Nodes < 30 %	**Unlinked Nodes > 30 %**
Trips Number	Average Time	Average Time
T > 150	72,01s	—-
100 < T < 150	21,19s	4,00s
50 < T < 100	3,21s	0,90s
20 < T < 50	0,36s	0,41s
T < 20	0,09s	0,07s

Table 6.2: Time Performance for the Ant Colony Approach

Understanding how the algorithm behaves when the complexity grows is extremely important and the time performance of the algorithm has a strict relation with the number of trips in the scheduling task. This increase in complexity when the trip number increases can be explained by the fact that with more trips in the schedule there will be more possible connections after any given trip which implicitly increases the number of links to consider. The analysis of the results found for the dataset allowed to understand that time increases exponentially when the number of trips increase as is shown in Figure 6.1.

6.2 Quasi-Assignment Algorithm Comparison

The Quasi-Assignment Algorithm that is present in the current solution provided to the clients only allows one depot and one vehicle type, it has very basic capacity in comparison to the requirements of the companies using it, that mostly use it as a basis to readapt manually.

Even though it is basic when considering the requirements it is a deterministic and optimal solver that can be considered as a lower bound for the solution proposed. It was tested against the proposed solution and analysed in terms of average deviation between the basic approach and the solution presented for tests that consider a single depot and a single vehicle type. It is expected that the solutions found by the Ant Colony approach

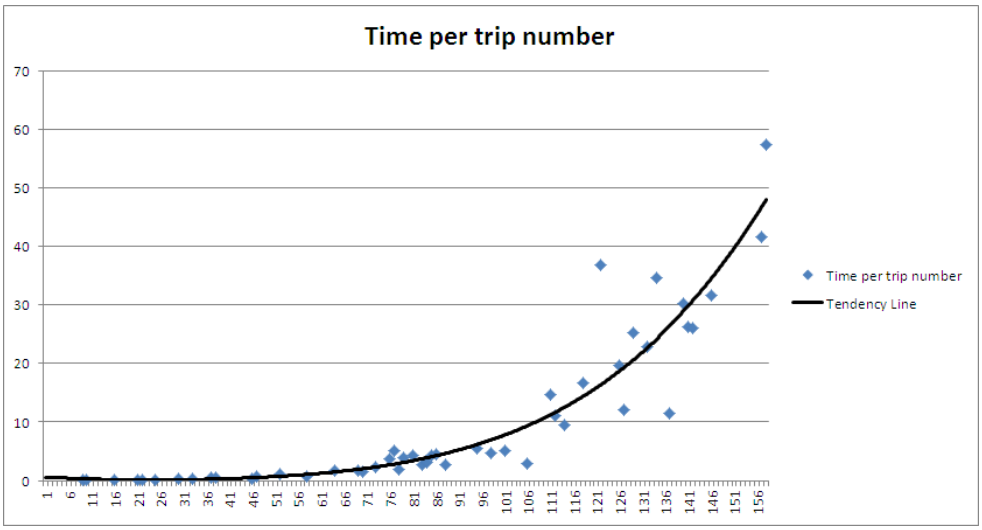

Figure 6.1: Tendency Line for time when the number of trips increases

perform slightly worse than the basic approach in this cases, but the goal is to prove that the deviation is very low, and so the methodology is useful. Results are presented in Tab. 6.3 where the difference between the costs of the two tested approaches in presented in percentage given by Eq. 6.1, where *acs* represents the total costs of the solution found by ant colony and *basic* represents the costs of the solution by the basic approach.

$$diff = \frac{acs - basic}{basic} * 100 \tag{6.1}$$

	Unlinked Nodes < 30 %	**Unlinked Nodes > 30 %**
Trips Number	Difference %	Difference %
T > 150	0,59%	—-
100 < T < 150	0,27%	0,02%
50 < T < 100	0,67%	0,54%
20 < T < 50	0,11%	0,00%
T < 20	0,00%	0,00%

Table 6.3: Difference Between the results from the Ant Colony System and Basic Algorithm

The results show that the algorithm performs very close to optimal solutions, with very small differences. A qualitative analysis of the results also shows that for the schedules in STCP there wasn't a solution with more vehicles required than the one provided by the basic algorithm, only slight differences in the vehicle sequences.

6.3 HFMDVSP Qualitative Results Analysis

Select different cases and different weights for each vehicle type and show qualitative results compared to hand made solutions.

