**POLITECNICO DI MILANO**

**Faculty of Engineering**

**Department of Electronics and Information (DEI)**


# OPTIMIZATION TOOL BASED ON MEMORY ADAPTIVE HEURISTICS AND INTEGER LINEAR PROGRAMMING FOR HOME CARE SCHEDULING PROBLEMS

**Supervisor: Prof. Federico Malucelli**

**M.Sc. Thesis by:**

**Helio Tadashi OMOTO**

**Matr. 734281**

**Academic year: 2011/2012**

# ABSTRACT

The project has the purpose of studying the viability of the use of a methodology based on genetic algorithms and local search on the resolution of Home Care scheduling problem; as well as the implementation of a system that solves the practical issue. Home Care is used in English as home-based care, and the main motivation to the use of Home Care is the reduction of the bottleneck faced in world´s big cities hospitals. This type of care allows certain patients to be treated at the comfort of home, along with their family and releasing a hospital bed. In contrast, the increasing use of this system has caused in developed countries problems in the assignment of human resources who are available to perform the visits at home. In this context, the project aims to apply artificial intelligence techniques to find good solutions to the problem nurses and health agents scheduling. The problem is described as a variant of the Vehicle Routing problem with Time Window. The final system is composed by a Web module for agents and patients management, an algorithm module responsible for the routes calculations and a mobile module ported by the agent during the visits.

**Key-words**: Home Care, scheduling, genetic algorithms, local search, Android, Vehicle Routing problem with Time Window

# RIASSUNTO

Il progetto ha lo scopo di studiare la fattibilità dell'utilizzo di una metodologia basata su algoritmi genetici e la ricerca locale in merito alla risoluzione del problema di scheduling Home Care, così come l'attuazione del sistema che risolve il problema pratico. Home Care è un termo usato in inglese per discrivere l'assistenza domiciliare, e la motivazione principale per l'uso di Home Care è la riduzione del collo di bottiglia affrontato negli ospedalidelle grandi città del mondo. Questo tipo di trattamenti permette ai pazienti di essere trattati alla comodità della casa, insieme a loro famiglie e liberando un letto d'ospedale. Al contrario, aumentando l'utilizzo di questo sistema ha causato nei paesi sviluppati problemi di assegnazione di risorse umane che sono disponibile ad effettuare le visite a casa. In questo contesto, si propone gli obiettivi del progetto di applicare tecniche di intelligenza artificiale per trovare buone soluzioni per i problemi di scheduling diagenti sanitari e infermieri. Il problema è descritto come una variante del problema di Routing Vehicule con finestra temporale. Il sistema finale è costituito da un modulo Web per i pazienti e gli agenti di gestione, un moduloAlgoritmo responsabile per i calcoli e un modulo Mobile portati dall'agente durante le visite.

**Parole chiave**: Home Care, scheduling, algoritimi genetici, ricerca locale, Android, problem di Vehicle Routing con finestra temporale

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABREVIATIONS

**GA** Genetic Algorithms

**LS** Local Search

**TSP** Traveling Salesman Problem

**VRP** Vehicle Routing Problem

**VRPTW** Vehicle Routing Problem with Time Window

**PMX** Partially Mapped Crossover

**OX** Ordered Crossover

**TSP** Travelling Salesman Problem

**RaR** Remove and Reinsert

# CONTENTS

# PART I

# INTRODUCTION

# 1 Introduction

## 1.1 Generic overview about the Brazilian and International Home Care scenario

Home Care is used in English as home-based care, being a specialization in the health field with a different vision from the average, i.e. instead of the patients going to the hospital to be treated, the health agents go to his residence to treat him.

The use of the Home Care is growing in Brazil, due to its advantages. First of all, the patient is treated outside of the hospital and in contact with his family, resulting in a cost reduction for the hospitals and health insurance, bed release and in an increase in the comfort of the patient, whom can be at his usual comfort.

Resting at home, the patient stays less expose to infections from the hospital environment and, at the comfort of his home, the patient has more autonomy and privacy.

Two big disadvantages of the Home Care are result of the structure, as probes, serotherapy, electronic equipment, and others, that must be kept in the patient residence; and from the management of the patient's needs, that goes from routine activities, such as alimentation, shower, and others, until specific activities, that requires specific professionals, like doctors, physiotherapist, nutritionist, and more. It is clear, according to the second argument, that this is an interdisciplinary work of difficult management.

The treated patient profiles is the ones with stable pathologies, mostlythose with chronic diseases, neurologicaldegenerative diseases and skeletal-lower case diseases, other than the elderly. The Home Care should not be restricted to this profile, and should be faced as an alternative treatment for all profiles, according to the stability of the patient.

According to the IBGE research released on September 2010[1], the life time expectation for Brazilians is 73.1 years. The value reaches 81 years, matching the current value for developed countries such as France and Japan, in 2040. The increase of life expectation demonstrates the potential for Home Caring. Potential that can be observed in developed countries like Denmark, where Home Care receives government support, and estimations from 2008 indicate the investment in the sector of more than 29 billion of Danish crowns (approximately 3.9 billion of euros) (1).

Given this scenario, the use of Home Care will bring positive effects on the health services overcrowd and relief of the future pension scenario. The demand increase of the service results in the need of more professionals, such as generalist doctors and nurse

[1] http://www.ibge.gov.br

technicians trained for home care; and the need for area consolidation, with better studies, and the improvement of the management of the necessary information.

For the reasons previously described, the field of home care is growing in Brazil. The development is concentrated in the private area and, occurred first in an unorganized way, without the government support. However, on January 26th 2006, ANVISA launched the RDC 11 that provides the first polices on its practice in Brazil.

Nowadays, there are many private institutions and associations, like “Portal Home Care[2], that are engaged on the growth of the field. Therefore, the scenario is promising for the development of web and mobile systems that assists the management of information, both patient and health agent.

## 1.2 Understanding of the Home Care problem and motivation

The problem to be treated in this project is the one to calculate and distribute the best scheduling route of visits to patients among the registered health agents. It aims the reduction of agents needed and a better management of the collected information and the routes traced.

On working days, the health agents and nurses collect in a Centre, which can be a hospital, the information of patients to be visited. This information concern the address and time of visit, the treatment that will be given to the patients during the service, i.e. necessary tasks and specialties, among other information that aren’t the focus of this project.

In possession of this information, the agents go out to the streets to treat the patients. On each visit, the agent collects information regarding the patient case using standard reports to facilitate the data consolidation by the management. By the end of the data collection, the agent returns to the starting point and gives the reports.

The data is consolidated and used to realize the remote and centralized monitoring of patients by doctors and specialized professionals. In case any anomaly is detected, the patient should be given a differentiated treatment or the tasks to him associated should be reevaluated.

Normally, during the initial phase, it is elected by the family an agent, whom will assess the patient from this family on his daily activities, such as alimentation, shower, transportation, toilet, and others. Each patient has different necessities, which may require more training, like insulin application, or less, like help the sick on his asepsis. Therefore,

[2] http://portalhomecare.com.br/

the registered health agents and nurses should have differentiated trainings to best assist the associated patients.

Every registered patient inform the best time for the visits. These times are represented as time windows during the day, that is, the patient has a proposed time for the initial of the treatment and a limit time for the treatment to be finished. Moreover, the tasks linked to the patient are estimated timeframes that represent the quantity of time needed to the realization of the task.

The scheduling of the visit should then take into consideration the suggested time windows, the service time associated, the dislocation time between visits, the waiting time to the start of the service (in case the agent arrives to the patient residence before the beginning of the window), the different specialties of the agents and the different necessities of the patients.

It is noticeable for a reasonable number of patients (more than 20) and agents; the manual realization of this task becomes hard, and could result in schedules and routes that do not attend the demands of the patients, other than incurring on the increase of expenses on the management of the service.

Associated to this problem, big challenges are faced, which easily become topic of researches, as an example, to find the best scheduling and routing solution for multiple agents with different specialties to targets that have determined time for treatment and specialized needs.

In order to understand the business model and context,it was used the work of (1), the homecare portal[2] and talks realized to involved professionals on this service.

Themotivation was to gather computation resources and knowledge acquired during the academic path and put them into practice with intuit of providing an alternative solution that covers the mentioned points of the problem.

## 1.3 Objective and scope of this work

The main objective of this project is to develop a system, which allows the automatic calculation of the scheduling and routing between health agents and nurses and patients using the genetic algorithm, enabling the cost reduction on the allocation of agents and the management in real time of collected information.

The project aimsto develop a system capable of receiving information from the agents and patients and distribute it in an optimal manner between the agents, giving the best possible route for the visit of these patients.

The study has as goal:

- The development of a web sub-system for the management of agent and patient information, and collected information;
- The development of a mobile sub-system that will be used by the agents during the visits to facilitate the collection of information and dispatch in real time to the central;
- The development of an algorithm sub-system that will be responsible for the generation of the scheduling and routing. For that, it was analyzed different combinations among the genetic algorithms.

In the context of this work, some restrictions were specified to the realization of the project. The specification and limitations of the goal are:

- The working journey of the agents starts at 8a.m. and ends at 17p.m.;
- The agents have as starting point to the visits a base, and by the end of the journey, they should return to the base;
- For the calculation of the scheduling, it will be available as many agents as needed, in other words, if the algorithm needs more agents, they are available;
- The considered agents are homogeneous, that is they all have only one specialty;
- The agents move by foot during the visits;
- The developed algorithm will not realize load balance, so one agent can be responsible for treating seven patients, while another will treat only tree patients;
- The patients will also be considered as homogeneous and have only one necessity, that is all patients are the same and have equal service time;
- Every patient is associated to only one service time during the working journey and that can vary from patient to patient;
- The service time of all patients fit inside the time window and are the same;
- There will be not associations between agents and patients, that is, it won't be possible to choose an agent according to the patient preference, and over the different work journeys, the same patient can be treated by different agents;
- There will be not treatment prioritize, so all patients have the same priority, and will be treated on the order that the algorithm suggests.

## 1.4 Methodology used

After the specification of the problem to be solved, the content of this project, the imposed limitations by the problem and the restrictions on the goal of the work, it was determined the methodology to be used for the study and realization of the project.

First of all, it was realized a deep understanding of the problem, in order to make it possible its modeling, fitting it on an existing model that best approaches the problem in hand. After that, it was done a technical research already used to solve the problem. Raised all the options, it was observed the most indicated techniques, combined to the technical knowledge previously possessed, and it was chosen the solution considered most adequate to the resolution of the problem.

To the execution of the project, this was divided into modules, each one related to an aspect to be treated on the solution of the problem: the algorithm to the optimization of patient distribution and the systems to be developed in order to make its use possible by the users involved. The modules were: the development of the routing algorithm of patients to the health agents, the development of the web sub-system to the management of scheduling activities of the Home Care and the development of the mobile sub-system to assist the health agents on the routine of patient treatment.

The development of the practical modules followed a methodology of structured development in studies of usage scenarios, specification of functional and non-functional requirements, implementation (using the Model-View-Controller – MVC – and relational database) and integration among sub-systems.

During and after the development of the algorithms, it was realized many performance tests. For those tests, it was used different parameters and techniques in relation to the possible aspects of the genetic algorithms (like crossover and mutation), and, from these tests, the definitive solution was finally chosen.

The tests of the modules Web and Mobile were done independently (functionality tests, based on the usage scenarios) and integrated (integration tests, consistence of data and consistence of interface).

## 1.5 Execution of the project

The project was divided into tree sub-systems: web, mobile and algorithm.

To better develop the main system, the subject was divided into two big phases: the first extensive called the static part of the problem, that is, all the information of patients,

nurses and health agents are known before the execution of the algorithm; and the second regarding the dynamic part of the problem, that is, the treatment of information during the execution of the dynamic algorithm, having as base the proposed solution by the static algorithm.

## 1.6 Structure of the thesis

The project is divided in the following chapters: Introduction, Routing Vehicle Problem with Time Window, Genetic Algorithm and Local Search, Developed System and Final Considerations.

The introduction has the objective to locate the reader on the scope of the project, clarifying the context of the project, the problem, the aim of the work and the motivation involved.

On Vehicle Routing Problem with Time Window the mathematic problems were described and served as a basis for the problem to be modeled and computationally developed.

On Genetic Algorithms and Local Search were elucidated the techniques and approaches to the drawn model be treated and the approximate solutions could be found, other than studies and results obtained.

The chapter of Developed System brings the product specification result of this project, tests and results, and has as objective understanding of the reader on how the system were implemented.

On the chapter of Final Considerations, the conclusions are presented and the space for future works is opened.

# PART II

# STUDY

# 2 Vehicle Routing Problem with Time Window

## 2.1 Traveling Salesman Problem – TSP

The Vehicle Routing Problem with Time Window is a derivation of the Traveling Salesman Problem (TSP) (2). The TSP is a problem considered NP-hard vey studied in Combinatorial Optimization in the area of Operational Research and Computer Science.

The TSP was initially defined by W. R. Hamilton and Thomas Kirkman, in the XIX century. Basically, the idea derivate from the fact that each salesman had to find the shortest route, allowing him to visit all the residences he was responsible for. The problem can be solved through attempts and comparison, however, the value of possible permutations increase in a factorial way in the number of nodes analyzed. This fact makes the system outdated for a big number of notes.

The Travelling Salesman Problem can be described for a graph$G = (V, A)$, composed by a set of vertices $V$, of size$n$, and edges $A$, of size m. Each vertex $v_i \in V, i = 1..n$, represents one house to be visited, and each edge $a_j = (v_i, v_k) \in A, j = 1 \dots m, i = 1 \dots n, k = 1 \dots n \; and \; i \neq k$, represents the path that should be done to go from house $v_i$ to house $v_k$. In case the graph is completed$m = n(n-1)/2$.

Associated to the edges can exist weights $w_j$ that symbolizes the cost of trace the edge $a_j \in A$. In that case, the graph can be asymmetric or symmetric. In the case of the symmetric graph, the weight $w_j$ associated to the edge $a_j = (v_i, v_k)$ is unique, not regardless of the direction taken. In the case of the asymmetric graph, the weight $w_{ik}$combined to the edge $a_{ik} = (v_i, v_k)$ can be different from the weight $w_{ki}$ associated to the edge $a_{ki} = (v_k, v_i), k \neq i$.

Also in relation to the weights, the problem can be metric or not. In the case of metric problem, the weights used will respect the triangular dissemblance, where $w_{ij} \leq w_{ik} + w_{kj}$, and so all weights are restricted to follow that formula. The in the non-metric problem, however, the weights can have variable values, according to how the problem was modeled. An example of TSP can be seen in Figure 2.1.

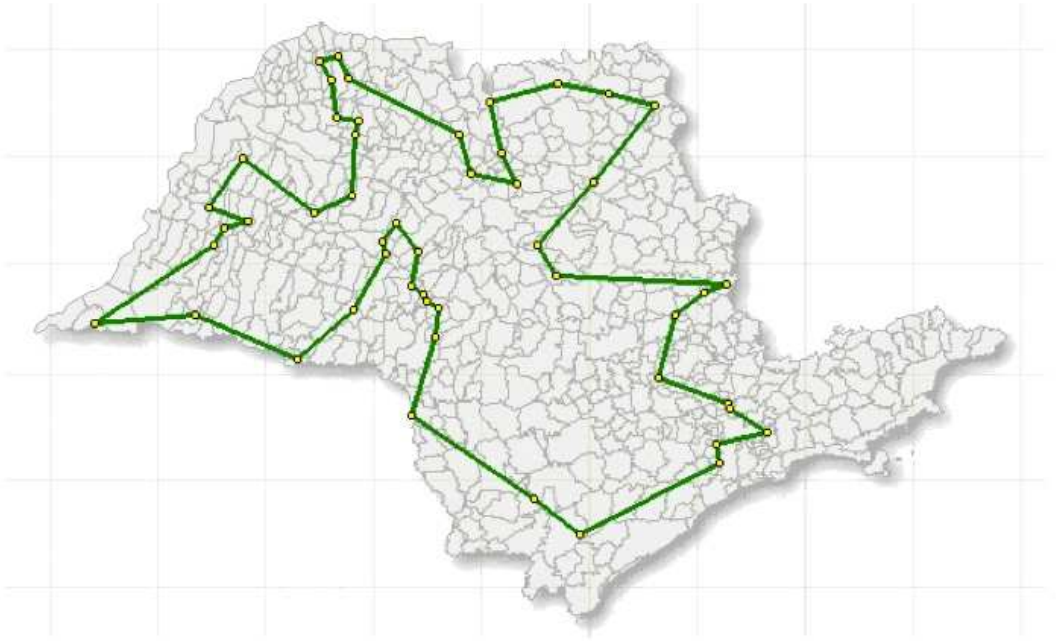

**Figure 2.1 –** São Paulo: Illustration of a possible route for the Traveling Salesman Problem

## 2.2 Vehicle Routing Problem – VRP

The Vehicle Routing Problem (3) regards the problems that involve the delivery or collection of goods. The delivery and collection of goods is related to the service provided by vehicles, in a determined period of time, to a series of consumers or deposits. In order to do so, the vehicle uses a transportation infra-structure that is sometimes modeled as a graph. The problem described on Chapter 1 will be modeled as a variant of the VRP, the Vehicle Routing Problem with Time Window, being part of a group of problems called Crew Scheduling Problems (1).

The graph $G = (V, A)$ used to do the modeling, as an extension of the Travelling Salesman Problem, is composed by edges $a_{ij} \in A$ which represent the streets that can be used to link the consumers $v_i$ and $v_j \in V$, that are represented by the vertices of the graph. Each edge $a_{ij}$ is associated to a cost $w_{ij}$to be used on the transport between the deposits $v_i$ and $v_j$. The following graph illustrates an example of the Vehicle Routing Problem modeling, where there is a central base from where the vehicles start the deposits visits and return after the end of it. The graph model, given a study of the business model treated, can be extended to real applications, such as garbage collection and ambulance routing. An example can be seen on Figure 2.2.

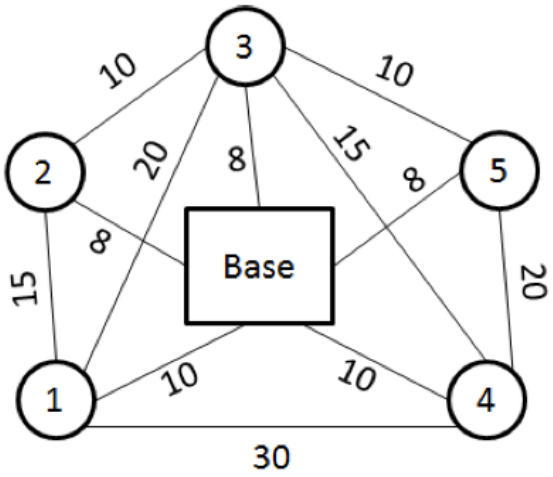


**Figure 2.2 –** Illustration of a graph model to VRP example

The key concepts of the problem are: the graph model, the infra-structure of the transportation network; the group of deposit, consumers that will be visited; the vehicles, drivers that will be used to do the visits; and all the restrictions imposed by the previous concepts, like the consumers should be served in pre-determined time of the day, or there is a minimum and maximum number of vehicles to be used. The role of the analyst behind the modeling of the problem is to attend all the restrictions imposed, relaxing it when needed, so that the solutions to be studied to this model can be approximated with a small margin of error, pre-determined and studied, close to the reality.

Given the characteristic of the variety of problems that can be modeled with VRP, studied in (3), normally the restrictions faced concern: the base to be used as a starting point, that can be unique or not; the capacity of the vehicles, how much load can they support; the heterogeneity of the fleet of vehicle, i.e. they can transport different goods, if so, it should be clear which vehicles can transport what; the cost involved in using the vehicle; the type of load existing; the costs involved on using a certain path, such as the travelling time and the wear of the vehicle; service time, imposed and also the minimum and maximum time of visit.

Besides the restrictions, the problems have goals that should be clear and well defined. The objective normally involves: minimization of costs; minimization of the number of vehicles used; maximization of the number of consumers visited; minimization of travelling time; minimization of waiting time; minimization of the penalties involved in the relaxation of the restrictions; balancing the traced routes; among others, direct related to the characteristics of the studies problem.

The relaxation of the restrictions refers to the fact of allowing restrictions not to be followed, in order to solutions to be traced. The non-fulfillment of these restrictions is

related to a penalty that is added to the quality of the solution, i.e. it would be ideal that penalties were avoided, however, the restrictions imposed can make the search for solutions impossible.

The aim of the minimization of the visits waiting time to the deposits is directly related to the model of the VRP used to the studied problem in the project. Each deposit is associated to a time window, regarding the period of the day that can be used for the visit of this consumer. Therefore, in case the vehicle arrives to the local before the opening of the time window, this difference of time will be added to the total waiting time of the traced route to the vehicle. The smaller the waiting time, better the quality of the solution.

## 2.3 Vehicle Routing Problem with Time Widow – VRPTW

As mentioned before, the VRPTW (3) is a variant of the Vehicle Routing Problem, in which the variable time is of great importance in the analyses of the traced solutions. The study of VRPTW will be used for the modeling of the problem in this project due to the imposed characteristics of the business model, in which each patient will be treated as a vertex of the graph analyzed and each edge will be the path used to connect the two patients. The vehicles relate to the health agents and nurses, and the initial base point, unique in this analyses, will be the health unity from where the agents leaves at the beginning of the day and return after the work journey.

Each patient is associated to a visit service. The service has a minimum time for the visit to start and a maximum for the visit to be ended; this timeline refers to the time window analyzed in the VRPTW. Other than that, each service will have a service time pre-determined regarding the time needed to the visit to finish. The restrictions imposed immediately are that the service time should be smaller or equal to the time window, the difference in schedules, and that the initial time of the agent's visit to the patient should respect the service time and maximum time of the visit.

The agents, on the other hand, have an initial time to the daily work routine and time to finish. The initial time corresponds to the time that they will depart from the initial base to visit the deposits and the final time corresponds to the time when they return to the base. Other than that, the number of agents used will be directly connected to the cost of the solution; better solution will be the n that uses the least number of agents. Moreover, the route traced for an agents, whom corresponds to a series of visits, has a total travelling

time and a total waiting time, both should be minimized so the better solutions can be found.

The model treated regards a VRP, in which the vehicles, the agents, do not have capability of transportation and should not carry loads among the nodes. To the model, it is the agent's responsibility only the fulfillment of the restrictions imposed by the time window and service time. The mathematic model used as base to the analyses of the problem and development of the algorithms will be the classic model proposed by (4), described next.