The data set used to test with real life multiple vehicle type and multiple depot instances was based on data provided by a company that serves in Lisbon and currently uses the GIST system to perform the planning tasks. Since the data was produced in previous versions of gist, it still didn't provide information on vehicle types restrictions. The approach to generate vehicle type restrictions was to cross information between vehicle duties and each vehicle duty type allowed to establish scenarios. This scenarios allow to have an overlook on how the algorithm behaves in different vehicle restrictions use cases.

Vehicle Type restrictions can be totally restrictive where every trip can only be performed by a given vehicle type, partially restrictive where some trips can be performed by only one vehicle type and the other have no restrictions, or can come in a completely irregular manner where some trips can be performed by all vehicles, others have a restrict group of vehicles and others have a single feasible vehicle type.

The problem instance used for this analysis has two vehicle types : Standard and Midi. The standard vehicle cost factor was considered 4,0 and the midi vehicle cost factor was considered 5,0 . In the original scheduling problem 584 trips are assigned to two vehicle duties preformed by Midi vehicles and twenty Standard vehicles, while in the smaller one there are 269 trips performed also by two Midi vehicles and only eight Standard.

6.3.1 Totally Restricted

Totally Restrictive instances can be seen as a different problem instance for each vehicle type, since the trips that can be performed by one vehicle type can't connect with trips from other vehicle types and vice versa. The results are satisfying even though the comparison with the result from the basic algorithm only can be seen as a lower bound because its scheduling doesn't consider that some solutions are unfeasible in terms of vehicle types. The most important comparison here performed is with the original scheduling that was a manual improvement of the basic approach, since this solution already considers vehicle type restrictions and is currently active. The two methods that were previously developed in OPT, Firsts and Clusters were also tested but the results were as disappointing as it was expected, as seen on Tab. 6.4.

The result from the Ant Colony approach in this smaller case were very satisfying since it was able to provide a solution with the same number of vehicles for each type and the value for the total costs was very close to the ones from the original scheduling, even in a case where 269 trips are considered.

	Midi Vehicles	Standard Vehicles	Total Costs
Original Solution	2	8	50628200
ACS Solution	2	8	51038220
Firsts Solution	2	10	Unfeasible Solution
Clusters Solution	5	14	Unfeasible Solution

Table 6.4: Results Analysis for the Totally Restricted Smaller Problem

This totally restricted case doesn't represent the bigger problem in vehicle type restrictions since the trips that can be performed by one vehicle can't be performed by the other and so it is as hard as solving the problem for each vehicle separately.

6.3.2 Partially Restricted

Partially Restrictive instances are a more complex scenario because all trips can be connected even though it is preferable to group trips from that are restricted to the most expensive vehicle type in order to achieve a solution with fewer expensive vehicles.

In this specific case the vehicle type that must be considered as expensive is the Midi vehicle type, because its cost factor is higher. In this test the smaller test case was used with 122 trips being restricted to use the Standard Vehicle, 84 trips forced to use Midi Vehicle, and 63 trips with no restriction. The ideal scheduling for this scenario requires only two Midi Vehicles, being the remaining trips satisfied by the Standard vehicle type. The results presented in Tab. 6.5 show that this restrictions structure makes the problem more difficult to solve efficiently, since it increases the search space, by allowing some trips to be connected with trips that should only be performed by a given vehicle type.

	Midi Vehicles	Standard Vehicles	Total Costs
Original Solution	2	8	50628200
ACS Solution	3	8	57594860

Table 6.5: Results Analysis for the Partially Restricted Smaller Problem

This restrictions were also tested on the original scheduling problem, but with restrictions being applied only to the Midi vehicle type, where trips that should be performed by the Standard vehicle type are allowed to be performed by any vehicle type. The results for this test are shown in Tab. 6.6

6.4 Parameter Settings

The parameter settings was based on the conclusions from the original application of the algorithm to the A-TSP problem [DL97].

	Midi Vehicles	Standard Vehicles
Original Solution	2	20
ACS Solution	8	15

Table 6.6: Results Analysis for the Partially Restricted Original Problem

The exploration factor that defines the probability at each decision of using a greedy heuristic that selects the currently understood as the best decision was set to $\sigma_0 = 0,8$.

The update factors were subject of small thorough qualitative study that shown that when the value of the local update factor was close to the global update factor, as was proposed by Dorigo and Gambardella with local update factor $\rho = 0,1$ and global update factor $\alpha = 0,1$ the pheromone evaporation would occur to fast to take advantage of the pheromone trail. The equilibrium that allows local update rule to be efficient in avoiding that consequent ants find the same solution and global update rule can influence ants to explore good characteristics of the search space was found with the values $\rho = 0.1$ and $\alpha = 0.2$.

The initial pheromone value was achieved by applying the Equation 5.7 with a first-Solution found with pheromone value defined to a static value, this value was considered $\tau_0 = 0.001$. Even though this value is not a good value to be considered as the initial pheromone value it allows the algorithm to find a solution that is based only on the cost of the connections and use this solutions costs to find a good initial pheromone value to start the process.

Colony Size value used was 10, the number that was suggested as having the best performance. This value was subject to tests that attempted to prove that this value was really the best, but after running several tests results were rather inconclusive, and even though it is known that in smaller instances smaller colonies can perform as well consuming less time it was considered as a good value to make the testing bed homogeneous in this matter.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The problem being solved is a NP-complex combinatorial problem, and few studies for this specific extension of the Vehicle Scheduling Problem are found in literature, and it represents a major improvement in public transportation systems. The algorithm created will be available in the GIST systems, that is used by almost 60% of the total Portuguese public transportations sector, and presents great value not only to the companies but to all Portuguese that use public transports. The companies using this system operate more than 6000 vehicles [eCdS00].

The results achieved indicate that the approach presented were very satisfying, the main goal of creating a method that could give a good trade-off between closeness to optimum and time to perform was achieved. The difference in terms of costs between the algorithm and the lower bound of the basic algorithm was close enough to create a valuable solution for real world problems. The time performance of the algorithm in comparison with the basic algorithm already presented was not so close that allows to make such a clear statement about how in day-to-day scheduling operations it can be accepted. Even though the times are not prohibitive for daily use the accommodation of the end users with almost instant solutions from the basic algorithm may become an issue in terms of acceptance of use.

The value created with this system intended not only to be the capacity of solving the extended problem with consideration for the vehicle type of the vehicle duties, but also to create a decision support system that provides more ways to use than the previous solutions. The ability to use partial solutions to achieve complete solutions becomes very useful to the companies using the system, since the constant change in the requirements for valid assignments can make any system obsolete and demand hand made changes to the assignments obtained.

The statement that meta heuristic methods can perform well enough to be used in

real world problems in the extended Vehicle Scheduling Problem was verified with an approach that simulates the ant colony behaviour when searching for the closest path to food. The Ant Colony System had good performances that show how this methodologies when adapted can bring new alternatives for solving real life optimisation problems.

The problem of dealing with unfeasible solutions was found in preliminary studies of possible search operators that could gather the best characteristics of a group of feasible solutions and combine them into a better solution, but this proved to be really hard to achieve since most of the best characteristics were not compatible. The hypothesis of using column generation to introduce new connections not present in the solutions being crossed in a way that best connections from both would become compatible may bring new light to this problem.

During the development of the thesis a peculiar case was found in which a company uses two different labels in their transportation system, and also uses three types of vehicles. This problem has a special interest because the company doesn't want their costumers to change their habits and so doesn't want to change the image the vehicles that serve some lines to change, and so they have to deal with six different types of vehicles and a special case of restrictions. This in example that shows that this kind of planning considering vehicle types is already being done in most companies and that the decision support systems currently do not provide efficient help in this matter, so it is solved by hand. This algorithm provided this company a simpler and significantly better way of dealing with the problem.

7.2 Future Work

A more thorough study of the termination criteria and how it can be adapted to problem instance characteristics could enhance the time performance.

Future work can come from adapting already known solutions for the Asymmetric Traveling Salesman Problem even though they may not be applicable to the interpretation of the HFMDVSP problem as an A-TSP. The main identified issue in doing this adaptation is related with the unfeasible paths in this case that in the original problem A-TSP do not exist and represent a serious barrier.

One methodology that could also be subject of future work is considering unfeasible solutions during the search procedure that could allow a better exploration of the search space and enhancing the causality of search operator. Although accepting unfeasible solutions during the search procedure can improve the capacity of algorithms in finding better neighbourhoods a good balance is required between the distance from feasibility of solutions and the penalisation of this situations. Otherwise characteristics of unfeasible solutions may never be translated into good feasible solutions.

The problem of finding feasible solutions for the problem instances revealed as a core problem to find optimal assignments in this approach. The approach presented does overcome the problem, but in some specific instances it shows that it may not do it in the most desirable way, since it seems not to explore all the possible solutions in this instances and so it may not be able to find the optimal solution even when enough computational time is given. Further work in this chapter can make the solution even better although this sub problem itself is a complex combinatorial problem that presents a fair challenge.

Tests with vehicle type restrictions suggests that results for bigger problems are not very satisfying, even though they are significantly better than the results from the previous approaches, and further work must be done in order to understand how to avoid using expensive vehicles when it is not necessary even in situations where trips are allowed to be performed by any vehicle.