**Kohl's Mathematic Model**

Formulation – VRP

$$\boldsymbol{G = (V, A)} \text{ graph } \textbf{(2.1)}$$

$$\boldsymbol{V = C \cup \{v_0, v_{n+1}\}} \text{Nodes including departure and end } \textbf{(2.2)}$$

$$\boldsymbol{A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}} \text{arches, passages } \textbf{(2.3)}$$

Costs and Time

$$\boldsymbol{A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}} \textbf{ (2.4)}$$

Transaction Cost: $c_{ij}$

Transaction Time: $t_{ij}$

Service Time: $s_i$

Service Window: $|e_i, l_i|$

Consider the following variables:

$$\boldsymbol{X_{ij}^k = \begin{matrix} 1, \textbf{if the vehicle } k \textbf{ travels the arch } (i,j) \\ 0, \textbf{otherwise}; \end{matrix}} \ \forall \boldsymbol{k \in K, \forall (i,j) \in A} \textbf{ (2.5)}$$

$$\boldsymbol{S_i^k = \textbf{time for the vehicle } k \textbf{ to start serving the client } i \ \forall k \in K, \forall i \in C \cup \{v_{n+1}\}} \textbf{ (2.6)}$$

The following mathematic model for the VRPTW was proposed by (4): objective

1.

$$\boldsymbol{min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} X_{ij}^k} \textbf{ (2.7)}$$

Restrictions

2.

$$\boldsymbol{\sum_{k \in K} \sum_{j \in C \cup \{v_{n+1}\}, j \neq i} X_{ij}^k = 1, \forall i \in C} \textbf{ (2.8)}$$

3.

$$\boldsymbol{\sum_{j \in C \cup \{v_{n+1}\}} X_{0j}^k = 1, \forall k \in K} \textbf{ (2.9)}$$

4.

$$\boldsymbol{\sum_{i \in C \cup \{v_o\}} X_{ih}^k - \sum_{j \in C \cup \{v_{n+1}\}} X_{hj}^k = 0, \forall h \in C, \ \forall k \in K} \textbf{ (2.10)}$$

5.

$$\sum_{i \in C} X_{i,n+1}^{k} = 1, \forall k \in V \textbf{ (2.11)}$$

6.

$$X_{ij}^{k}\left(S_i^k + t_{ij} - S_j^k\right) \leq 0, \forall (i.j) \in A, \forall k \in K \textbf{ (2.12)}$$

7.

$$e_i \leq S_i^k \leq l_i, \forall i \in C \cup \{v_{n+1}\}, \forall k \in K \textbf{ (2.13)}$$

8.

$$X_{ij}^{k} \in \{0, 1\}, \forall (i, j) \in A, \forall k \in K \textbf{ (2.14)}$$

The function objective (2.7) searches to minimize the transition costs for the traced routes. The restriction (2.8) is related to the fact that one patient will be treated only by one agent. The restriction (2.9) imposes that the vehicles should start from the base and the restriction (2.11) imposes that the vehicles should come back to the base. The limitation (2.10) is regards the fact that everything that enters a node should leave the node (conservation). The restriction (2.12) correlates the concept of time window to the viability of realizing the visit to the node j by the vehicle k. The limitation (2.13) imposes that that the initial service time should be inside the time window. Finally, the restriction (2.14) imposes that the analyzed variable is binary.

Distinct approaches for the described problem can be found in literature, for example, Set Partitioning Problem, Upper Bounds or Lower Bounds or Branch and Bound algorithms, which were detailed by (3). On this project will betreated onlymetaheuristics for the approximationsof solutions, as is the case of the proposed approach that involves the genetic algorithms, better described on Chapter 3. To better study and approach the VRPTW, the system to be developed will be treated according to the division of VRPTW in static and dynamic.

### 2.3.1 VRPTW Static

The static VRPTW (3) regards the case which all the needed information to the search of solutions are previously known, before the algorithm execution. The majority of the problems involving vehicle routing analyzed on the literature (3)are static.

### 2.3.2 VRPTW Dynamic

The dynamic VRPTW (5) is related to the problems that, throughout the algorithm execution, the information analyzed is modified, either by addition or removal of

data.Consequently, the routes previously defined should be redesigned in execution time, what can promote worst solutions, but that respect the restrictions involved.

Besides the problem of dealing with new entry, the dynamic routing should deal with the evaluation of the analyzed objects, because the insertion of new data can increase the total of penalties or reduce the quality of the service. The development of the algorithm should consider also the response time to the new entry, from the moment it is being insert to the moment that a new solution is given. A detailed and updated discussion about the problem can be found at (5).

In the problem treated in this project, a group of patients, service and agents is previously known, which is applied the static model, and new data will be added to the problem along the working day, with new service scheduling, cancel of service and agents that will not be able to realize the visit. Services incorporated that cannot be treated due to the restrictions imposed by the problem, will be delayed or will be signalized to be treated outside the scope of the algorithm.

## 2.4 Metaheuristics for VRPTW

Metaheuristics are used as techniques for the approximation of solutions for combinatory problems since the principles of the operational research. During the 70s, with the evolution of the studies on algorithms complexity, many of these problems, among them the Travelling Salesman, were classified as NP-hard. With that, the existing hopes for the search of exact solutions for all instances, gave place for the search of, each time more sophisticated, techniques to solution approximation.

Each problem instance is related to a group of solutions, to which initially is collected the needed information so the search and the classification of solution can be executed.Figure 2.3 illustrates the spectrum of solutions possible for a given problem. It gets clear that the group of solutions will have local maximums that, according on how the search algorithm was developed, could represent points where the search can be ended. The ideal is that the developed algorithm minimizes the difference between the quality of the optimum solution, global maximum and the final solution founded, so it is interesting that the algorithm don’t get locked on the local maximums. It is the developer’s responsibility to study and balance the execution cost, directly related to the execution time, and the quality of the solution wanted.

### 2.4.1 Local Search

Basically, the Local Search (LS) techniques (6) consist of, iteratively, search for better solutions for the proposed problem. Each iteration starts with a solution and a new solution is looked for at the neighbors of the current solution. In case the new solution is better than the current one, a substitution is done and a new iteration is realized until the stop criteria, previously identified, is reached. It is clear, as can be seen on Figure 2.3, that the probability of finding local maximums is high.

Therefore, the LS techniques should be used carefully and, if possible, in cases where the expected quality is known, and so, it can be evaluated, verifying if it satisfies the needed minimum. Differently from a random search, the local search on combinatory optimization problems is based on the business modelapproachedto do the searchand comparesolutions throughobjective functions. Following, will be described some of this metaheuristics that use the concept of the local search as base.

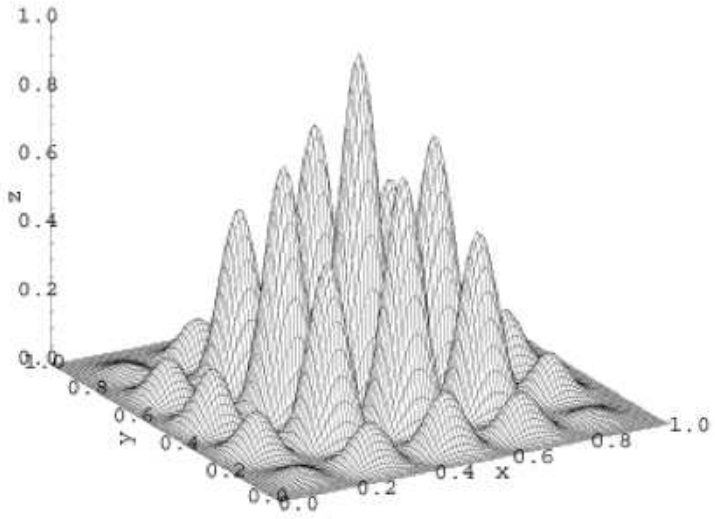


**Figure 2.3 –** Spectrum of solutions illustrating local maximums

### 2.4.2 Simulated Annealing

The Simulated Annealing (SA) technique was described independently by (7) and by (8). Consists on a mechanism inspired on the metal hardening, where it is heated and controlled cooled down.

This algorithm, based on the local search techniques, provides a mechanism to the solutions that are being searched to exit the local maximums with a defined probability. At

each iteration, exists a probability that the search exits the local neighborhood and search for solutions in another neighborhood, chosen randomly. The algorithm initiates with a probability T, 0<T<1, of realize the jump and, at each iteration, the probability is reduced (the metal is cooled down slowly) until it gets to 0.

This probability increases the quality of the solution founded and, as demonstrated at (7), tends to the optimum solution in an infinite time of simulation. The description of the SA is important, because it was a mark on the search for algorithms of approximation of solutions for combinatory problems, raising the interest of the researchers to look for different ways of dealing with these problems. Emerged, then, other methods, for example, the described below.

### 2.4.3 Tabu Search

This approach, named Tabu Search (TS), was proposed in (9) and is based on previous studies from the same author. The basic idea of the TS is to use the technique LS making possible also movements that do not improve the final solution. On the previous case, the movements were performed only when the solution was improved, otherwise, other solution was searched.

The TS manages to realize this search using memories, called tabu lists, that can store the movements already realized so that those searches do not take place again, as pointed out by (10) and (11).

### 2.4.4 Ant colony

Inspired on the observation of the natural behavior of ants, which search for a path between her colony and the food source, the optimization of the ant colony algorithm was initially proposed by Marco Dorigo in 1992 in her doctorate thesis (12). This is a probabilistic technique that has as goal to find good, not necessarily optimum, paths on graphs. This metaheuristics can be used to the search of solutions for the travelling salesman problem, as mentioned in (13) and (14), where the initial base of each salesman corresponds to an ant colony and each city corresponds to a food source. Doing the parallel with the scheduling problem, this will be modeled through a graph where *m* ant colonies – each one for one cooperator – will compete on the search of *n* food sources, one for each task. More information can be found on (15) and (16).

### 2.4.5 Genetic Algorithms

#### 2.4.5.1 The evolutionary approach

The theoretical base of the evolutionary approach was presented to the world by Darwin in a book entitled On the Origin of Species (17). The theory proposes that individuals that have favorable characteristics have bigger chances of survival, given an environment, in comparison to other individuals, what makes them more suitable for reproduction and so to pass on their characteristics to their descendants.

Those characteristics are concentrated on the genetic load of the individuals, in the called chromosomes that are constituted of genes. These genes are responsible for the fact that individuals have certain features different from the others and for transmitting these features to their descendants. Darwin believed that, due to natural evolution, the species evolved gradually as the number of individuals carrying the favorable characteristics increased.

However, how to explain the fact that creatures so different are derived from the same ancestral? During the individual life or during the procreation, the chromosomes can suffer, with a low probability, from mutation. The effect changes the structure of the genes and causes the appearance of new features, favorable or not.

At the 70s, inspired by Darwin's theory, John Holland *et al*. invented the concept of the Genetic Algorithm (18) on the book Adaptation in Natural and Artificial Systems. The idea is based on the evolutionary approach of Darwin, on the fact that a group of solutions can be treated as a group of individuals, on which each solution is represented by a chromosome and is target of natural solution based on the quality of the solution in relation to the others, given an evaluation function.

Therefore, the algorithm has the role of natural selection, acting on a population composed by solutions modeled as chromosomes and realizing the selection to the reproduction of the best solutions. The quality of the solution is measured by an evaluation function that is based on the characteristics of the problem to associate a quantitative factor to the chromosomes target of evaluation.

As mentioned before, the genetic algorithms are used to increase the efficiency on the search for solutions to a given problem. Therefore, it is necessary to realize the study of the problem approached, so that the possible solutions are modeled inchromosomesthat will be used on theexecutionof the algorithm. The better this modeling of the possible solutions, more reliable the solutions founded during the search are.

Each step of the algorithm execution is constituted by operations that simulate the selection, reproduction and mutation of the individuals. The selection operation simulates the basic procedure of the natural selection that is to select the best solutions evaluated by the evaluation function that will be target of the reproduction procedure. This procedure is simulated by the operation of crossover that, as the genetic operation known on biology, is responsible for originate new child solutions through crossing mechanisms of the parents chromosomes structures. After the reproduction, the solutions are target of a mutation operation that, with a defined probability, can alter the structure of the chromosomes in question. To the end of each step of execution, the number of solutions belonging to the population is constant.

The concept of genetic algorithms is of easy assimilation. Its programming, based on a preliminary study of the problem so that the modeling of the possible solutions is successful, is simple and the algorithm execution converges, as the studies of (19), to approximate solutions that solve the approached problem in a satisfactory way. Different from other metaheuristics, such as Simulated Annealing and Tabu Search, which start from an initial solution seeking for other solutions through mechanisms that lead to state transaction, the genetic algorithms start from an initial population of solutions, which decrease the probability of the algorithm to end in an optimal local. However, a special attention should be given to the modeling of the solutions on chromosomes and to the composition of the evaluation function in order to guarantee the convergence of the algorithm.

#### 2.4.5.2 Applications of the genetic algorithm

Nowadays, the applications of the genetic algorithms can be seen in many scientific areas. On the field of the aircraft and automobile design, the use of the algorithms can be found on the analyses of the materials composition and on the analyses of the drawing format of the aircrafts, as pointed out by (20), with the objective of obtain vehicles more light, fast and that use less fuel. On the traditional procedure, several models are designed for tests on air tunnel, causing a high cost of process. However, using the evolutionary techniques, the different model can be simulated and virtually optimized.

In the field of robotics, the genetic algorithms are used, for example, to find combinations of the robot functionality that can be used for the resolution of a certain problem in an optimal manner. Besides that, the evolutionary aspect can result in robots capable of solving many tasks having a generalist role.

Other two areas where the genetic algorithms have high potential for application are the electronic games and the information security. In the field of electronic games, the evolutionary techniques are used for the development of sophisticated games capable of simulating actions and human reactions and so provide a stronger sense of reality when applied against a human player. At this field, the games theory is strongly used on the search of solutions to different virtual problems, from reacting to the steps of the human player until hiding to an ambush. On the information security field, the genetic algorithms are used on the cryptanalysis, i.e. on the attempt to find a clear message from a ciphertext, different techniques were elucidated on (21) and (22).

Furthermore, applications can be found in the research of molecular design (23); on the gene analyses of living beings; financial engineering, on the development ofinvestment strategies and of algorithms that use historical data to preview the tendencies; in marketing and advertisement area; in the field of vehicle routing, well explored on (3) and scheduling; among others that can be found on (24) and (25).

On the routing applications, the genetic algorithms can be seen on applications related to the travelling salesman problem, on which a salesman visits a certain number of cities using the shortest path, and derivatives, such as the vehicle routing problem with time window, target of study on this project. Instead, on the field of scheduling, the genetic algorithms can be seen, for example, on the scheduling of manufacture production, as elucidated by (26). These applications are extended to the industry of fleets routing, commercial agents routing, aircraft scheduling, production scheduling, and others.

## 2.5 Summary

At this chapter, the vehicle routing problem was discussed, and a variable of this problem was covered, the vehicle routing problem with time window.

On the vehicle routing problem with time window, a number of vehicles, one or more, should visit a group of consumers, each consumer should be visited on a specific time of the day and the visit has an estimated duration.

The objective of the problem is to minimize the number of vehicles needed to realize the visit to all consumers, satisfying the imposed conditions.

It was discussed how this problem can be used to model the Home Care scheduling, elucidated on Chapter 1. At this modeling, each patient is considered a consumer; the Basic Unit of Health (USB) or hospital is modeled as the deposit and the agents are represented by the vehicles that will be routed.

Some simplifications were done in order to allow the realization of this project among the time limit, in special, it was not considered the specialties on the agents trainings and there is no memory of the realized visits. These two aspects have the same consequence: the agents have a uniform probability of visiting a specific patient.

At the end of the chapter, it was presented a series of methodologies found in literature for the resolution of the proposed problem. Today, it does not exist an optimum method to the problem of Home Care scheduling , but there is several methods that allow a good approximation of the ideal solution (1).

On the next chapter, it will be discussed the theory of the genetic algorithms and local searches that will be used on this project. These algorithms have enabled obtaining results above average, with sufficient performance to the treated application. It is interesting to point out that on the static case, that will consume the biggest part of the calculation time, the time restrictions are light, the algorithm can run during the night, for example.

# PART III

# DEVELOPMENT

# 3 Algorithm Sub-system

## 3.1 Function requirements of the algorithm

The Algorithm sub-system has the operational purpose only, being used by the Mobi and Web sub-systems, and it has the following functional requirements:

1. To generate routes from the scheduled services information and active agents for the next day of execution;
2. To deal with addition, removal and edition of services during the work journey reorganizing the already generated routes – dynamic part of the algorithm;
3. To provide access layer to the database and entity models for the Web sub-system.

## 3.2 The genetic approach proposed

### 3.2.1 The chromosome

A chromosome is a structure that will store all the information that can suffer changes during the evolution of the beings. Therefore, for each type of problem that has to be solved, there will be a determined group of parameters (genes) to be inserted in a chromosome.

The vehicle routing problem with time window served as base to the composition of the chromosome model for the codification and decoding of viable solutions on the problem resolution. As mentioned before, the study of the chromosome model is of vital importance for the execution of the algorithm, since the quality of the solution is intrinsically related to how it is codified.

To better explain how these solutions will be modeled, the scenario on Figure 3.1 will be used to illustrate the chromosomes. The central square symbolizes the starting and returning point, while the circles represent the points to be visited.

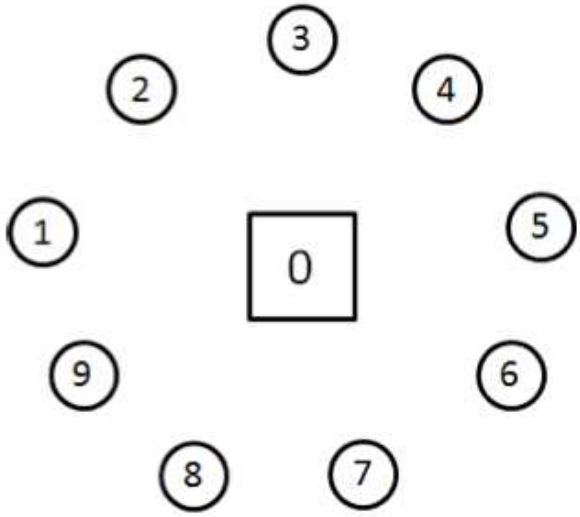


**Figure 3.1 –** Scenario for the solutions study for the routing problem

The scenario can be represented by a characters sequence:

$$\mathbf{0-1-2-3-4-5-6-7-8-9} \quad \textbf{(3.1)}$$

In other words, the scenario has the base node 0 and 9 more nodes that should be visited in an order to be determined. The order determination is related to the variables, already discussed, approached in this problem:

- Initial time of service and final time of service, represented by the time window of the node;
- Service time;
- Travelling time between the nodes;
- Number of agents available to realize the visits.

Therefore, given the service restrictions imposed by the variables, two possible solutions to the problem can be seen on Figure 3.2.

That can be also represented by the characters sequence:

Solution to the left: $0-2-1-9-8-0-3-4-5-6-7$

Solution to the right: $0-3-2-1-9-0-4-5-6-7-8$

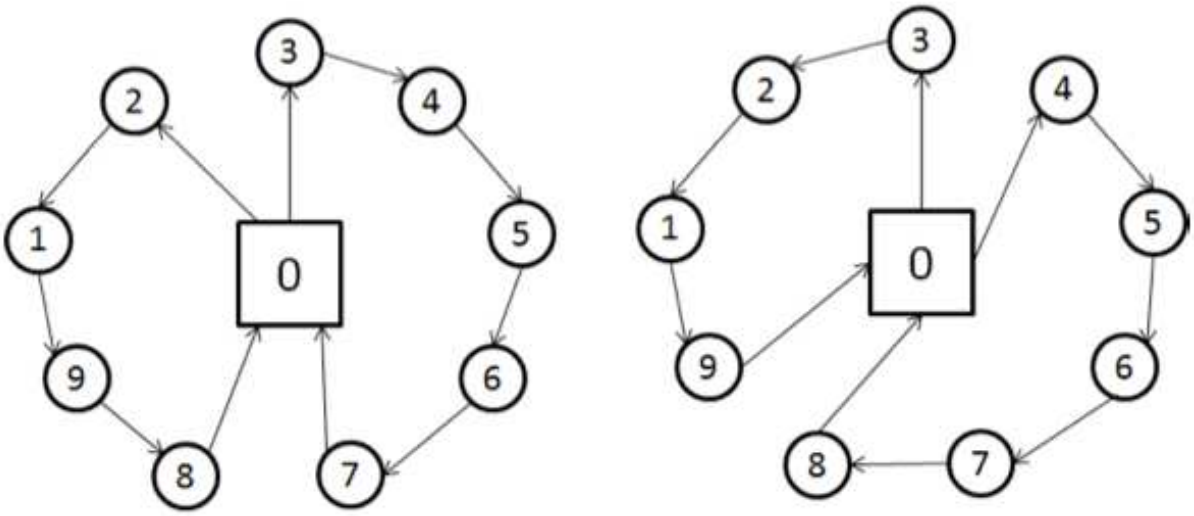


**Figure 3.2 –** Possible solutions for the studied scenario of routing problem

Analyzing the solution to the right, it can be seen that it is composed by to routes, the first composed of four nodes and the second of five nodes. The agents always start from the base, node 0, and return to the same after the end of the visits, and so, the sequences start with 0 and finish with 0 – it is not necessary to represent the 0 to the end and start of the next route, it is enough one representation. At the sequence, can be seen that the first route should start from 0, follow to 2 and then to 1, 9 and 8 and return to 0; and the next route start from 0, pass through 3, 4, 5 and 7 and return, once more, to 0.

The representation of the solutions by a character sequence is simple, instantaneous and elucidates how the visits will be realized. Therefore, this representation will be the model for the chromosomes used on the genetic algorithms studied at this work and the population to be analyzed on the execution of the algorithm will be composed of those sequences. However, there is still the problem on how to maintain only the solutions that respect the imposed restrictions by the variable at the moment of analyzes of the generated chromosomes. For that, the concepts presented in (27) of positive and negative incompatibility can be used to the right structuration of solutions and are described on the next section.

### 3.2.2 Incompatibilities

The concept of incompatibility expresses the viability of the analyzed arches to be traced in possible solutions.

### 3.2.2.1 Negative Incompatibility

The negative incompatibility, shown on Figure 3.3, represents that the arch analyzed can be traced and should be added to a solution at the moment of its modeling. In that case, the subsequence (…)-1-2-(…) can be visualized at the generated solutions.

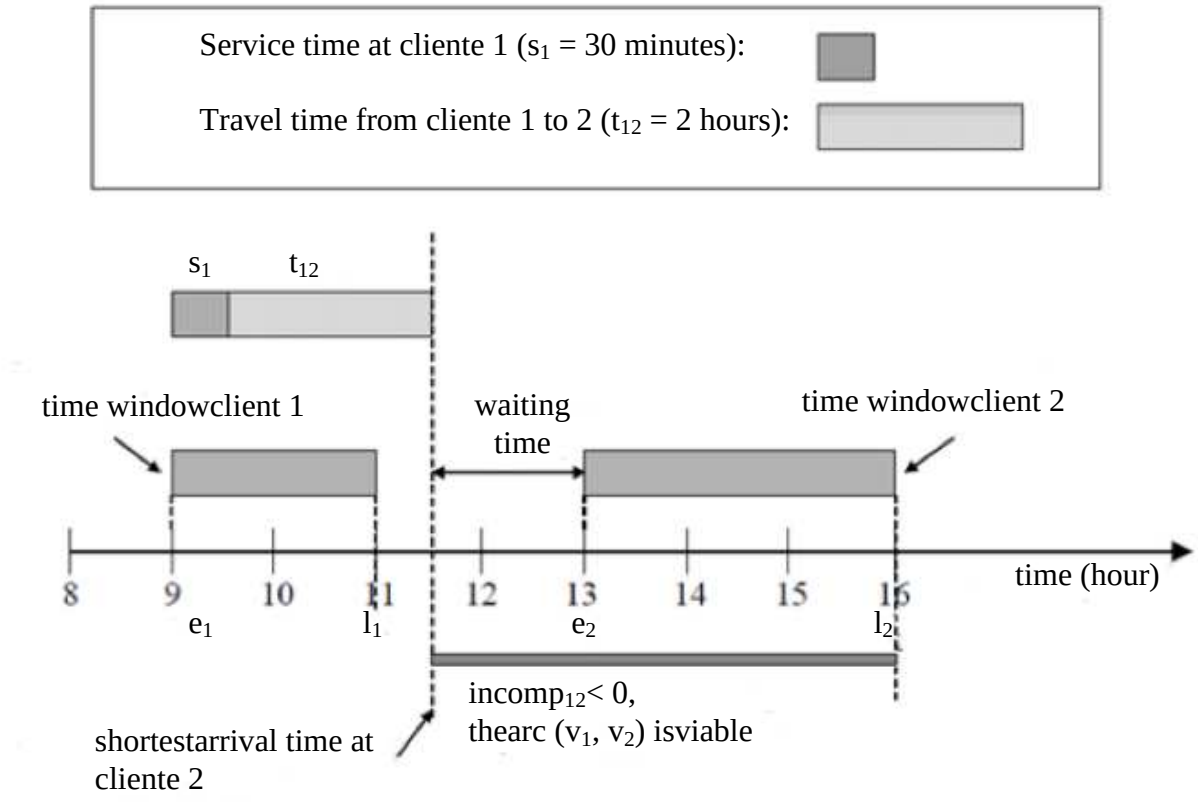


**Figure 3.3 –** Example of negative incompatibility, viable arch.

### 3.2.2.2 Positive Incompatibility

On the other hand, the positive incompatibility, shown on Figure 3.4, represents that the arch analyzed should not be traced, since it is not possible to fulfill the imposed restrictions by the variables. In this case, the subsequence (…)-1-2-(…) cannot be seen at the generated solutions, i.e. generated solutions that contain the subsequence should be eliminated. Therefore, at the moment of analyzes, in case the node in question is the 1 and the next node is 2, this route should be interrupted, i.e. the next node should be 0 and a new route should be initiated, starting from 0 and following to 2, that will be then analyzed.

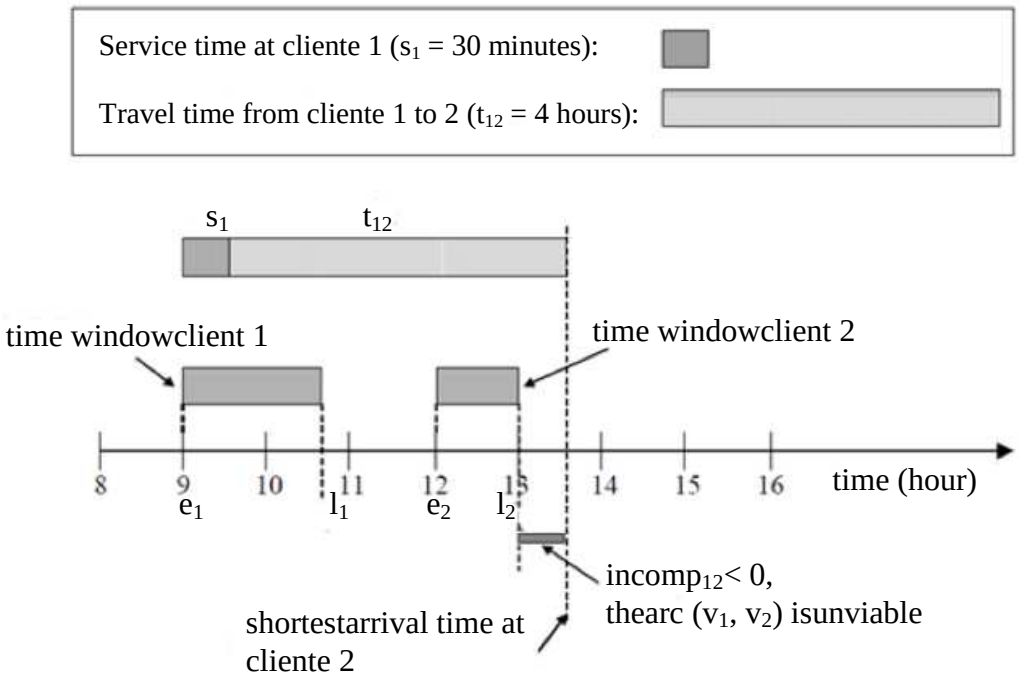


**Figure 3.4 –** Example of positive incompatibility, unviable arch

### 3.2.3 Application of concepts

The flowchart on Figure 3.5 illustrates the application of these concepts, as well as with the use of the variables of the problem, to express the viability of the visit to a node at the moment of the algorithm execution.

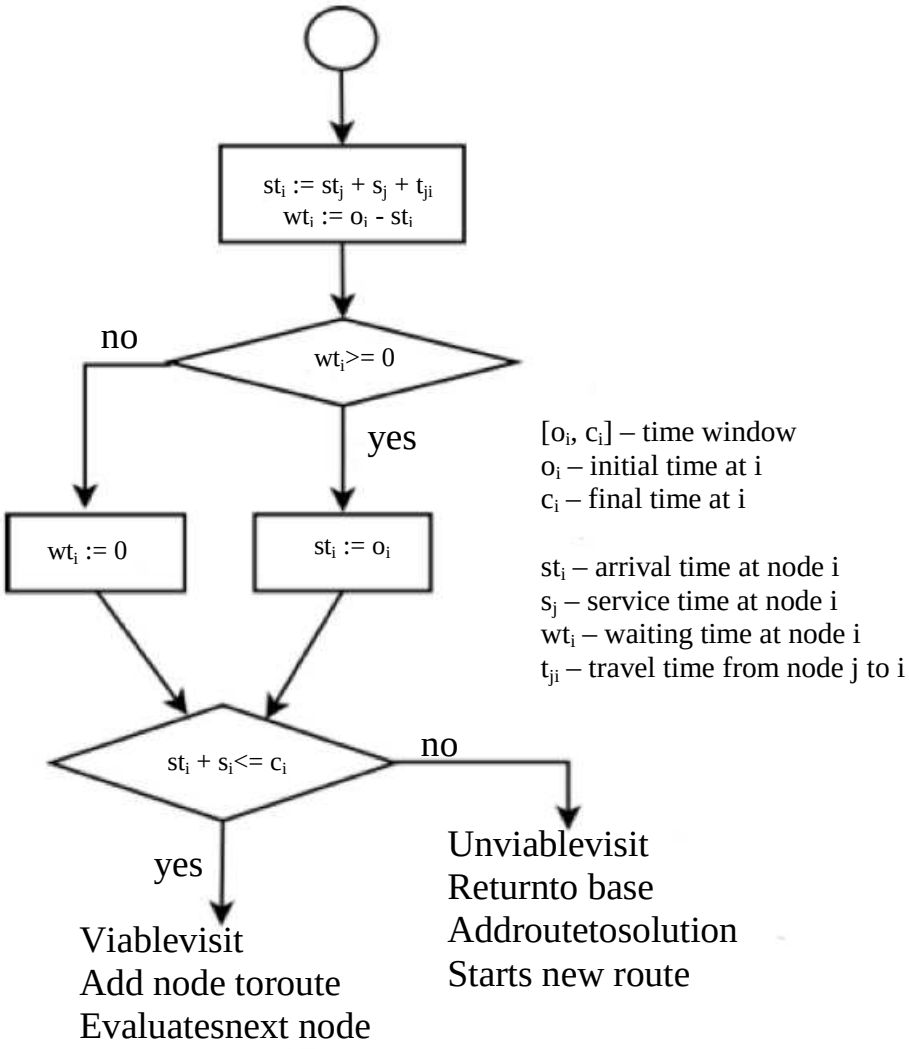


**Figure 3.5 –** Flowchartelucidatingthe viabilityof a node visit

The flowchart on Figure 3.6 illustrates how all the concepts and variable will be treated during the algorithm execution to the modeling of the viable solutions given a chromosome population. The flowchart was proposed so that all the evaluated solutions can be visualized by the algorithm, incurring on penalties on the evaluation function for the solutions that requires more agents, longer travelling time or longer waiting time.

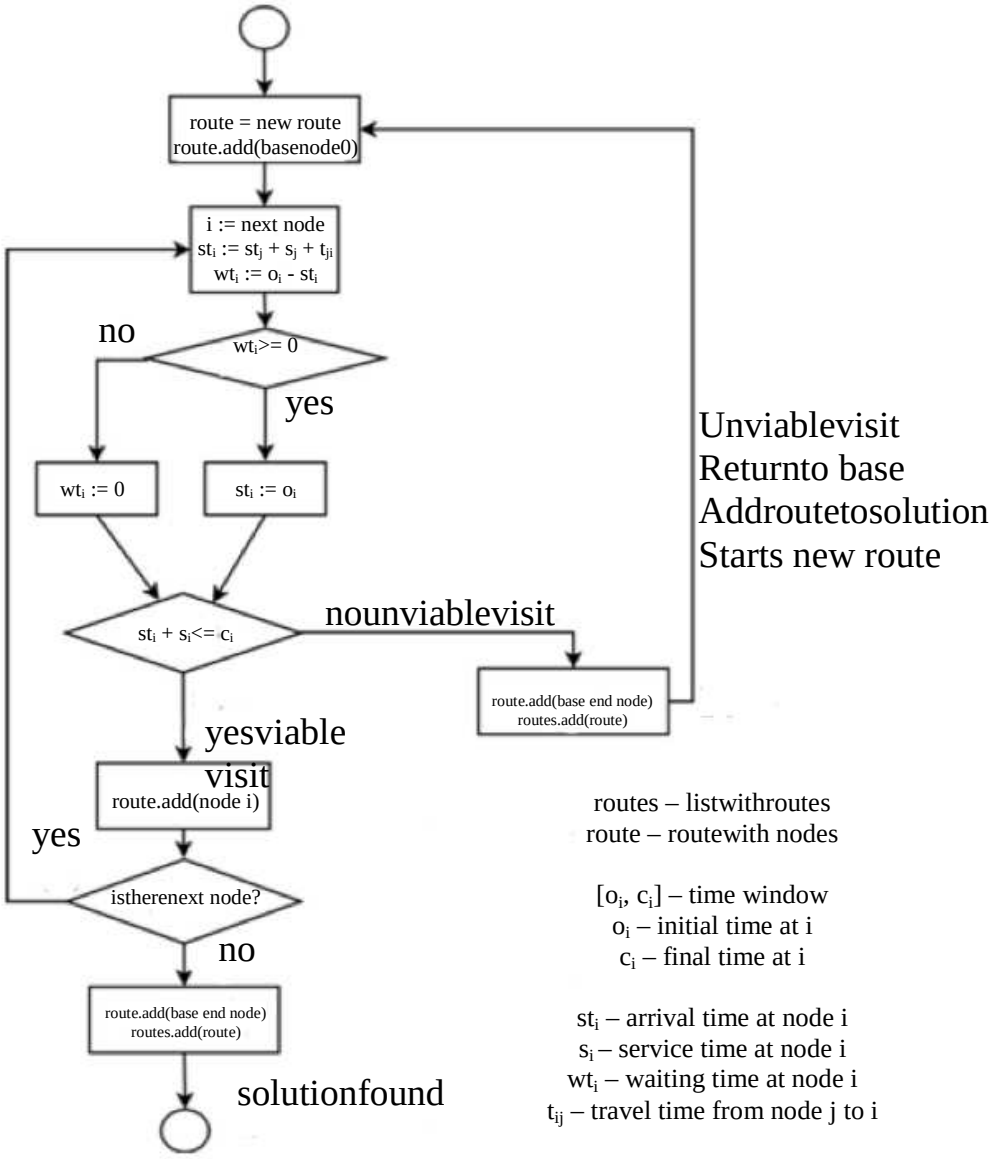


**Figure 3.6 –** Complete flowchart elucidating the solution modeling

The structure presented on Figure 3.7 illustrates the model used on this project for the development of the genetic algorithms, based on the classic model that can be found on the book (6), that will be used as base for the development of the code. At each iteration, i.e. at each step of selection, application of crossover for reproduction and mutation, the population size should stay the same. This size can be maintained selecting initially the population individuals and then applying the operator, but also can be maintained applying initially the operator and then selecting the individuals both from the initial population and the new generation. In both cases, in order to maintain the population size, the solutions should be disposed at each step of the algorithm.

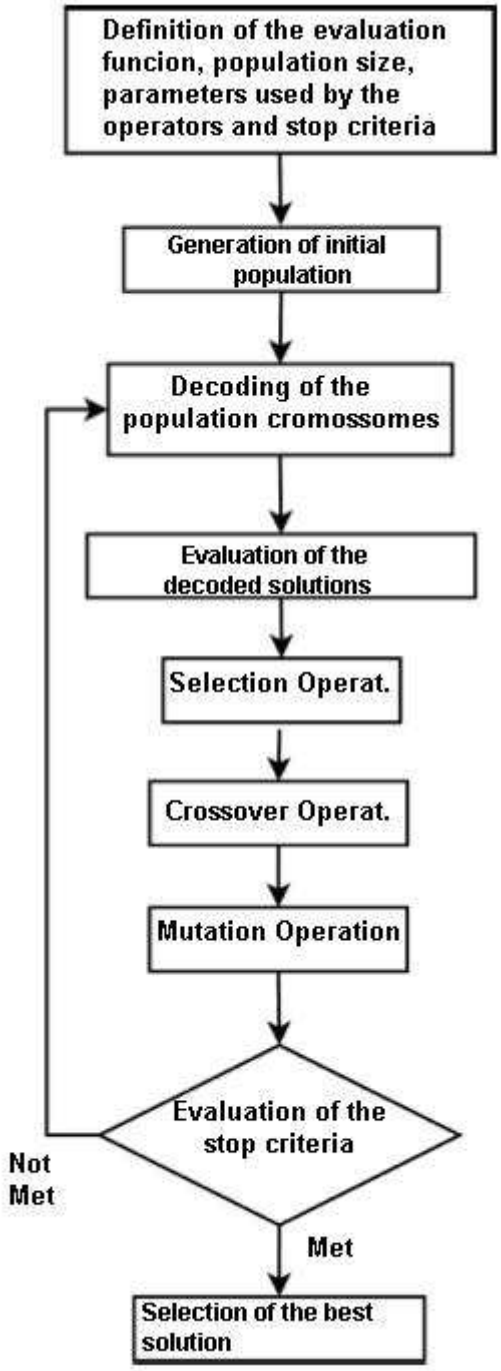


**Figure 3.7 –** Structure of the Genetic Algorithm developed

At the first algorithm execution, the initial population is composed by individuals and sequences of numbers, generated randomly. As the algorithm is executed, the worst solutions generated are substituted by better solutions, always maintaining the population size constant. Following, it will be explained the steps of the model of the genetic algorithms, what will give base to the analyses of the obtained results and of the developed system.

### 3.2.4 Evaluation Function

According to the book (6), the evaluation function is a function $f_i$ that maps solutions *x* to numeric values $f_i(x)$, called fitness values. The aim of the evaluation function is to give quantitative characteristics to the solutions, so that these can be classified in relation to the quality of the obtained result. It is assumed that the bigger the value retuned by the function, the better is the solution.

These values are used by the selection operation, because this is based on the quantitative characteristics to select the best individuals that will be exposed to the other operators. Therefore, the function should be modeled so that represents the analyzed solutions. On the treated case, the variables in question are the agent number or the created routes, the total traveling time per route and the total waiting time per route. These variables are used to the function model. From the problem analyses, it is understood that a low number of agents used, a low average waiting time and a low average traveling time are characteristics of a good solution. Having in mind all the details approached, the function used on the development of the algorithms can be found on functions from (3.2) to (3.5).

$$\boldsymbol{f(S_i) = \alpha * RA_i + \beta * RTE_i + \gamma * RTV_i} \textbf{ (3.2)}$$

$$\boldsymbol{RA_i = (NAPS - NA_i)/NAPS} \textbf{ (3.3)}$$

$$\boldsymbol{RTV_i = (TTVPS - TTV_i)/TTVPS} \textbf{ (3.4)}$$

$$\boldsymbol{RTE_i = (TTEPS - TTE_i)/TTEPS} \textbf{ (3.5)}$$

S = Solution
RA = Agents ratio
RTE = Waiting Time ratio
RTV = Traveling Time ratio
NAPS = Agents on Worst Solution
NA = Agents
TTVPS = Total Traveling Time of the Worst Solution
TTV = Total Traveling Time
TTEPS = Total Waiting Time of the Worst Solution
TTE = Total Waiting Time

It can be observed on the functions showed above, that the values taken for analyses are taken as proportions of unique values, used to all solutions. These unique values are represented by the variables obtained from the worst solutions among all executions until the present moment, i.e. from the solution that presents the bigger number of agents used, that presents the longest traveling time per route and that presents the longest waiting time per route. At each iteration, in case worst values were found, these parameters are updated and the values of the evaluation functions are recalculated. This procedure is adopted so that the obtained value is uniform and can serve as comparison for the analyzed solutions using equal values as base. Moreover, to each variable there is a configuration parameter, represented by α, for the number of agents; β, for the total traveling time per route; and $\boldsymbol{\gamma}$**,** for the total waiting time per route. These parameters are used to alter the relevance of each variable. The bigger the value of the variable parameter, the bigger will be its relevance.

### 3.2.5 Selection

The operation of selection is intrinsically correlated to the evaluation function, since this is used for the analyses of which solutions are most suitable to be used later. At this operation, the factor of selection pressure is the critical point for the maintenance of the population diversity. The bigger the pressure, i.e. the bigger the tendency of selecting the best individuals, the bigger the velocity with which the population will convert and will become homogeneous and bigger the chances of obtaining an optimum local, in this case, the mutation tends to be the only source of diversity. However, the smaller the pressure, slower will be the convergence and there is a risk of executing a blind search, i.e. a search where the solutions are analyzed randomly without any criteria. Nevertheless, a high selection pressure could end with the diversity and the system could converge to a local maximum, or a region of plains, from where it won't be able to exit.

The problem of convergence is related to the velocity with which the solutions get better and to the factorto thehomogenization of thepopulation ofsolutions. The faster the convergence, bigger are the chances that the solutions with the worst evaluation are being disposed, preventing them to pass their characteristics to the next generation. This way, only the best solutions will be maintained on the population, what can cause in stagnation. It is clear that the equilibrium of this pressure is fundamental to the good execution of the algorithm. At this work, it will be approached and analyzed two selection operations: the

Tournament Selection, well described and studied on (28) and on (29); and the Roulette Wheel Selection or Proportional Selection, detailed on (29).

#### 3.2.5.1 Tournament Selection

The tournament selection, as the name shows, simulates a tournament between the solutions, in order to select only the winners. The parameter in focus on this selection is the size of the tournament, i.e. the quantity of solutions that will participate per round. According to (28), the tournament size is directly related to the selection pressure of the operator. The bigger the quantity of competitors, bigger the selection pressure will be. This factor contributed to thepopularity of thistypeof selection, since the pressure can be controlled by one parameter, making it easier torefinethe operator.

The function of the operator is of easy assimilation and implementation. Given a tournament size *k*, it is selected k individuals randomly to compete among them for the chance of proceed on the algorithm. The competition is based on the evaluation valueofthesolutionsof theselected subgroup, and also on the selection probability *p* of the individual that has a better evaluation, and so, the best individual is selected with a probability *p*, the second best with a probability$p * (1 - p)^{k-1}$. The value of p adopted at this project will be always 1, i.e. the best solution will be selected by subgroup.

The solutions can be chosen to the tournament with or without replacement. The choice of without replacement increases the selection pressure, since bigger the chances of intermediary or worse solutions to be eliminated immediately.

In the case of a choice with replacement, a value *k* equal to 2 and a population size *P*, it will be obtained $P/2$ solutions that, after crossover, will result in *P* solutions, maintaining the original size of the population. Another way to maintain the population size constant would be executing $P/2$ tournaments with size *k* and replacement. It is noticeable that if *k* is equal to *P*, the solutions obtained would be the $P/2$ better evaluated (bigger selection pressure) and, the smaller the size of *k*, more disperse (smaller selection pressure) would be the choices.

#### 3.2.5.2 Roulette Wheel Selection or Proportional Selection

As mentioned before, to each solution there is an evaluation value mapped by the evaluation function. The proportional selection, well detailed on (29), will select the solutions based only on the proportions between the evaluation values, and so, the bigger

the evaluation value of a solution, bigger will be the chances of it to be chosen for reproduction. As the name illustrates, the process simulates a roulette wheel, on which the spaces drawn to each solution is proportional to its evaluation value and, the bigger the space occupied at the roulette, bigger the chances of the pointer to end at that area. Figure 3.8 exemplifies the roulette.