References

[BCG87] A. Bertossi, P. Carraresi, and G. Gallo. On some matching problems arising in vehicle scheduling models. *Networks*, pages 17:271–281, 1987.

[BDL$^+$97] H. Bersini, M. Dorigo, S. Langerman, G. Seront, and L. Gambardella. Results of the first international contest on evolutionary optimisation (1st iceo). *IRIDIA*, pages 611 – 615, 1997.

[BGAB83] L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and scheduling of vehicles and crews: the state of the art. *Computers and Operations Research*, pages 10(2):63–211, 1983.

[BK09] S. Bunte and N. Kliewer. An overview on vehicle scheduling models. *Public Transport*, 1:299–317, 2009.

[BRK78] L. Bodin, D. Rosenfield, and A. Kydes. Ucost: A micro approach to a transit planning problem. *Journal of Urban Analysis*, pages 4:47–69, 1978.

[CDFT89] G. Carpaneto, M. Dell'Amico, M. Fischetti, and P. Toth. A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, page 19:531– 548, 1989.

[DFT93] M. Dell'Amico, M. Fischetti, and P. Toth. Heuristic algorithms for the multiple depot vehicle scheduling problem. *Management Science*, pages 39:115–125, 1993.

[DL97] M. Dorigo and L.Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, Vol I.*, pages 53 – 66, 1997.

[Dor92] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano., Milano, Italie, 1992.

[eCdS00] J. Falcão e Cunha and J. Pinho de Sousa. The bus stops here - gist: Decision support system for public transport planning. *OR/MS Today, Vol.27 n^{o}2*, pages pp. 48–53, 2000.

[FM96] B. Freisleben and P. Merz. Genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. *Proc. IEEE Int. Conf. Evolutionary Computation, IEEE-EC 96*, page 616–621, 1996.

[HMS06] A. Hadjar, O. Marcotte, and F. Soumis. A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. *Operations Research*, page 54(1):130–149, 2006.

[KMS02] N. Kliewer, T. Mellouli, and L. Suhl. A new solution model for multi-depot multi-vehicle-type vehicle scheduling in (sub)urban public transport. *In Proceedings of the 13th Mini-EURO Conference and the 9th meeting of the EURO working group on transportation*, 2002.

[KMS06] N. Kliewer, T. Mellouli, and L. Suhl. A time-space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, pages 175(3):1616–1627, 2006.

[LÖ97] A. Löbel. *Code-Generation On-the-Fly: A Key to Portable Software*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 1997.

[LÖ98] A. Löbel. Vehicle scheduling in public transit and lagrangian pricing. *Management Science*, pages 44,1637–1649, 1998.

[LÖ99] A. Löbel. Solving large-scale multiple-depot vehicle scheduling problems. *in Nigel H.M. Wilson, editor, Computer-Aided Transit Scheduling, Lecture Notes in Ecomonmics and Mathematical Systems*, pages 193–220, 1999.

[LK81] J. Lenstra and A. Kahn. Complexity of vehice routing and scheduling problems. *Networks*, pages 11(2):221–227, 1981.

[MP92] M. Mesquita and J. Paixao. Multiple depot vehicle scheduling problem: A new heuristic based on quasi-assignment algorithms. *Computer-Aided Transit Scheduling*, pages 167–180, 1992.

[NG09] S. Neves and T. Galvão. The heterogeneous fleet multi depot vehicle scheduling problem. *14º Congresso da APDIO*, pages 139–147, 2009.

[PB87] J. Paixao and I. Branco. A quasi-assignment algorithm for bus scheduling. *Networks*, page 17:249–269, 1987.

[PDHH06] A. Pepin, G. Desaulniers, A. Hertz, and D. Huisman. Comparison of heuristic approaches for the multi depot vehicle scheduling problem. Technical report, Econometric Institute, 2006.

[Rei08] G. Reinelt. Tsplib. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/, 2008.

[RRP] J. Ramos, L. P. Reis, and D. Pedrosa. Solving heterogeneous fleet multiple depot vehicle scheduling problem as an asymmetric traveling salesman problem. *EPIA2011 - 15th Portuguese Conference on Artificial Intelligence*.

[RS94] C. Ribeiro and F. Soumis. A column generation approach to the multiple depot vehicle scheduling problem. *Operations Research*, pages 42,41–52, 1994.

[Wei09] T. Weise. *Global Optimization Algorithms - Theory and Application*. Available on www.it-weise.de/projects/book.pdf edition, 2009.