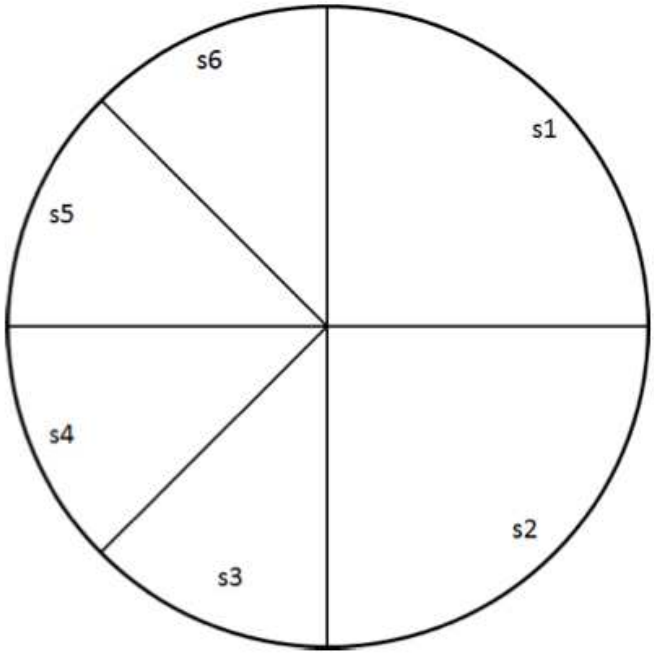


**Figure 3.8 –** Proportional Selection, solutions s1 and s2 add up 50% of the probability of choice

Because the selection is completely correlated to the evaluation value, the chances of lower solutions to be eliminated are high, while solutions with high evaluation values will tend to be always selected. It is noticeable that the selection pressure of this operator is high, resulting on a fast convergence of the solutions.

#### 3.2.5.3 Elitism

The elitism is the process by which the best chromosomes are maintained on the population during the algorithm execution. During the selection process and crossover, the best chromosomes, solutions with bigger evaluation values at that step, can be lost. The role of the elitism is to maintain these chromosomes. Knowing the role of the selection pressure, it is observed that the elitism acts on the increment of this pressure, since maintain, immediately, the best solutions to the next reproduction.

The elitism factor can be combined with the selection operator to balance the selection pressure. This factor is used on the algorithm through the determination of the number of individuals that should be maintained at each step. For example, in case the value to be maintained is 10 individuals, at each step will be maintained 10 individuals better placed, and so, in case a better solution occupies the ninth position of the list of 10: the previous ninth will occupy the tenth position, the tenth will occupy the eleventh and will be eliminated.

### 3.2.6 Crossover

At the genetic algorithms, the crossover operation is used to vary the chromosomes from generation to generation. Being analogous to the biological crossover, more specifically to the anaphase phase, on which, after the duplication of the chromosomes, occurs the separation of the chromatids of each chromosome given origin to the chromosomes child (30).

The principal objective of the crossover is the exchange of characteristics between the two chromosomes, parent's solutions, selected by the selection operation and giving origin to two chromosomes child that have information from both parents. This exchange of information can incur in chromosomes better evaluated by the evaluation function that will tend to be selected on the next phase of the algorithm. However, the opposite can occur, chromosomes child of lower quality that tend to be disposed.

The chromosome model used at this project was explained on section 3.2 The genetic approach, that consists in a sequence of numbers that represent the summary of nodes to be visited. Removing the zeros that are implicit computed by the algorithm to maintain the order and separation of the visits routes, the configuration of the chromosomes can be visualized below.

Chromosome parent 1: 1 2 3 4 5 6 7 8 9

Chromosome parent 2: 9 8 7 6 5 4 3 2 1

Among the crossover operator existing in literature, two were selected for analyses and development during this project: Order Crossover, described on (31) and (32); and partially Mapped Crossover (PMX), described on (32).

#### 3.2.6.1 Order Crossover

The Order Crossover (OX), well described on (31), will be the first operation discussed in the project. The operation selects two cutting points on the parent's chromosomes randomly. Assuming the example below, the indexes selected randomly from the first parent for the cut were 2 and 4, also, starting to count from the first gene with index 0 from the left to the right.

Chromosome parent 1: 1 2 3 4 5 6 7 8 9

Chromosome parent 2: 9 8 7 6 5 4 3 2 1

The selection, in bold, is transferred to the first chromosome child, according to the example below.

Chromosome child 1: _ _ **3 4 5** _ _ _ _

The second chromosome parent is used to fulfill the rest of the genes from the chromosome child 1. Going from the left to the right, in order (hence the name of the operator), the vacancies from the chromosome child 1 are fulfilled and repetitions will be avoided.

Chromosome child 1: 9 8 **3 4 5** 7 6 2 1

To the generation of the chromosome child 2, the cutting operation is repeated to the second parent and the vacancies are fulfilled using the order found on the first parent. Assuming cutting index at 4 and 5, randomly selected, the second chromosome will have the following configuration:

Chromosome child 2: 1 2 6 7 **5 4** 8 9

It is noticed, then, that this operation preserves the order of the visits found on the chromosome parent used to the fulfillment of the vacancies of the chromosome child.

#### 3.2.6.2 Partially Mapped Crossover

The partially mapped crossover, described on (32), can be seen as a permutation crossover that guaranties that the permuted positions among the two selected chromosomes parent are found only once at the chromosomes child generated. Both chromosomes child receive genes from each chromosome parent through a mapping of permutation.

In a similar way to the order crossover, two cutting points will be selected randomly, but this time, the cuts will be done uniformly in both parents. Assuming that the cutting

indexes are 5 and 8, the configuration of the parents for the mapping of the permutation is illustrated below.

Chromosome parent 1: 1 2 3 4 **5 6 7 8** 9

Chromosome parent 2: 9 8 7 6 **5 4 3 2** 1

As can be observed, both parents received a mark for cutting. After the marking, two sub-chains are transferred to the chromosomes child, being the one from chromosome parent 1 transferred to child 2 and vice-versa.

Chromosome child 1: _ _ _ _ **5 4 3 2**_

Chromosome child 2: _ _ _ _ **5 6 7 8**_

The permutation mapping is realized analyzing the two sub-chains marked on the chromosomes child. Therefore:

- The gene 4 will be permutated with the gene 6;
- The gene 3 will be permutated with the gene 7;
- The gene 8 will be permutated with the gene 2.

After the mapping, the chromosomes parents will be analyzed and the permutations will be realized according to the mapping to the chromosomes child respectively, i.e. chromosome parent 1 to chromosome child 1 and parent 2 to child 2. Always that a gene that was mapped is found, it will be substituted. The chromosomes child generated, then, will have the configurations described below, the genes in bold are those that suffered permutation.

Chromosome parent 1: 1 2 3 4 **5 6 7 8** 9

Chromosome parent 2: 9 8 7 6 **5 4 3 2** 1

Chromosome child 1: 1 8 7 6 **5 4 3 2** 9

Chromosome child 2: 9 2 3 4 **5 6 7 8 1**

As well as the order crossover, the described crossover operator maintains the viability of the chromosome, i.e. the non-repetition of visits, through the mapping done after the marking of the chromosome parents. Therefore, if each chromosome parent were represented by a vector, the mapping is concluded after one pass by the sub-chains marked and the chromosomes child are found after the analyses of the genes from both chromosomes parents so that the permutation is realized. At the worst scenario, in which the marking is given in the entire chromosome, the algorithm should go through the vector 3 times: one for the marking and one to each chromosome parent. Representing a linear complexity, $O(n)$.

### 3.2.7 Mutation

The mutation operator, well described and analyzed in (33), is used on the genetic algorithms as a way of adding a factor of randomly variation to each solution belonging to the analyzed population. The objective of the operator is to prevent the loss of diversity, i.e. to avoid the algorithm convergence to a local maximum, or to a plan region.

The traditional implementation of the operator uses a parameter that indicates the probability of a randomly selected gene to be altered. For example, if the parameter is used at 5%, the chromosomes from the solution have a 5% probability that any gene suffers mutation. Assuming that the mutation occurs and that the selected gene of index 7 is substituted by the gene 4, the configurations before and after mutation are described below.

Chromosome before mutation: 1 2 3 4 5 6 7 **8** 9

Chromosome after mutation: 1 2 3 4 5 6 7 **4** 9

The chromosome after mutation has two genes 4, and so, two visits will be done to the node represented by the index 4, violating the restriction of one unique visit to each node. Therefore, the traditional implementation do not respect the imposed conditions by the problem description.

On this project, the mutation operations developed and analyzed that respect the restrictions of uniqueness of the genes, are: the operation of Remove-and-Reinsert (RaR), described in (34); and the Inverse operation described in (33).

#### 3.2.7.1 Removal-and-Reinsert

The operation of Removal-and-Reinsert, described in (34), randomly selects two indexes from the chromosome, removes the gene corresponding to the first index and reinserts it at the corresponding point to the second index, readjusting the position of the remaining genes. The following example illustrates the process of Removal-and-Reinsert, given the drawn indexes 4 and 7.

Chromosome before mutation: 1 2 3 4 **5** 6 7 8 9

Chromosome after mutation: 1 2 3 4 6 7 8 **5** 9

#### 3.2.7.2 Inversion

The inversion operation, described in (33), randomly selects two indexes from the chromosome and, opposite to the Removal-and-Insert, makes the change of genes corresponding to the selected indexes. The following example illustrates the inversion process, given the drawn indexes 4 and 7.

Chromosome before mutation: 1 2 3 4 **5** 6 7 **8** 9

Chromosome after mutation: 1 2 3 4 **8** 6 7 **5** 9

Chromosome after RaR: 1 2 3 4 6 7 8 **5** 9

### 3.2.8 Chromosomes duplications

After the initial generation of the chromosome population and after each step of the algorithm execution, there is the possibility of a chromosome to be repeated on the population. The repetition tends to occur in a bigger scale with the best solutions, since those ended up being selected in bigger proportions. To deal with duplications, the algorithm can be developed as following:

- Accept duplications;
- Limit the number of duplications;
- Do not accept duplication.

  At this project, the algorithms developed won't accept chromosome duplications.

### 3.2.9 Stopping criteria

The stopping criteria adopted on the development of the algorithm is checked at the end of each iteration, and in case it is met, leads to the end of the execution of the algorithm and the selection of the best solution f0ound until that moment. A detailed and extensive analysis of stopping criteria can be found in (35). Among the possible criteria at the end of each iteration are:

- Number of new chromosomes founded;
- Number of new and unique chromosomes founded;
- Number of generations (iterations) done;
- Percentagedifferencebetweengenerationsof evaluation;

- Execution n-parallel of the same algorithm and verification of the error produced between the n samples. The stop will occurs when this error goes to a minimum previously determined.

During these tests and executions of the algorithms developed in this work, the stopping criteria used will be the number of generations realized. Therefore, according to the number of nodes analyzed, it will be defined a limit number of iterations to be executed to verify the conversion of the algorithm.

## 3.3 Local Search Proposed

The local search proposed will be executed after the steps of the genetic algorithm, at each iteration. The idea of using the local search comes from the fact that studies (34)(3) demonstrated that genetic algorithms only are not capable of approximate efficient solutions to the Vehicle Routing Problem.

The local search illustrated on Figure 3.9 was inspired on the $\lambda - Opt$searches, being the most common the λ=2, $2 - Opt$, and λ=3,$3 - Opt$. The algorithm $2 - Opt$, introduced on (36), remove two sections from a route chosen randomly and recombine them in all the possible ways given origin to many solutions, that are compared and the best is chosen. The $3 - Opt$, proposed initially in (37), removes 3 sections from a route and proceed in a similar way as $2 - Opt$.

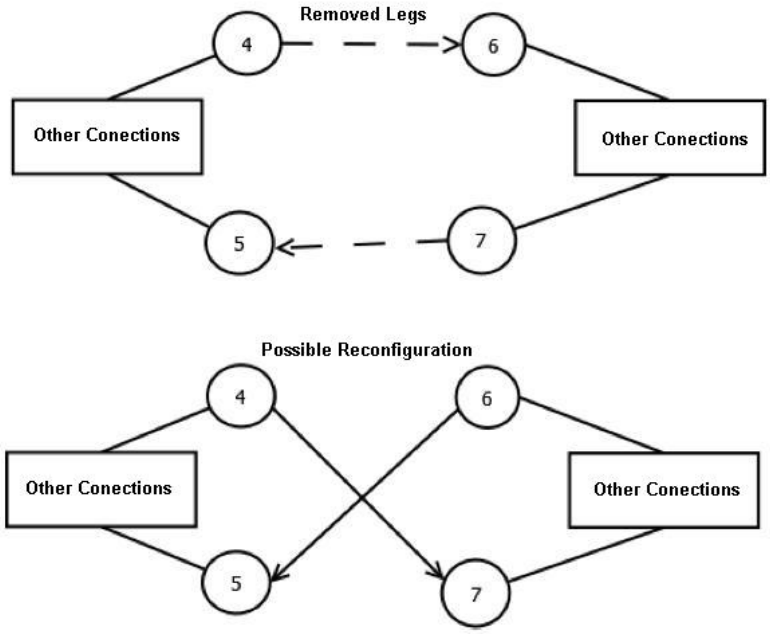


**Figure 3.9 –** Local search 2-Opt scheme

The proposed idea, oriented to the scheduling problem of health agents, consists in, at each iteration step of the genetic algorithm, going through all the solutions founded, dismembering the worst route and trying to reconstruct the services from the dismembered route in the remaining routes from each solution analyzed. This way, the final algorithm would be orientated to reduce the number of agents, removing the worst route means decrease the number of agents used, and to maximize the number of services provided in each remaining route.

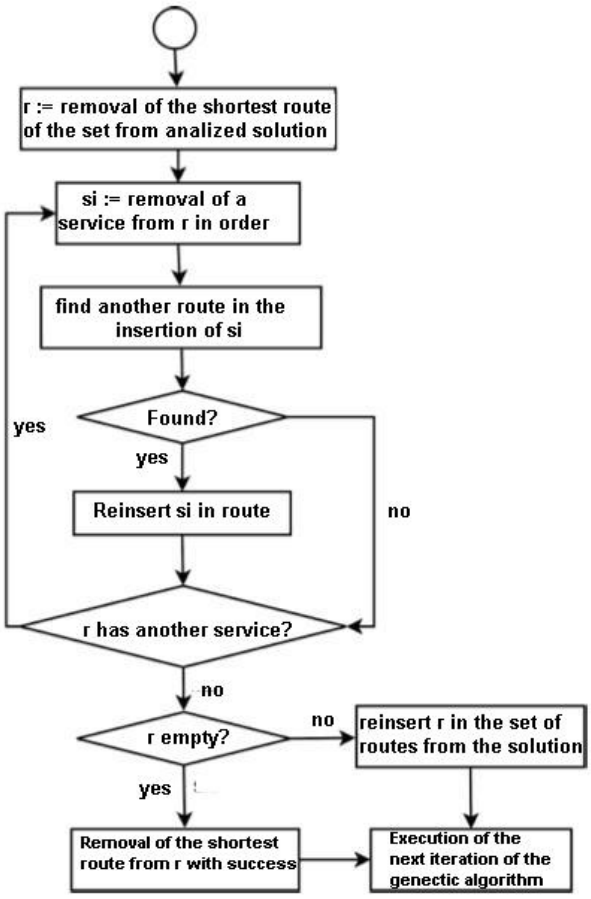


**Figure 3.10 –** Structure of the local search proposed

The results founded using the genetic algorithm proposed and the local search shown on Figure 3.10 are presented in the next section. The results of the use of the local search oriented to the aim of the problem exceeded the expectations, and so, to the algorithm sub-system proposed, it will be used the combination of the genetic algorithm and the local search.

## 3.4 Procedure and Results

### 3.4.1 Computer used for the tests

The results obtained and presented in this section were derivate from the data collected in a machine with the following configuration:

- Processor: Intel ® Pentium ® Dual Core T4300, 2,10GHz, 800MHz FSB, 1MB cache L2;
- Memory: 03GB, DDR2, 667MHz;
- Hard Drive: 250GB, 5400RPM, SATA.

### 3.4.2 Methodology and Data

To the realization of the tests, it was created databases with services allocations that allowed the calculation of the value of the optimum solution, in the categories of number of agents used, average waiting time per agents and average traveling time per agent. The databases created have the following number of nodes: 50, 60, 70, 80, 90, 100, 110, 120.

These databases were created to allow the execution of the possible combinations of the genetic algorithms, in relation to the selection type, crossover, mutation, and others, with the objective of observing the theoretical concepts in practice and of selecting the combinations to be used to the final algorithm sub-system.

The data were obtained through 10 different samples, in which the stopping criteria adopted was the maximum number of iterations pre-determined, and to each group of data, it was analyzed the average and the standard deviation from the samples. This procedure was adopted so that eventually distortions founded in a sample could be redeemed thought the analyses of other samples in the same context.

All the results obtained and presented in this chapter were given using the static configuration of the algorithms, i.e. all information needed was already available in the created database. Moreover, in order to analyze a specific operation, as an example the mutation, the algorithm was configured with other operations, for example crossover and selection that were kept unchangeable during the whole analyses of the different target operations, in the case, mutation.

| Parameter | Configuration |
|---|---|
| Population size | 100 |
| Evaluation Function | All analyzed variables with same weight 1 |
| Selection | Proportional, RouletteWheel |
| Crossover | Crossover OX |
| Mutation | Analyzed parameter |
| Elitism | 10% of population |

**Table 3.1 –** Example of the base table used to the configuration of the genetic algorithm analyzed

In the majority of analyses, the variables analyzed will be the ones discussed in previous chapters:

- Number of agents/routes needed;
- Average travelling time per route;
- Average waiting time per route.

These variables serve as basis to the evaluation function, and so, are of extreme importance to the right configuration of the algorithm.

### 3.4.3 Analyses of the effect of the population size and execution time

Among the context of the possible configurations and the refining of parameters used by the genetic algorithm, the appropriate decision was to analyze first the effect of the population size used by the algorithm, since this value has a direct impact on the final execution time of the algorithm.

In order to analyze the effect of the population size, the team used a genetic algorithm base, without local search, composed of the configuration exposed bellow. Firstly, it was analyzed the execution time of the genetic algorithm with and without local search to identify the execution time of a package of 1000 iterations, base to the rest of the analyses.

The analyses of the execution time was done for a population of 100 chromosomes varying the number of nodes at the database analyzed, always maintaining the number of 1000 iterations. The compilation of the obtained data is found on Figure 3.11 and at Table 3.2 and Table 3.3.

| Parameter | Configuration |
|---|---|
| Population size | Analised parameter |
| Evaluation Function | All analyzed variables with same weight 1 |
| Selection | Proportional, RouletteWheel |
| Crossover | Crossover OX |
| Mutation | Inversion with probability of 5% |
| Elitism | 10% of population |

**Table 3.2 –** Genetic algorithms configuration for the analyses of population size

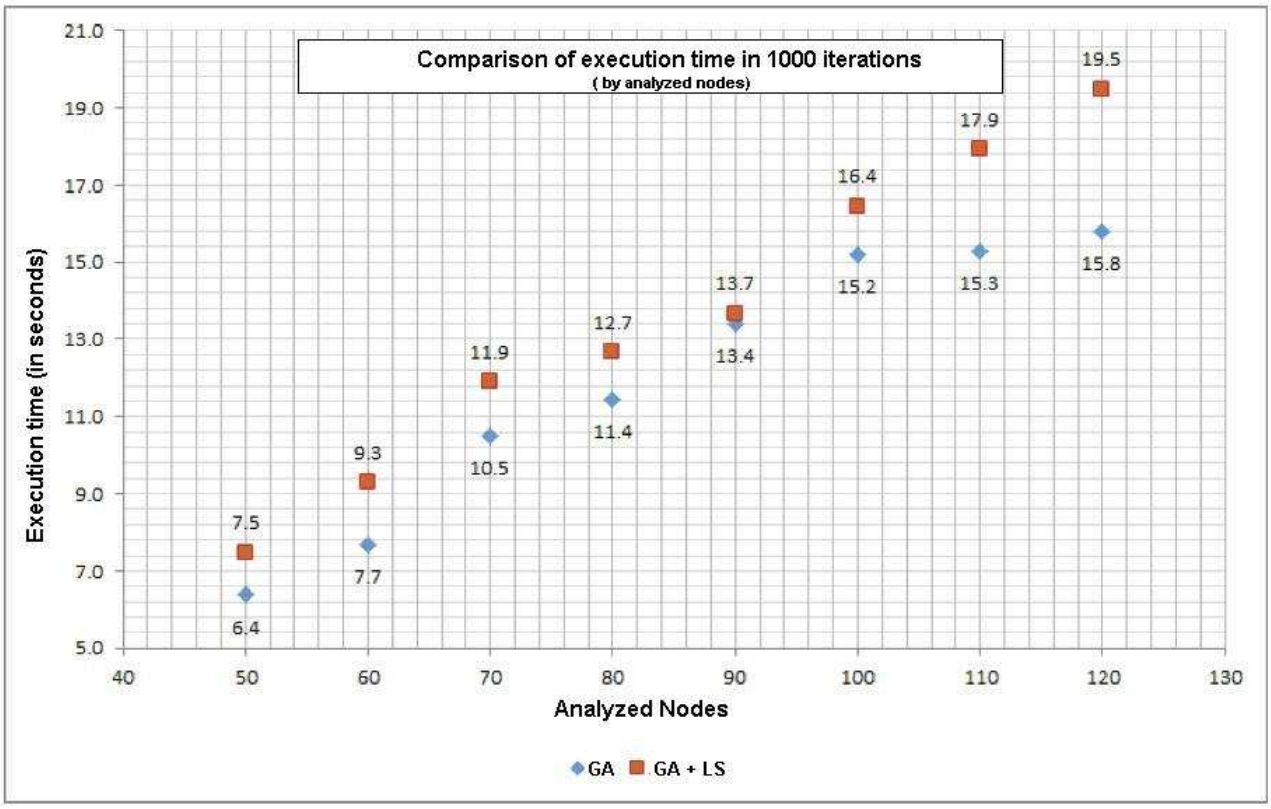


**Figure 3.11 –** Comparison of the execution time between the genetic algorithm with and without local search varying the number of analyzed nodes, 1000 iterations

| Nodes | GA(s) | GA+LS(s) |
|---|---|---|
| 50 | 6.4±0.7 | 7.5±0.3 |
| 60 | 7.7±0.4 | 9.3±0.6 |
| 70 | 10.5±1.3 | 11.9±0.7 |
| 80 | 11.4±1.6 | 12.7±1.1 |
| 90 | 13.4±3.1 | 13.7±0.5 |
| 100 | 15.2±1.8 | 16.4±1.7 |
| 110 | 15,3±1.2 | 17.9±0.8 |
| 120 | 15.8±0.9 | 19.5±1.7 |

**Table 3.3 –** Comparison of the execution time between the genetic algorithm with and without local search varying the number of analyzed nodes, 1000 iterations

It is observed that the execution time varies linearly with the number of nodes analyzed and that the additional time, given the execution of the local search, compensates the quality of the solution founded, as will be demonstrated the superiority of the combination of the genetic algorithm and the local search (shown in the next section).

The analyses at Figure 3.12 and Table 3.4 was realized varying the size of the chromosome population for the same 50 nodes, without the local search proposed.

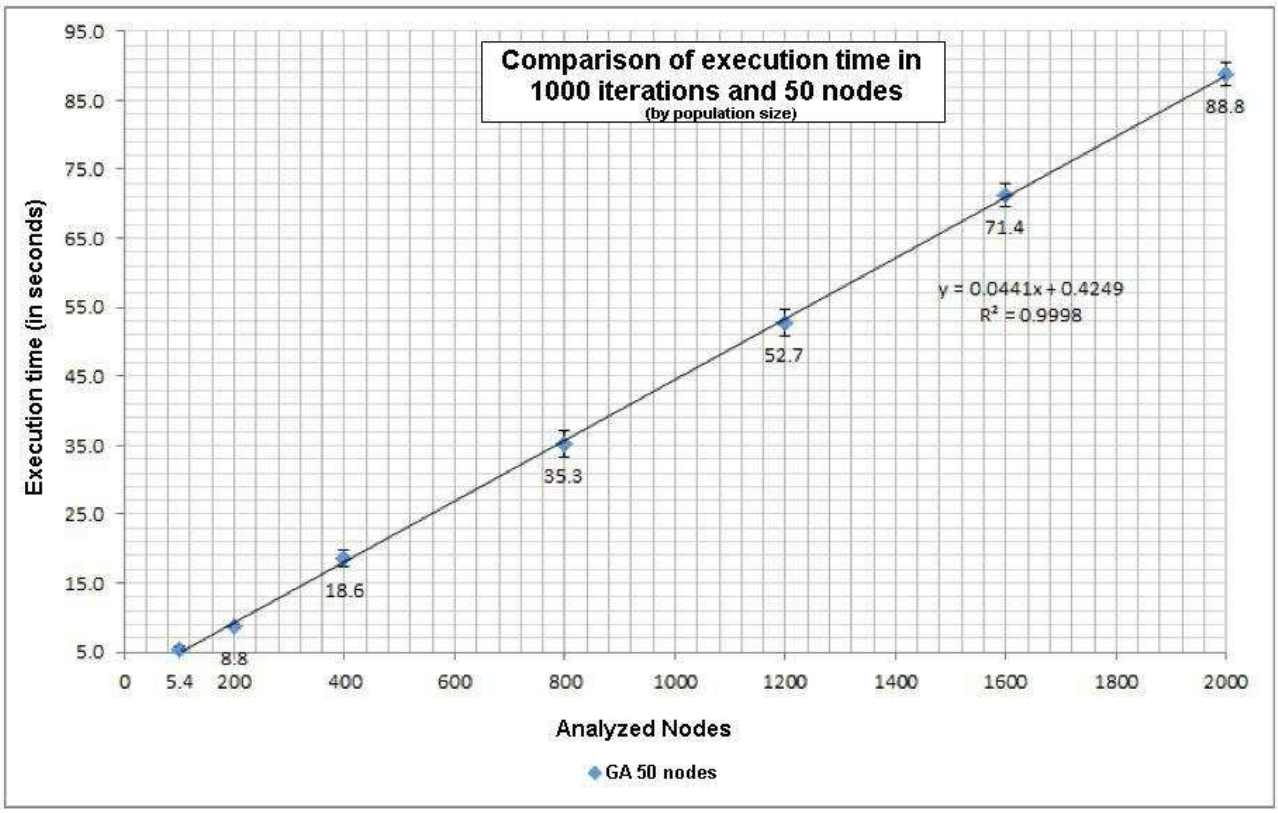


**Figure 3.12–**Comparison of the execution time for different population sizes, base of 50 nodes and 1000 iterations

| Population | Execution time (s) |
|---|---|
| 40 | 2.1±0.2 |
| 100 | 5.4±0.6 |
| 200 | 8.8±0.3 |
| 400 | 18.6±1.1 |
| 800 | 35.3±1.9 |
| 1200 | 52.7±2 |
| 1600 | 71.4±1.7 |
| 2000 | 88.8±1.6 |

**Table 3.4 -** Comparison of the execution time for different population sizes, base of 50 nodes and 1000 iterations

It was adopted the package of 1000 iterations as bases to the subsequent analyses and it was considered as the limit time the value of 40s per package, for the base case of 50 nodes, considering that longer times would make impossible the analyses and tests of the algorithms, since the satisfactory results, as will be shown, will be found at the end of the execution of 50 packages (approximately 30min).

Analyzing the graphic from Figure 3.12 and keeping in mind the limit of 40s per package, the limit size of population adopted will be, in average, 600 chromosomes. Therefore, it was adopted for the next analyses, the sizes of 40, 100, 200 and 400 chromosomes per population.

The first variable to be analyzed is the one that refers to the number of agents needed, i.e. the number or routes created by the algorithm for the working day. The graphic from Figure 3.13 and Table 3.5 illustrate the effect of the population size used by the algorithm during its execution, represented by the number of iterations. The graphics generated below show only the average of the samples; the standard deviation was left aside at this moment to not pollute the data.

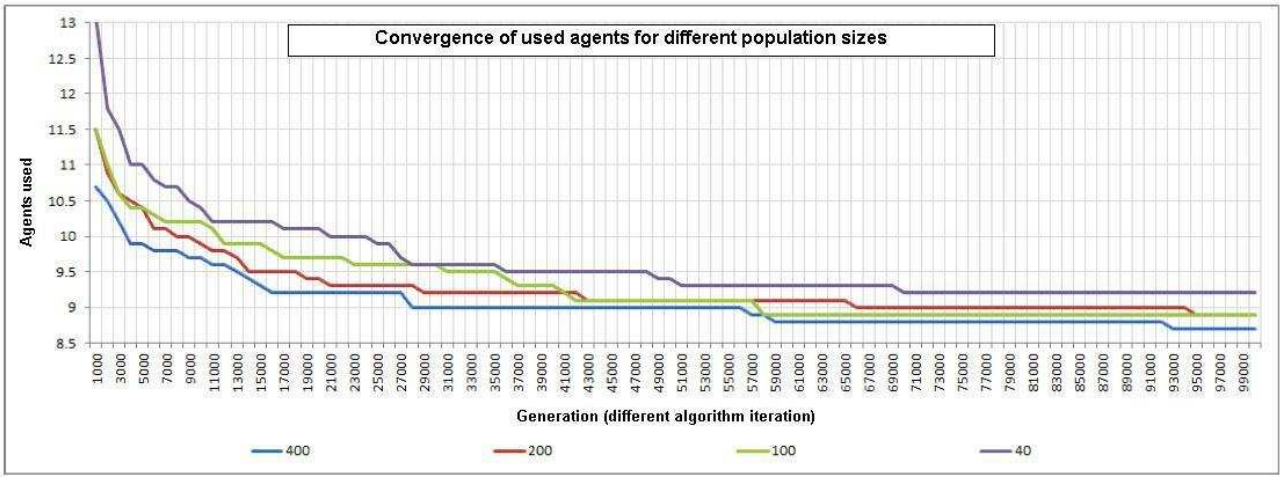


**Figure 3.13 –** Number of agents/routes per iteration varying the population size used

| Population size | Optimum | Agents | Difference |
|---|---|---|---|
| 40 | 8 | 9.2±0.4 | 13.04% |
| 100 | 8 | 8.9±0.3 | 10.11% |
| 200 | 8 | 8.9±0.3 | 10.11% |
| 400 | 8 | 8.7±0.5 | 8.05% |

**Table 3.5 –** Final conversion of agents for different population size

Analyzing the graphic at Figure 3.13, it is possible to notice that the smaller the population size used, bigger the number of routes traced initially, however, this value tends

to reduce with the iterations and stabilizes (theoretically, at infinity, it would tend to stabilize at the value of the optimum solution).

The next variable analyzed in Figure 3.14 and Table 3.6 regards the average waiting time per route to start to realize the service. It is expected that better solutions have smaller waiting time, since the efficiency of the agent associated to the scheduling and traced route would be bigger. The graphic on Figure 3.14 illustrates the effect of the population size used by the algorithm during its execution, represented by the number of iterations.

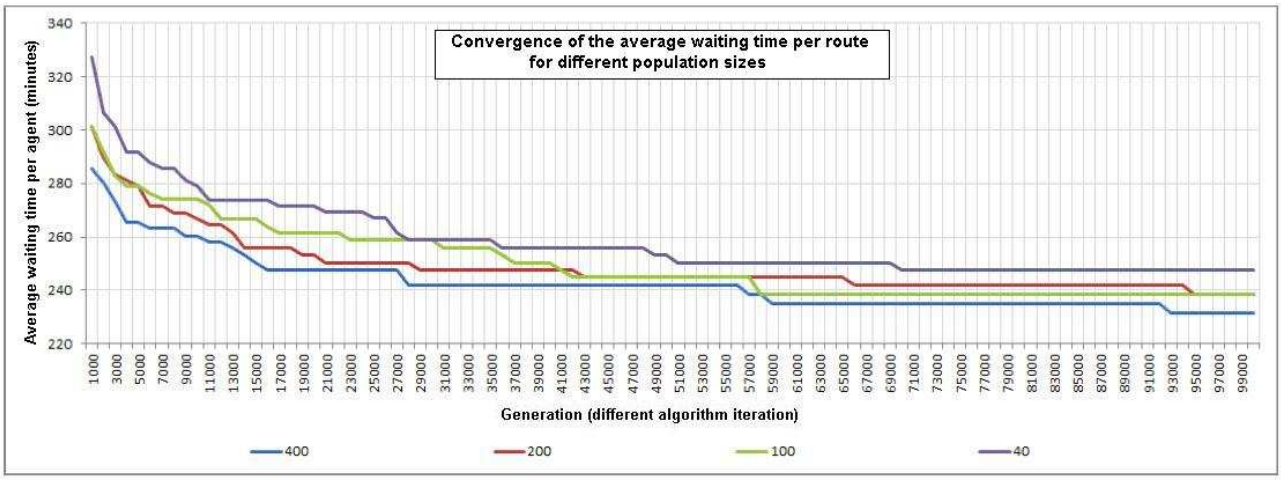


**Figure 3.14 –** Average waiting time per iteration varying the population size used

| Population size | Optimum | Average waiting time per route | Difference |
|---|---|---|---|
| 40 | 135 | 248±11 | 45.56% |
| 100 | 135 | 239±11 | 43.51% |
| 200 | 135 | 239±11 | 43.51% |
| 400 | 135 | 232±16 | 41.81% |

**Table 3.6 –** Final conversion of the waiting traveling time per route for the different population sizes

Though the graphic on Figure 3.14, it is observed that the variable waiting time behaves in a similar way as the variable number of agents needed, i.e. the smaller the population size used, bigger will be the average waiting time (smaller efficiency) estimated initially, however, this value is reduced with the iterations and stabilizes (theoretically, at infinity, it would tend to stabilize at the value of the optimum solution).

The last variable analyzed in Figure 3.15 and Table 3.7 concerns the average travelling time per route, i.e. the total time of dislocation between patients. It is expected that the behavior of this variable is the opposite from the previously variable analyzed and

so better solutions have longer travelling times. That because the larger the number of patients per route, bigger will be the traveling time consumed to treat them and smaller the number of agents needed.

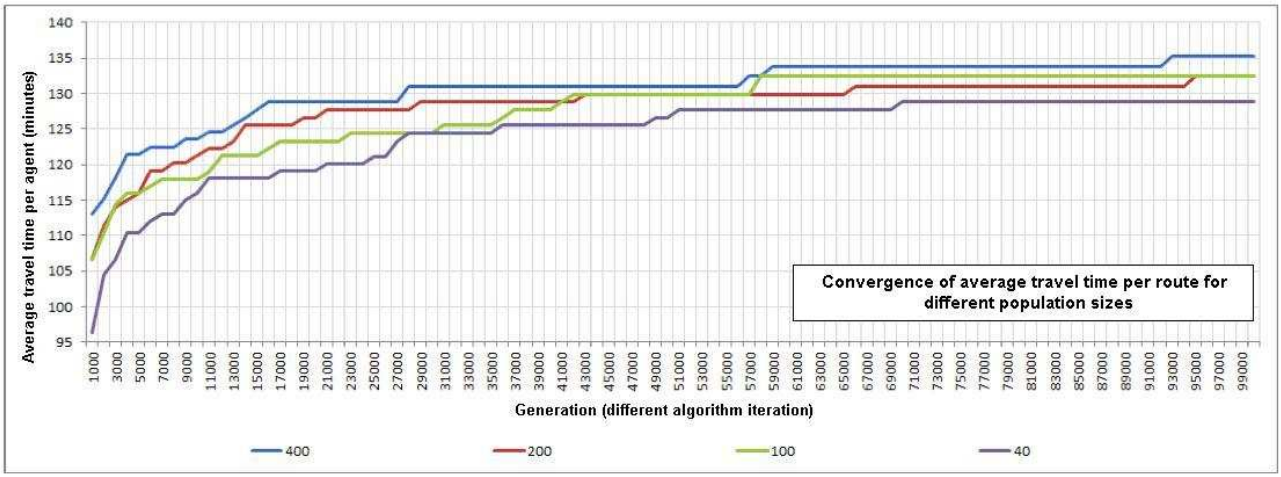


**Figure 3.15 –** Average travelling time per iteration varying the population size used

| Population size | Optimum | Average waiting time per route | Difference |
|---|---|---|---|
| 40 | 217.5 | 129±4 | 40.69% |
| 100 | 217.5 | 132±4 | 39.31% |
| 200 | 217.5 | 132±4 | 39.31% |
| 400 | 217.5 | 135±6 | 37.93% |

**Table 3.7 –** Final conversion of the average traveling time per route for the differentpopulation sizes

The variable analyzed on Figure 3.15 behaved as expected, i.e. stabilized in bigger travelling times with the increase of iterations, due to the reduction of the number of agents needed (equivalent to more patients per route).

To finish the comparison between the four population sizes, it was consolidated the previously analyses using the evaluation function described in section 3.2.1 using weights equal to the tree variables. And so obtaining the graphic of the evaluation value of the solutions from Figure 3.16 and Table 3.8.

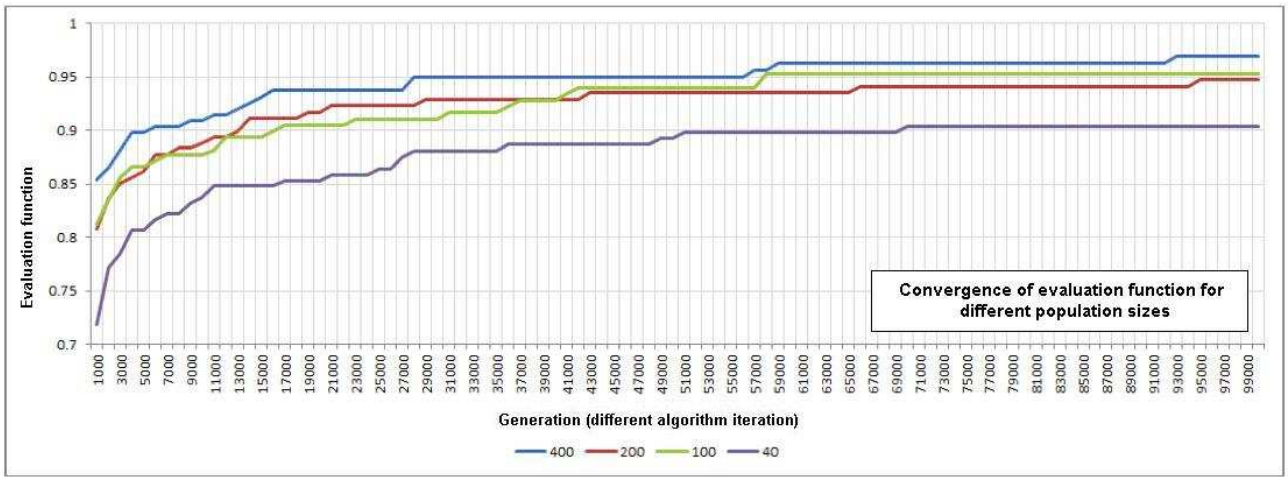


**Figure 3.16 –** Evaluation of the solutions per iteration varying the population size used

| Population size | Evaluation |
|---|---|
| 40 | 0.90±0.03 |
| 100 | 0.95±0.02 |
| 200 | 0.95±0.02 |
| 400 | 0.97±0.03 |

**Table 3.8–**Final conversion of the evaluation function for different population sizes

As initially expected, the solutions founded have smaller evaluation values, this value increases with the iterations and stabilizes (converges), theoretically, at the evaluation value of the optimum solution, i.e. the solution that better represents the allocation of the agents to the patients in the context of execution, considering the tree variables analyzed previously.

Considering the execution time and the quality of the solutions founded, it was adopted, to the next analyses and comparisons, the value of the population size of 100 chromosomes, since this value represented the best cost-benefit relation among the ones considered.

### 3.4.4 Analyses of the stopping criteria by iteration number with and without the proposed Local Search

This section has as goal to analyze the stopping criteria using the maximum number of iterations, for the genetic algorithm and to its combination with the local search. The tree variables (number of agents, average traveling and waiting time) will be analyzed separated from each other in the final conversion (given a maximum value of iterations)

and to the conversion in 1000 initial iterations in the algorithm, in which is noticeable the superiority of the combined execution.

The analyses were realized using the configuration of the genetic algorithm presented on Table 3.9.

| Parameter | Configuration |
|---|---|
| Population size | 100 |
| Evaluation Function | All analyzed variables with same weight 1 |
| Selection | Proportional, RouletteWheel |
| Crossover | Crossover OX |
| Mutation | Inversion, 5% |
| Elitism | 10% of the population size |

**Table 3.9 –** Genetic algorithm configuration for the conversion analyses in function of the stopping criteria

Table 3.10 illustrates the number of iterations defined to the size of nodes at the base.

| Number of nodes | Iterations |
|---|---|
| 50 | 50,000 |
| 60 | 50,000 |
| 70 | 50,000 |
| 80 | 100,000 |
| 90 | 100,000 |
| 100 | 100,000 |
| 110 | 120,000 |
| 120 | 120,000 |

**Table 3.10 –** Number of maximum iterations per number of nodes

**Number of agents used**

The first variable studied is the number of agents used in the solution. Figure 3.17 and Figure 3.18 and Table 3.11, Table 3.12 and Table 3.13 illustrate the behavior of this variable in the different scenarios analyzed.

| GA | GA+LS |
|---|---|
| 12.2±0.8 | 8.6±0.5 |

**Table 3.11 –** Comparison of conversion of number of agents between genetic algorithm with and without local search in the 1000 initial iterations for 50 nodes

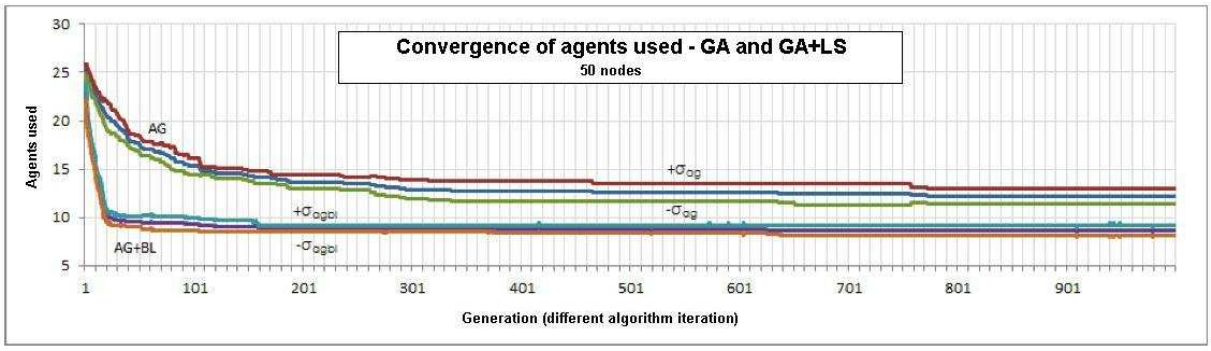


**Figure 3.17 -** Comparison of conversion of number of agents between genetic algorithm with and without local search in the 1000 initial iterations for 50 nodes

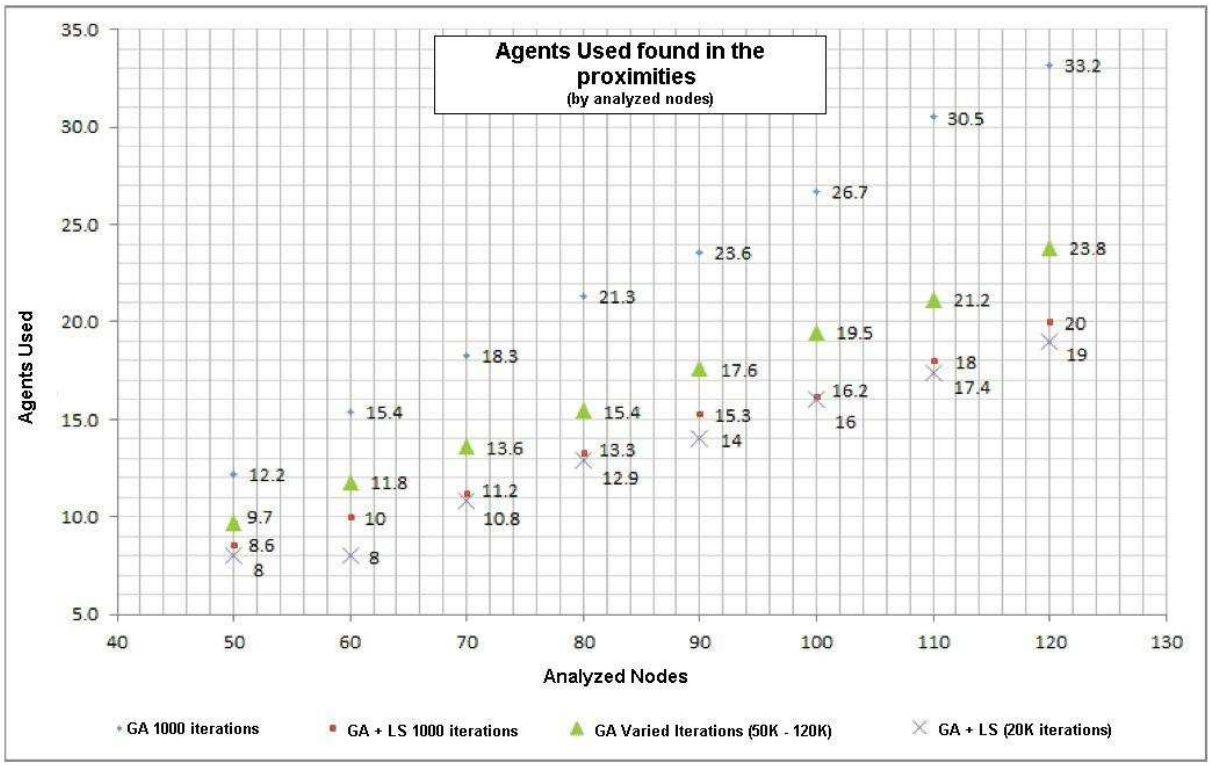


**Figure 3.18 -** Comparison of conversion of number of agents between genetic algorithm with and without local search in different number of iterations

| Nodes | Optimum | GA | Difference | GA+LS | Difference |
|---|---|---|---|---|---|
| 50 | 8 | 12.2±0.8 | 34.43% | 8.6±0.5 | 6.98% |
| 60 | 8 | 15.4±0.8 | 48.05% | 10.0±0.0 | 20.00% |
| 70 | 10 | 18.3±1.2 | 45.36% | 11.2±0.4 | 10.71% |
| 80 | 12 | 21.3±0.9 | 43.75% | 13.3±0.5 | 9.77% |
| 90 | 14 | 23.6±0.7 | 40.68% | 15.3±0.5 | 8.50% |
| 100 | 16 | 26.7±0.8 | 40.07% | 16.2±0.4 | 1.23% |
| 110 | 16 | 30.5±2,1 | 47.54% | 18.0±0.0 | 11.11% |
| 120 | 18 | 33.2±1.0 | 45.78% | 20.0±0.0 | 10.00% |

**Table 3.12 -** Comparison of conversion of number of agents between genetic algorithm with and without local search in the 1000 initial iterations

| Nodes | Optimum | GA | Difference | GA+LS | Difference |
|---|---|---|---|---|---|
| 50 | 8 | 9.7±0.5 | 17.53% | 8.0±0.0 | 0.00% |
| 60 | 8 | 11.8±0.4 | 32.20% | 8.0±0.0 | 0.00% |
| 70 | 10 | 13.6±0.7 | 26.47% | 10.8±0.4 | 7.41% |
| 80 | 12 | 15.4±0.5 | 22.30% | 12.9±0.3 | 6.98% |
| 90 | 14 | 17.6±0.7 | 20.45% | 14.0±0.0 | 0.00% |
| 100 | 16 | 19.5±0.7 | 17.95% | 16.0±0.0 | 0.00% |
| 110 | 16 | 21.2±0.6 | 24.53% | 17.4±0.7 | 8.05% |
| 120 | 18 | 23.8±0.9 | 24.37% | 19.0±0.0 | 5.26% |

**Table 3.13 -** Comparison of conversion of number of agents between genetic algorithm with and without local search in the number of maximum iteration

### Average waiting time per route

Figure 3.19 and Figure 3.20 and Table 3.14, Table 3.15 and Table 3.16 show the behavior of the variable average waiting time per route.

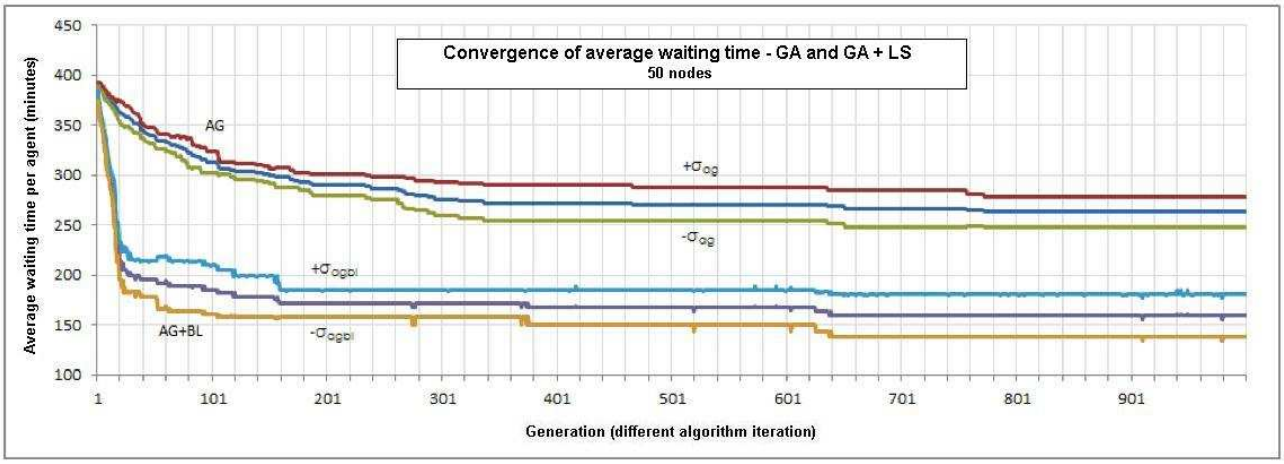


**Figure 3.19 –** Comparison of conversion of average waiting time per route between genetic algorithm with and without local search in the 1000 initial iterations for 50 nodes

| GA | GA+LS |
|---|---|
| 263.1±15.0 | 159.6±21.2 |

**Table 3.14-** Comparison of conversion of average waiting time per route between genetic algorithm with and without local search in the 1000 initial iterations for 50 nodes

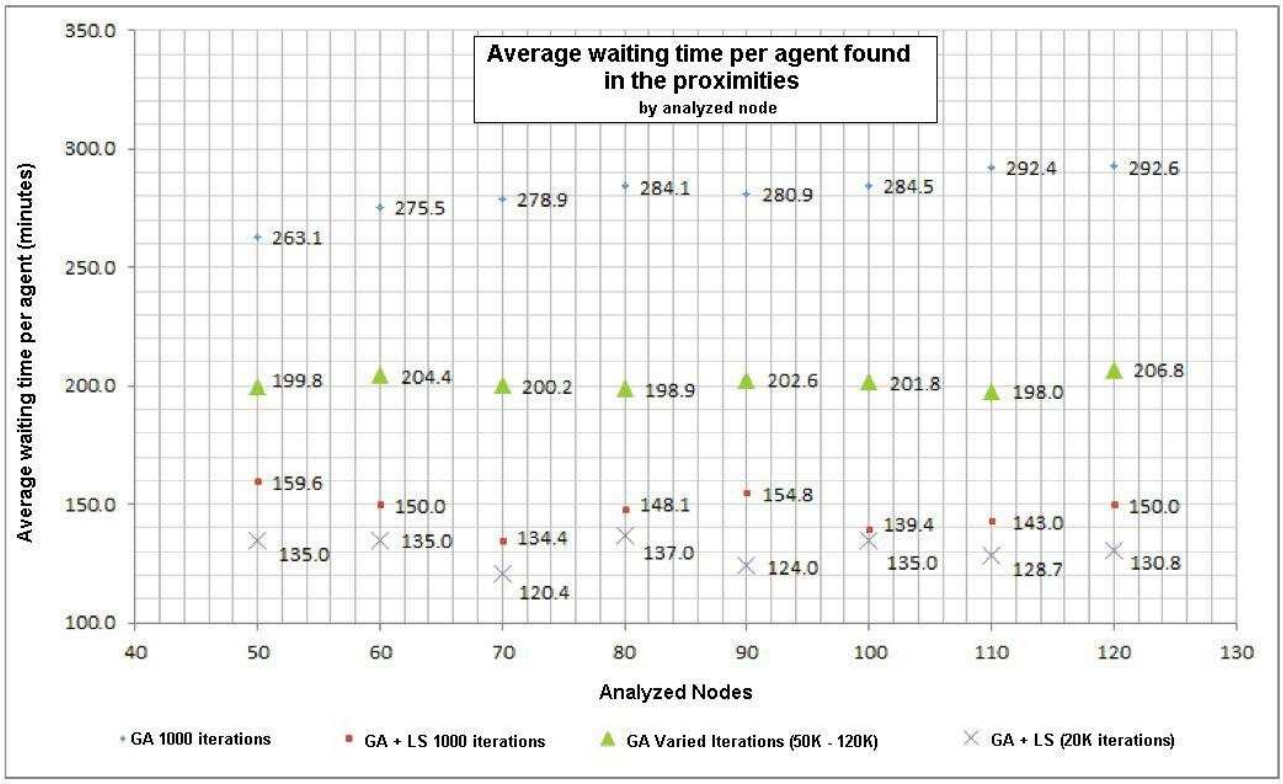


**Figure 3.20 -** Comparison of conversion of average waiting time per route between genetic algorithm with and without local search in different number of iterations

| Nodes | Optimum | GA | Difference | GA+LS | Difference |
|---|---|---|---|---|---|
| 50 | 135.0 | 263±15.0 | 48.69% | 159.6±21.2 | 15.41% |
| 60 | 135.0 | 275.5±12.8 | 51.00% | 150.0±0.0 | 10.00% |
| 70 | 90.0 | 278.9±15.3 | 67.73% | 134.4±13.5 | 33.04% |
| 80 | 110.0 | 284.1±9.0 | 61.28% | 148.1±13.0 | 25.73% |
| 90 | 117.9 | 280.9±6.8 | 58.04% | 154.8±8.1 | 23.86% |
| 100 | 112.5 | 284.5±6.8 | 60.46% | 139.4±9.3 | 19.30% |
| 110 | 69.4 | 292.4±13.9 | 76.27% | 143.0±0.0 | 51.47% |
| 120 | 85.0 | 292.6±6.7 | 70.95% | 150.0±0.0 | 43.33% |

**Table 3.15 -** Comparison of conversion of average waiting time per route between genetic algorithm with and without local search in the 1000 initial iterations

| Nodes | Optimum | GA | Difference | GA+LS | Difference |
|---|---|---|---|---|---|
| 50 | 135.0 | 199.8±16.4 | 32.43% | 135.0±0.0 | 0.00% |
| 60 | 135.0 | 204.4±11.8 | 33.95% | 135.0±0.0 | 0.00% |
| 70 | 90.0 | 200.2±17.3 | 55.04% | 120.4±16.0 | 25.25% |
| 80 | 110.0 | 198.9±10.5 | 44.69% | 137.0±9.5 | 19.71% |
| 90 | 117.9 | 202.6±13.1 | 41.83% | 124.0±0.0 | 4.95% |
| 100 | 112.5 | 201.8±10.9 | 44.25% | 135.0±0.0 | 16.67% |
| 110 | 69.4 | 198.0±9.5 | 64.95% | 128.7±12.8 | 46.08% |
| 120 | 85.0 | 206.8±12.3 | 58.90% | 130.8±0.6 | 35.02% |

**Table 3.16 -** Comparison of conversion of average waiting time per route between genetic algorithm with and without local search in the number of maximum iterations

### Average travelling time per route

Figure 3.21 and Figure 3.22 and Table 3.17, Table 3.18 and Table 3.19 illustrate the behavior of the variable average travelling time per route.

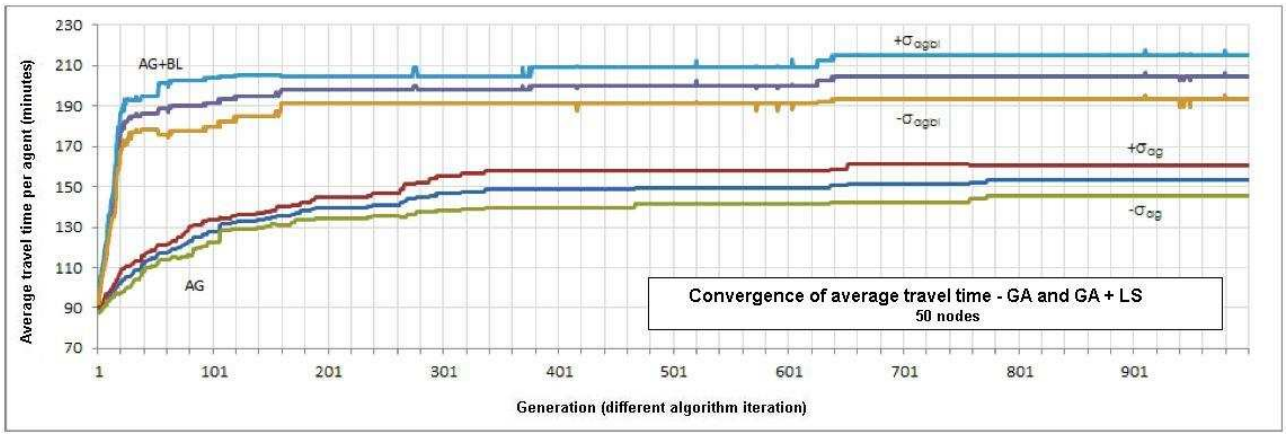


**Figure 3.21 -** Comparison of conversion of average traveling time per route between genetic algorithm with and without local search in the 1000 initial iterations for 50 nodes

| GA | GA+LS |
|---|---|
| 153.3±7.6 | 204.4±10.8 |

**Table 3.17 -** Comparison of conversion of average traveling time per route between genetic algorithm with and without local search in the 1000 initial iterations for 50 nodes

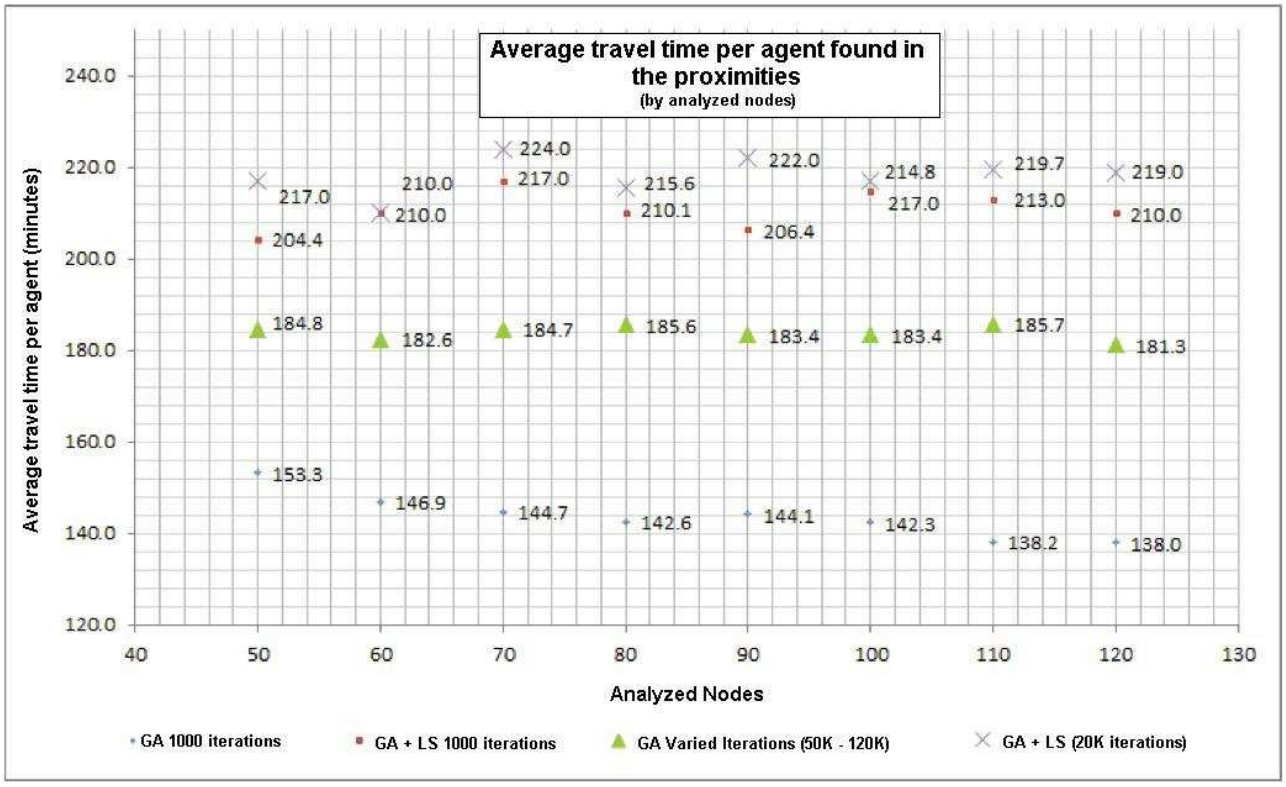


**Figure 3.22 -** Comparison of conversion of average traveling time per route between genetic algorithm with and without local search in different number of iterations

| Nodes | Optimum | GA | Difference | GA+LS | Difference |
|---|---|---|---|---|---|
| 50 | 217.5 | 153.3±7.6 | 29.52% | 204.4±10.8 | 6.02% |
| 60 | 210.0 | 146.9±6.5 | 30.05% | 210.0±0.0 | 0.00% |
| 70 | 240.0 | 144.7±7.6 | 39.71% | 217.0±6.3 | 9.58% |
| 80 | 230.0 | 142.6±4.7 | 38.02% | 210.1±6.3 | 8.65% |
| 90 | 222.9 | 144.1±3.3 | 35.34% | 206.4±5.8 | 7.39% |
| 100 | 217.5 | 142.3±3.5 | 34.57% | 214.8±4.6 | 1.24% |
| 110 | 236.6 | 138.2±6.8 | 41.59% | 213.0±0.0 | 9.97% |
| 120 | 230.0 | 138.0±3.6 | 40.00% | 210.0±0.0 | 8.70% |

**Table 3.18 -** Comparison of conversion of average traveling time per route between genetic algorithm with and without local search in the 1000 initial iterations

It is clear by the tests and comparisons realized that the execution of the genetic algorithm combined with the local search converge to better results when compared to the execution of the genet algorithm alone. Even though the execution of the combination demands of a bigger time, when compared to genetic algorithm alone, the conversion to better results occurs faster, requiring less iterations. Therefore, the algorithm sub-system proposed to the final system will use the local search and will be configured to a maximum (stopping criteria) of 20,000 iterations.

| Nodes | Optimum | GA | Difference | GA+LS | Difference |
|---|---|---|---|---|---|
| 50 | 217.5 | 184.8±7.7 | 15.03% | 217.0±0.0 | 0.23% |
| 60 | 210.0 | 182.6±5.5 | 13.05% | 210.0±0.0 | 0.00% |
| 70 | 240.0 | 184.7±8.5 | 23.04% | 224.0±8.4 | 6.67% |
| 80 | 230.0 | 185.6±5.3 | 19.32% | 215.6±5.1 | 6.26% |
| 90 | 222.9 | 183.4±6.1 | 17.71% | 222.0±0.0 | 0.39% |
| 100 | 217.5 | 183.4±5.2 | 15.68% | 217.0±0.0 | 0.23% |
| 110 | 236.6 | 185.7±4.6 | 21.51% | 219.7±7.4 | 7.14% |
| 120 | 230.0 | 181.3±5.7 | 21.17% | 219.0±0.0 | 4.78% |

**Table 3.19 -** Comparison of conversion of average traveling time per route between genetic algorithm with and without local search in the number of maximum iterations

### 3.4.5 Selection Operators

At this section, the comparison between operator by Tournament and Proportional will be analyzed. First, it was analyzed the selection pressure of the operators, varying its entry parameters, in order to verify in the practice the theory related to the operators, with the objective of choosing the best configuration to the final algorithm. The evaluation parameters are found on Table 3.20.

| Parameter | Configuration |
|---|---|
| Population size | 100 |
| Evaluation Function | All analyzed variables with same weight 1 |
| Selection | Analyzed parameter |
| Crossover | Crossover OX |
| Mutation | 5% |
| Elitism | 10% of the population size |

**Table 3.20 –** Genetic algorithm configuration for analyses of selection operators

#### 3.4.5.1 Tournament Selection

According to (28), the bigger the tournament size, bigger will be the selection pressure associated to the operator, i.e. faster will occur the conversion of the algorithm. This fast conversion can result in local maximums; however, a small selection pressure can result in a high execution time before a good solution is found. Therefore, it is important to have a balance between selection pressure and the required quality; this

balance has a subjective character and can prove the art touch needed to handle the genetic algorithms.

The selection pressure of the Tournament selection was analyzed varying the size of the tournament in question. With the increase of the tournament size, it is expected the conversion to be faster, and a bigger selection pressure, because the probability of worst solutions to be chosen is reduced. The variable agents used was analyzed according to the tournament size. The first curve represents the worst solution founded, the central curve represents the average solution of population and the third represents the best solution founded. It is possible to observe that the bigger the population size, faster the curves converge and approximate, and so the quality of the solutions tend to be equal. At Table 3.21 and

| Tournament Size | Optimum | Best Solution | Difference | Average Solution | Difference | Worst Solution | Difference |
|---|---|---|---|---|---|---|---|
| 2 | 8 | 13.0±1.2 | 38.46% | 13.1±1.2 | 38.73% | 15.0±1.2 | 46.67% |
| 4 | 8 | 12.9±0.7 | 37.98% | 12.9±0.7 | 38.19% | 14.6±0.7 | 45.21% |
| 6 | 8 | 12.9±1.1 | 37.98% | 13.0±1.1 | 38.24% | 15.0±1.2 | 46.67% |
| 8 | 8 | 12.9±0.9 | 37.98% | 12.9±0.9 | 38.17% | 14.4±0.8 | 44.44% |
| 10 | 8 | 13.4±0.7 | 40.30% | 13.5±0.7 | 40.53% | 15.4±0.7 | 48.05% |
| 20 | 8 | 13.2±0.6 | 39.39% | 13.2±0.6 | 39.61% | 14.8±0.9 | 45.95% |

Table 3.22 it was done a comparison between the conversion values to 100 iterations and 1000 iterations.

| Tournament Size | Optimum | Best Solution | Difference | Average Solution | Difference | Worst Solution | Difference |
|---|---|---|---|---|---|---|---|
| 2 | 8 | 14.7±0.9 | 45.58% | 15.0±1.2 | 46.71% | 17.1±2.4 | 53.22% |
| 4 | 8 | 14.6±1.4 | 45.21% | 14.7±1.3 | 45.76% | 16.6±1.1 | 51.81% |
| 6 | 8 | 15.4±1.4 | 48.05% | 15.5±1.6 | 48.48% | 16.8±1.7 | 52.38% |
| 8 | 8 | 15.7±0.9 | 49.04% | 15.8±1.0 | 49.43% | 17.9±1.1 | 55.31% |
| 10 | 8 | 15.7±0.8 | 49.04% | 15.8±0.8 | 49.22% | 17.6±0.7 | 54.55% |
| 20 | 8 | 15.1±1.8 | 47.02% | 15.1±1.8 | 47.15% | 16.6±1.8 | 51.81% |

**Table 3.21 –** Conversion values of agents number after 100 iterations for the tournament selection operator with different tournaments sizes for 50 nodes

| Tournament Size | Optimum | Best Solution | Difference | Average Solution | Difference | Worst Solution | Difference |
|---|---|---|---|---|---|---|---|
| 2 | 8 | 13.0±1.2 | 38.46% | 13.1±1.2 | 38.73% | 15.0±1.2 | 46.67% |
| 4 | 8 | 12.9±0.7 | 37.98% | 12.9±0.7 | 38.19% | 14.6±0.7 | 45.21% |
| 6 | 8 | 12.9±1.1 | 37.98% | 13.0±1.1 | 38.24% | 15.0±1.2 | 46.67% |
| 8 | 8 | 12.9±0.9 | 37.98% | 12.9±0.9 | 38.17% | 14.4±0.8 | 44.44% |
| 10 | 8 | 13.4±0.7 | 40.30% | 13.5±0.7 | 40.53% | 15.4±0.7 | 48.05% |
| 20 | 8 | 13.2±0.6 | 39.39% | 13.2±0.6 | 39.61% | 14.8±0.9 | 45.95% |

**Table 3.22 -** Conversion values of agents number after 10,000 iterations for the tournament selection operator with different tournaments sizes for 50 nodes

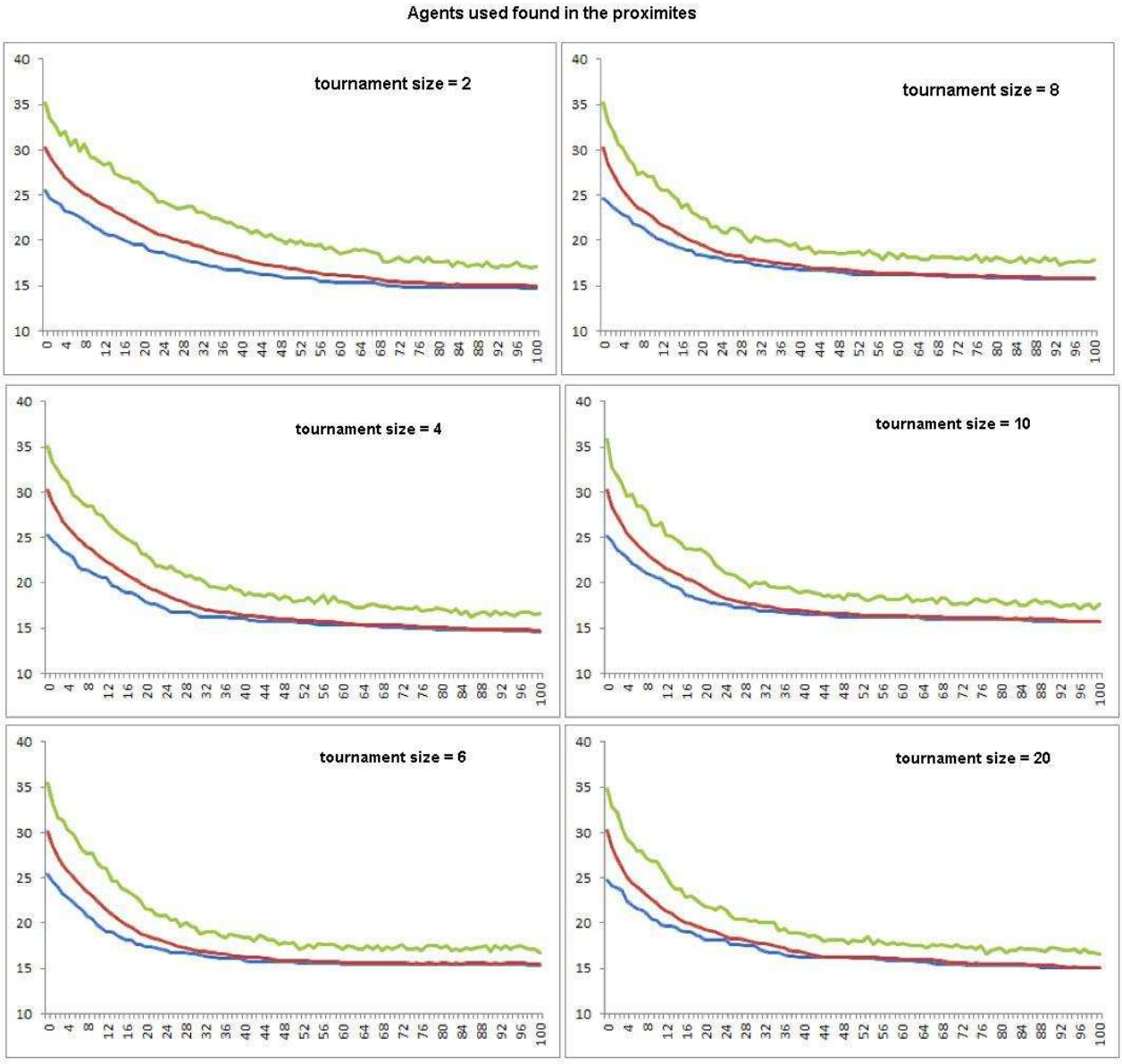


**Figure 3.23 –** Comparison of the conversion for the different tournament sizes of the Tournament selection operator

### 3.4.5.2 Roulette Wheel Selection or Proportional Selection

The implementation of the Proportional selection is totally related to the model of the evaluation function, since the selection is realized proportionally to the evaluation value of each solution. The analyses of this operator will be done in comparison to the previous operator with tournament size *k=2*.

Table 3.23 illustrates the number of iteration defined to the size of nodes at the base.

| Number of nodes | Iterations |
|---|---|
| 50 | 50,000 |
| 60 | 50,000 |
| 70 | 50,000 |
| 80 | 100,000 |
| 90 | 100,000 |
| 100 | 100,000 |
| 110 | 120,000 |
| 120 | 120,000 |

**Table 3.23 –** Number of maximum iterations used per number of nodes for the comparison of selection operators

**Number of agents used**

The first variable analyzed will be the number of agents used in the solution. Figure 3.24 and Table 3.24 and Table 3.25 show the behavior of this variable in the different scenarios of selection operation studied.

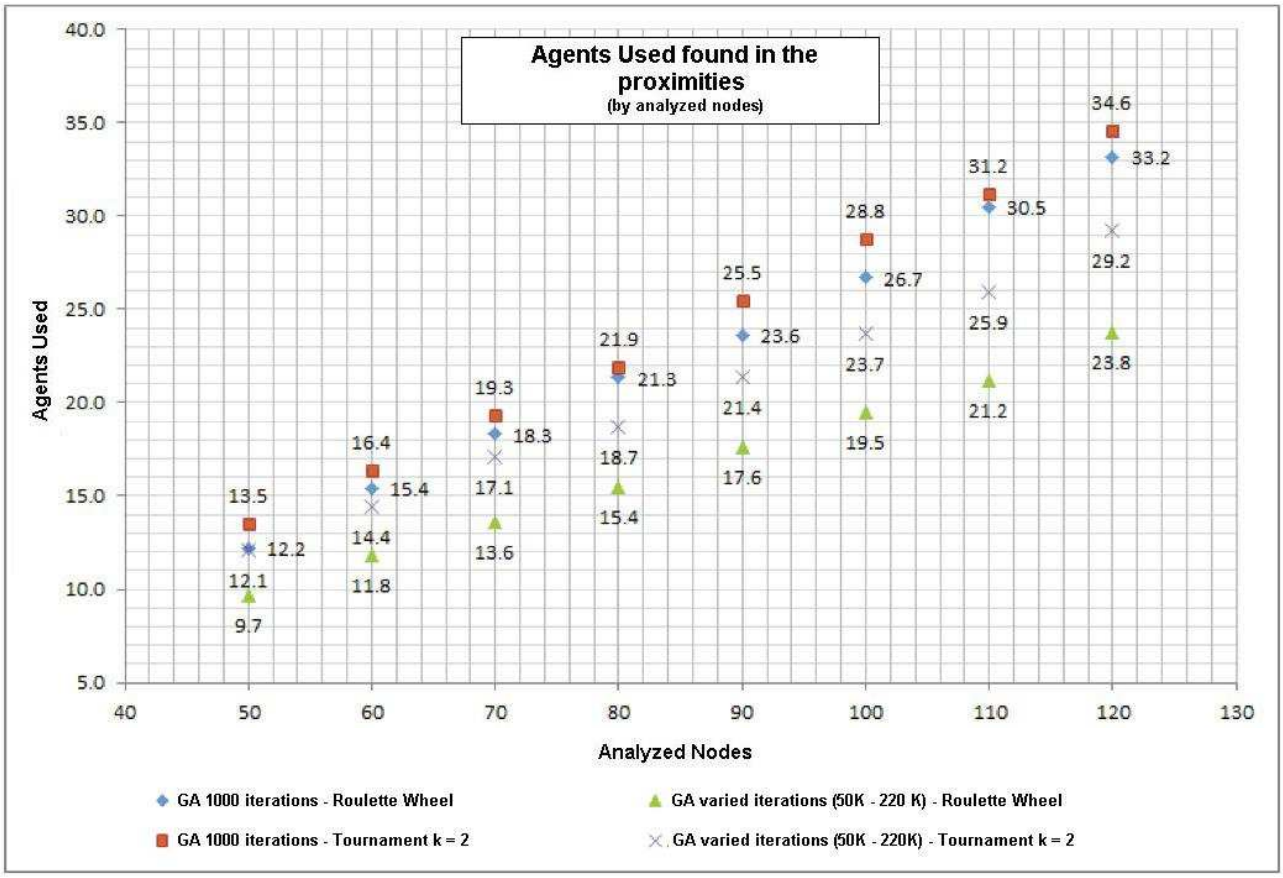


**Figure 3.24–**Comparison between selection operators, variable analyzed: number of agents used

| Nodes | Optimum | Proportional | Difference | Tournament *k=2* | Difference |
|---|---|---|---|---|---|
| 50 | 8 | 12.2±0.8 | 34.43% | 13.5±0.8 | 40.74% |
| 60 | 8 | 15.4±0.8 | 48.05% | 16.4±1.1 | 51.22% |
| 70 | 10 | 18.3±1.2 | 45.36% | 19.3±1.3 | 48.19% |
| 80 | 12 | 21.3±0.9 | 43.75% | 21.9±1.7 | 45.21% |
| 90 | 14 | 23.6±0.7 | 40.68% | 25.5±1.0 | 45.10% |
| 100 | 16 | 26.7±0.8 | 40.07% | 28.8±1.1 | 44.44% |
| 110 | 16 | 30.5±2.1 | 47.54% | 31.2±0.9 | 48.72% |
| 120 | 18 | 33.2±1.0 | 45.78% | 34.6±1.4 | 47.98% |

**Table 3.24 -** Comparison between selection operators, variable analyzed: number of agents used for 1,000 initial iterations

| Nodes | Optimum | Proportional | Difference | Tournament *k=2* | Difference |
|---|---|---|---|---|---|
| 50 | 8 | 9.7±0.5 | 17.53% | 12.1±0.6 | 33.88% |
| 60 | 8 | 11.8±0.4 | 32.20% | 14.4±0.8 | 44.44% |
| 70 | 10 | 13.6±0.7 | 26.47% | 17.1±0.7 | 41.52% |
| 80 | 12 | 15.4±0.5 | 22.30% | 18.7±0.7 | 35.83% |
| 90 | 14 | 17.6±0.7 | 20.45% | 21.4±1.3 | 34.58% |
| 100 | 16 | 19.5±0.7 | 17.95% | 23.7±0.8 | 32.49% |
| 110 | 16 | 21.2±0.6 | 24.53% | 25.9±1.1 | 38.22% |
| 120 | 18 | 23.8±0.9 | 24.37% | 29.2±1.1 | 38.36% |

**Table 3.25 -** Comparison between selection operators, variable analyzed: number of agents used for maximum iterations

### Average waiting time per route

Figure 3.25 and Table 3.26 and Table 3.27 illustrate the behavior of the variable average waiting time per route.

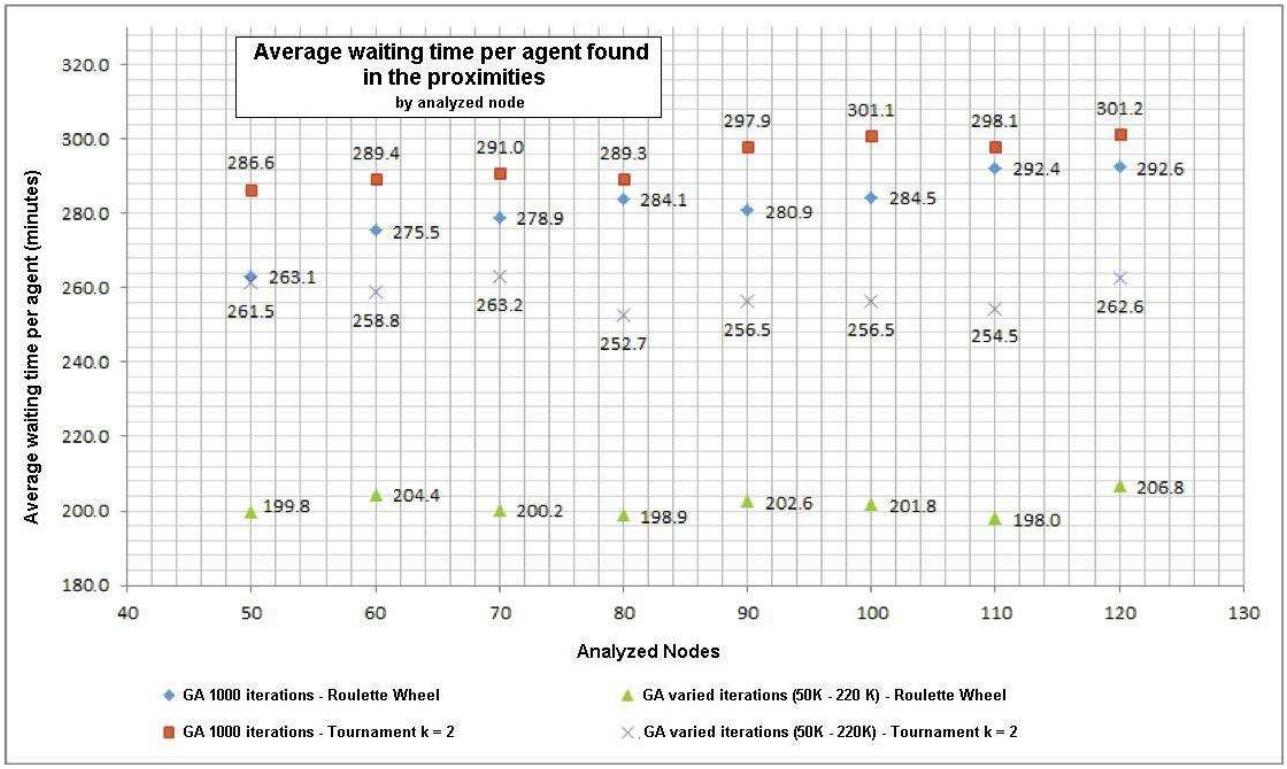


**Figure 3.25 -** Comparison between selection operators, variable analyzed: average waiting time per route

| Nodes | Optimum | Proportional | Difference | Tournament k=2 | Difference |
|---|---|---|---|---|---|
| 50 | 135.0 | 263.1±15.0 | 48.69% | 286.6±14.0 | 52.90% |
| 60 | 135.0 | 275.5±12.8 | 51.00% | 289.4±15.6 | 53.35% |
| 70 | 90.0 | 278.9±15.3 | 67.73% | 291.0±14.9 | 69.07% |
| 80 | 110.0 | 284.1±9.0 | 61.28% | 289.3±15.2 | 61.98% |
| 90 | 117.9 | 280.9±6.8 | 58.04% | 297.9±8.0 | 60.44% |
| 100 | 112.5 | 284.5±6.8 | 60.46% | 301.1±8.7 | 62.64% |
| 110 | 69.4 | 292.4±13.9 | 76.27% | 298.1±6.0 | 76.72% |
| 120 | 85.0 | 292.6±6.7 | 70.95% | 301.2±8.8 | 71.78% |

**Table 3.26 -** Comparison between selection operators, variable analyzed: average waiting time per route for 1,000 initial iterations

| Nodes | Optimum | Proportional | Difference | Tournament k=2 | Difference |
|---|---|---|---|---|---|
| 50 | 135.0 | 199.8±16.4 | 32.43% | 261.5±11.7 | 48.37% |
| 60 | 135.0 | 204.4±11.8 | 33.95% | 258.8±14.8 | 47.84% |
| 70 | 90.0 | 200.2±17.3 | 55.04% | 263.2±10.7 | 65.81% |
| 80 | 110.0 | 198.9±10.5 | 44.69% | 252.7±9.2 | 56.47% |
| 90 | 117.9 | 202.6±13.1 | 41.83% | 256.5±14.8 | 54.05% |
| 100 | 112.5 | 201.8±10.9 | 44.25% | 256.5±8.7 | 56.14% |
| 110 | 69.4 | 198.0±9,5 | 64.95% | 254.5±11.1 | 72.73% |
| 120 | 85.0 | 206.8±12.3 | 58.90% | 262.6±9.9 | 67.63% |

**Table 3.27 -** Comparison between selection operators, variable analyzed: average waiting time per route for maximum iterations

### Average travelling time per route

Figure 3.26 and Table 3.28 and Table 3.29 illustrate the behavior of the variable average travelling time per route.


**Figure 3.26 -** Comparison between selection operators, variable analyzed: average traveling time per route

| Nodes | Optimum | Proportional | Difference | Tournament k=2 | Difference |
|---|---|---|---|---|---|
| 50 | 217.5 | 153.3±7.6 | 29.52% | 141.3±7.0 | 35.03% |
| 60 | 210.0 | 146.9±6.5 | 30.05% | 139.6±7.7 | 33.52% |
| 70 | 240.0 | 144.7±7.6 | 39.71% | 138.9±7.0 | 42.13% |
| 80 | 230.0 | 142.6±4.7 | 38.02% | 139.9±7.7 | 39.17% |
| 90 | 222.9 | 144.1±3.3 | 35.34% | 135.8±4.0 | 39.06% |
| 100 | 217.5 | 142.3±3.5 | 34.57% | 134.1±4.2 | 38.34% |
| 110 | 236.6 | 138.2±6.8 | 41.59% | 135.6±3.1 | 42.69% |
| 120 | 230.0 | 138.0±3.6 | 40.00% | 133.8±4.3 | 41.83% |

**Table 3.28 -** Comparison between selection operators, variable analyzed: average traveling time per route for 1,000 initial iterations

| Nodes | Optimum | Proportional | Difference | Tournament k=2 | Difference |
|---|---|---|---|---|---|
| 50 | 217.5 | 184.8±7.7 | 15.03% | 154.1±5.9 | 29.15% |
| 60 | 210.0 | 182.6±5.5 | 13.05% | 155±7.1 | 26.19% |
| 70 | 240.0 | 184.7±8.5 | 23.04% | 152.5±5.5 | 36.46% |
| 80 | 230.0 | 185.6±5.3 | 19.32% | 158.2±4.5 | 31.22% |
| 90 | 222.9 | 183.4±6.1 | 17.71% | 156.2±7.6 | 29.91% |
| 100 | 217.5 | 183.4±5.2 | 15.68% | 156.5±4.1 | 28.05% |
| 110 | 236.6 | 185.7±4.6 | 21.51% | 157±5.6 | 33.64% |
| 120 | 230.0 | 181.3±5.7 | 21.17% | 153.3±4.8 | 33.35% |

**Table 3.29 -** Comparison between selection operators, variable analyzed: average traveling time per route for maximum iterations

From the previous analyses, the students will use on the final algorithm sub-system the Proportional Selection operator, since this provided the conversion to better values and more consistent when compared to the Tournament Selection operator.

### 3.4.6 Crossover Operators

In this section, the comparison between the crossover operators Ordered (OX) and Partially Mapped (PMX) will be done. The tree variables will be once more analyzed one by one: number of agents used, average waiting time per route and average travelling time per route, according to Table 3.30 and Table 3.31.

| Parameter | Configuration |
|---|---|
| Population size | 100 |
| Evaluation Function | All analyzed variables with same weight 1 |
| Selection | Proportional Selection |
| Crossover | Analyzed parameter |
| Mutation | 5% |
| Elitism | 10% of population |

**Table 3.30 –** Configuration of the genetic algorithm for the analyses of the crossover operator

| Number of nodes | Iterations |
|---|---|
| 50 | 50,000 |
| 60 | 50,000 |
| 70 | 50,000 |
| 80 | 100,000 |
| 90 | 100,000 |
| 100 | 100,000 |
| 110 | 120,000 |
| 120 | 120,000 |

**Table 3.31 –** Number of maximum iterations used per number of nodes for the comparison of the crossover operator

### Number of agents used

The first variable studied is the number of agents used in the solution. Figure 3.27 and Table 3.32 illustrate the behavior of this variable in different scenarios of operation with both crossover studied.

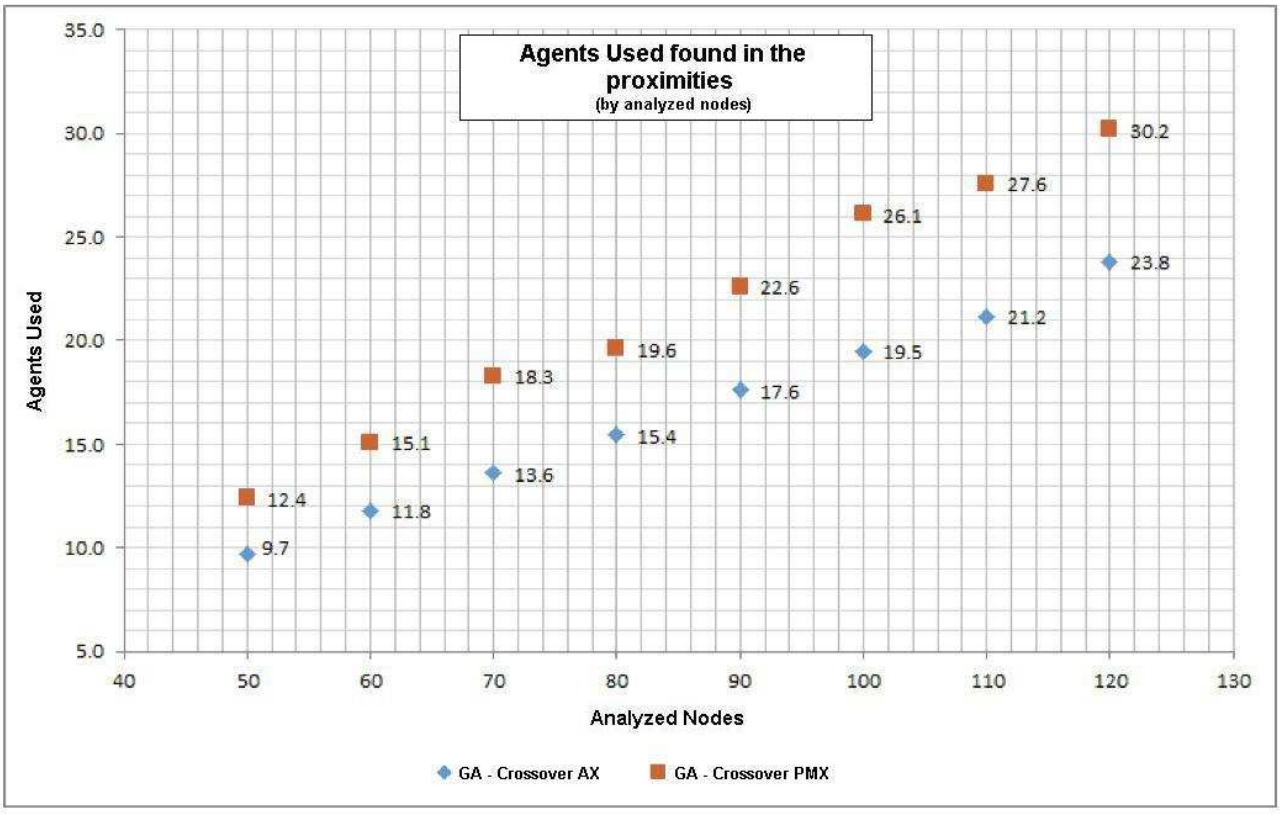


**Figure 3.27 –** Comparison between the crossover operators, variable analyzed: number of agents used

| Nodes | Optimum | OX | Difference | PMX | Difference |
|---|---|---|---|---|---|
| 50 | 8 | 9.7±0.5 | 17.53% | 12.4±1.0 | 35.48% |
| 60 | 8 | 11.8±0.4 | 32.20% | 15.1±1.4 | 47.02% |
| 70 | 10 | 13.6±0.7 | 26.47% | 18.3±0.9 | 45.36% |
| 80 | 12 | 15.4±0.5 | 22.30% | 19.6±1.0 | 38.78% |
| 90 | 14 | 17.6±0.7 | 20.45% | 22.6±1.0 | 38.05% |
| 100 | 16 | 19.5±0.7 | 17.95% | 26.1±1.3 | 38.70% |
| 110 | 16 | 21.2±0.6 | 24.53% | 27.6±1.5 | 42.03% |
| 120 | 18 | 23.8±0.9 | 24.37% | 30.2±1.5 | 40.40% |

**Table 3.32 -** Comparison between the crossover operators, variable analyzed: number of agents used per maximum iterations

### Average waiting time per route

Figure 3.28 and Table 3.33 show the behavior of the variable average waiting time per route.

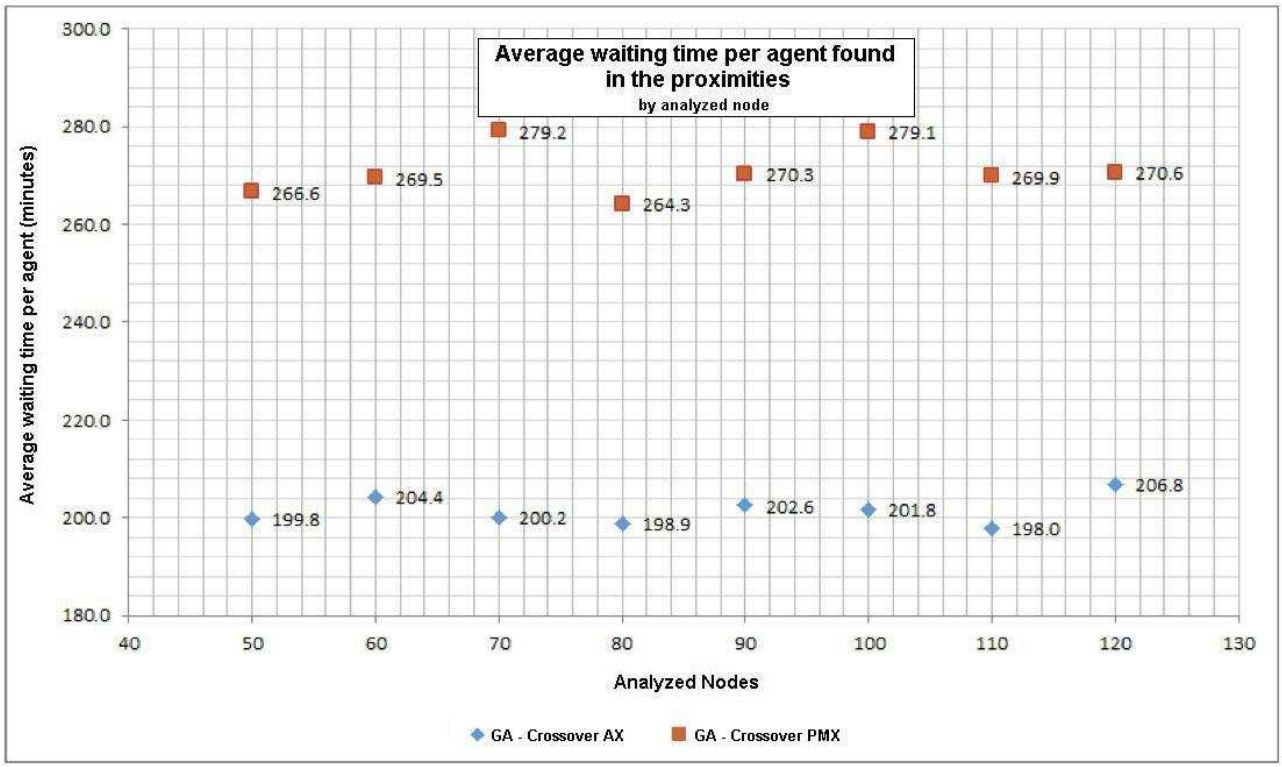


**Figure 3.28 -** Comparison between the crossover operators, variable analyzed: average waiting time per route

| Nodes | Optimum | OX | Difference | PMX | Difference |
|---|---|---|---|---|---|
| 50 | 135.0 | 199.8±16.4 | 32.43% | 266.6±18.0 | 49.36% |
| 60 | 135.0 | 204.4±11.8 | 33.95% | 269.5±18.0 | 49.91% |
| 70 | 90.0 | 200.2±17.3 | 55.04% | 279.2±18.0 | 67.77% |
| 80 | 110.0 | 198.9±10.5 | 44.69% | 264.3±0.0 | 58.38% |
| 90 | 117.9 | 202.6±13.1 | 41.83% | 270.3±0.0 | 56.40% |
| 100 | 112.5 | 201.8±10.9 | 44.25% | 279.1±0.0 | 59.69% |
| 110 | 69.4 | 198.0±9.5 | 64.95% | 269.9±0.0 | 74.29% |
| 120 | 85.0 | 206.8±12.3 | 58.90% | 270.6±0.0 | 68.59% |

**Table 3.33 -** Comparison between the crossover operators, variable analyzed: average waiting time per route for maximum iterations

### Average travelling time per route

Figure 3.29 and Table 3.34 illustrate the behavior of the variable average travelling time per route.

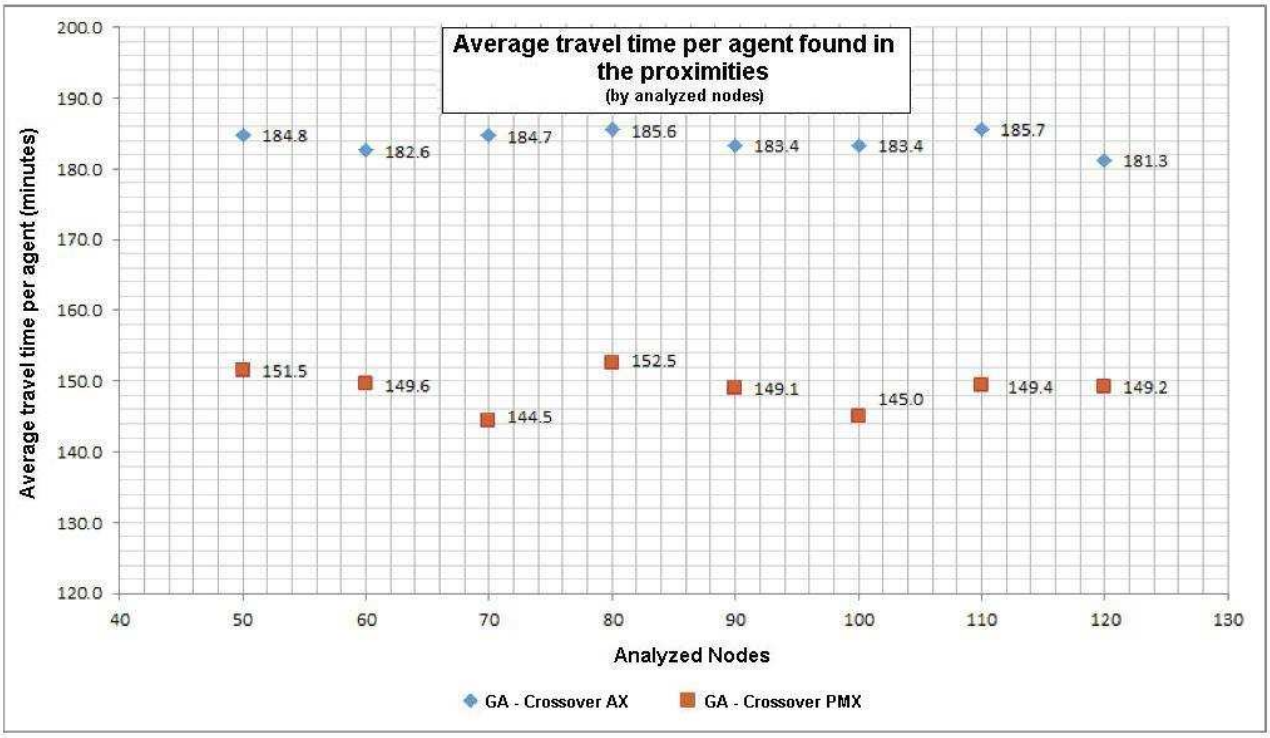


**Figure 3.29 –** Spectrum of **s**olutions illustrating the local maximums

Observing the behavior of both crossover operators, it is possible to visualize the superiority of the Ordered Crossover (OX). Apart from being of easy implementation, the operator allowed to obtain superior values when compared to the Partially Mapped (PMX) operator.

| Nodes | Optimum | OX | Difference | PMX | Difference |
|---|---|---|---|---|---|
| 50 | 217.5 | 184.8±7.7 | 15.03% | 151.5±9.1 | 30.34% |
| 60 | 210.0 | 182.6±5.5 | 13.05% | 149.6±11.0 | 28.76% |
| 70 | 240.0 | 184.7±8.5 | 23.04% | 144.5±5.8 | 39.79% |
| 80 | 230.0 | 185.6±5.3 | 19.32% | 152.5±6.0 | 33.70% |
| 90 | 222.9 | 183.4±6.1 | 17.71% | 149.1±4.6 | 33.10% |
| 100 | 217.5 | 183.4±5.2 | 15.68% | 145.0±6.1 | 33.33% |
| 110 | 236.6 | 185.7±4.6 | 21.51% | 149.4±6.7 | 36.86% |
| 120 | 230.0 | 181.3±5.7 | 21.17% | 149.2±5.9 | 35.13% |

**Table 3.34 -** Comparison between the crossover operators, variable analyzed: average travelling time per route for maximum iterations

### 3.4.7 Mutation Operators

In this section, the two mutation operators Inversion and Removal-and-Insert will be confronted. The variables from the business model will be once more analyzed separately with the aim of choosing the best operator and best parameter of probability of mutation used. The results were obtained using the database of 50 nodes and with stopping criteria of 1,000 iterations; the configuration can be seen on Table 3.35.

| Parameter | Configuration |
|---|---|
| Population size | 100 |
| Evaluation Function | All analyzed variables with same weight 1 |
| Selection | Proportional Selection |
| Crossover | Crossover OX |
| Mutation | Analyzes parameter |
| Elitism | 10% of size population |

**Table 3.35 –** Genetic algorithm configuration for the analyses of mutation operators

**Number of agents used**

Figure 3.30 and Figure 3.31 and Table 3.36 illustrate the behavior of this variable in the different scenarios of probability of both mutation operators.

It can be noticed the formation of a valley to probabilities below 10% and that the values increase with the varying of the mutation probability. Analyzing the theory, it can be concluded that to small values of mutation probability, the variation of solutions is smaller, so the selection pressure is bigger and the conversion to a local maximum is faster. The opposite happens to probability values above 10%, with bigger variation of solutions, the conversion is slower and worst values are founded to the same number of iterations.

Given the results founded, considering the variable agents used, the value of 5% represents a good value to be used to the mutation probability.

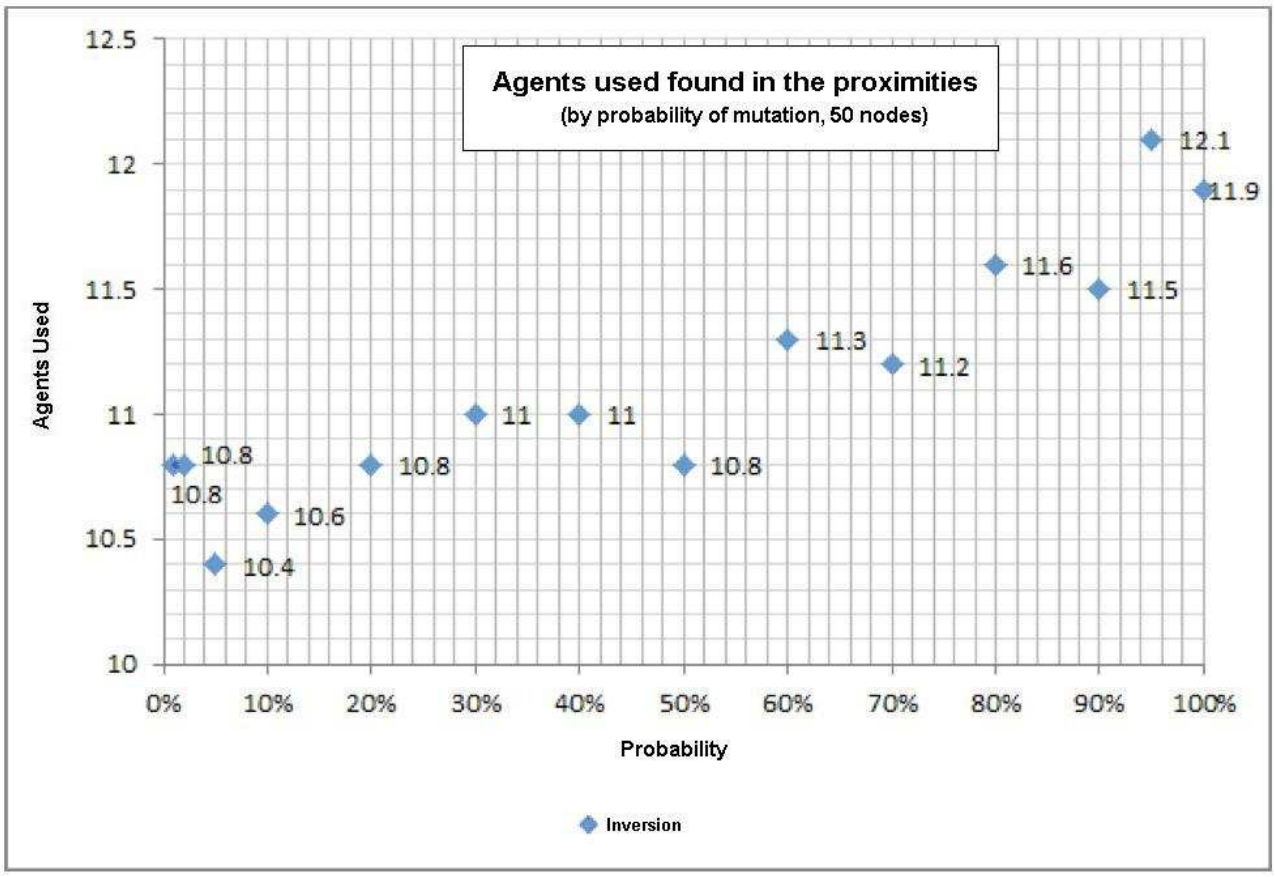


**Figure 3.30 –** Mutation operator – Inversion, variable analyzed: number of agents used

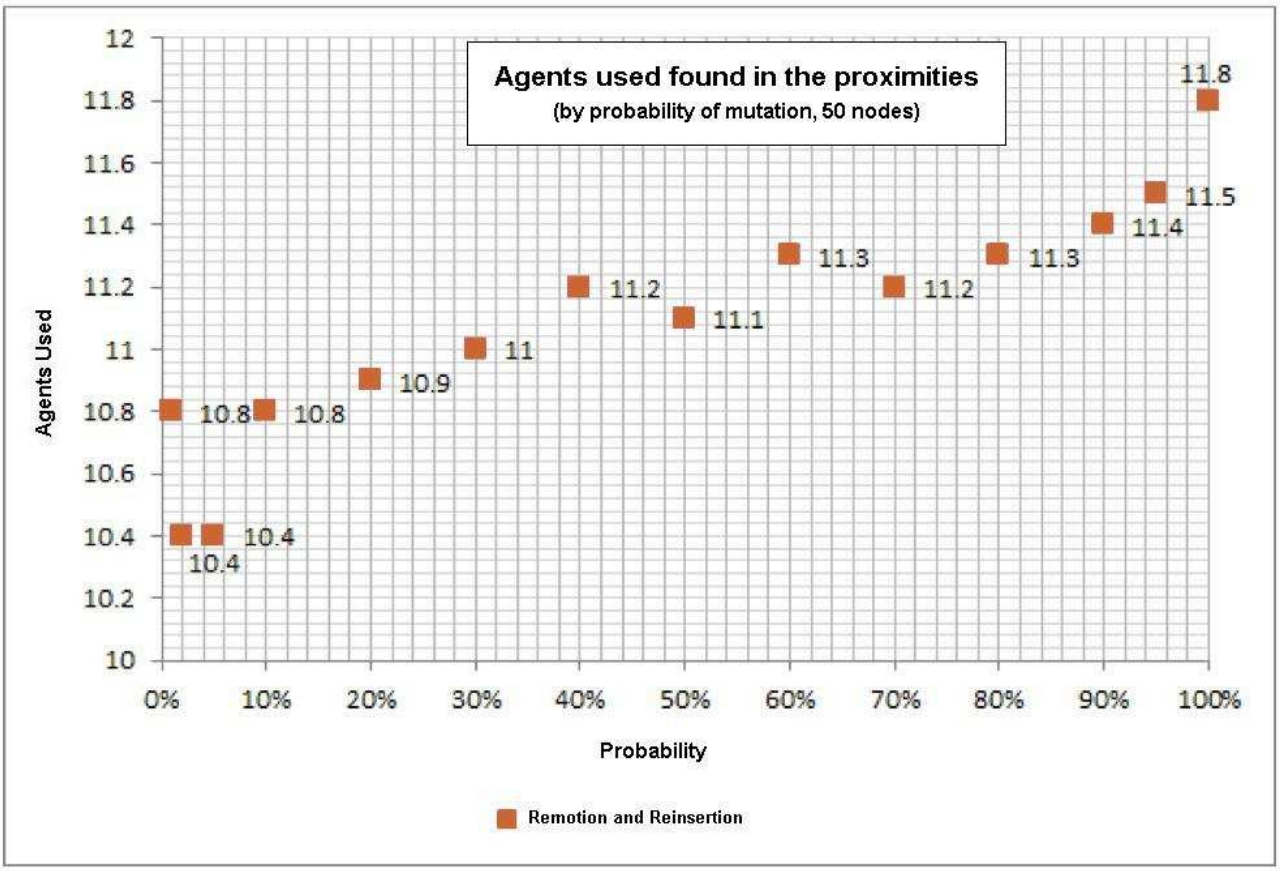


**Figure 3.31 -** Mutation operator – Removal-and-Reinsert, variable analyzed: number of agents used

| Probability | Optimum | Inversion | Difference | Removal-and-Reinsert | Difference |
|---|---|---|---|---|---|
| 1 | 8 | 10.8±0.4 | 25.93% | 10.8±0.5 | 25.93% |
| 2 | 8 | 10.8±0.5 | 25.93% | 10.4±0.7 | 23.08% |
| 5 | 8 | 10.4±0.4 | 23.08% | 10.4±0.4 | 23.08% |
| 10 | 8 | 10.6±0.5 | 24.53% | 10.8±0.4 | 25.93% |
| 20 | 8 | 10.8±0.6 | 25.93% | 10.9±0.3 | 26.61% |
| 30 | 8 | 11.0±0.8 | 27.27% | 11.0±0.5 | 27.27% |
| 40 | 8 | 11.0±0.0 | 27.27% | 11.2±0.5 | 28.57% |
| 50 | 8 | 10.8±0.8 | 25.93% | 11.1±0,5 | 27.93% |
| 60 | 8 | 11.3±0.7 | 29.20% | 11.3±0.7 | 29.20% |
| 70 | 8 | 11.2±0.6 | 28.57% | 11.2±0.5 | 28.57% |
| 80 | 8 | 11.6±0.5 | 31.03% | 11.3±0.5 | 29.20% |
| 90 | 8 | 11.5±0.7 | 30.43% | 11.4±0.5 | 29.82% |
| 95 | 8 | 12.1±0.7 | 33.88% | 11.5±0.5 | 30.43% |
| 100 | 8 | 11.9±0.9 | 32.77% | 11.8±0.3 | 32.20% |

**Table 3.36 –** Comparison between the mutation operators, variable analyzed: number of agents used for 10,000 iterations

**Average waiting time per route**

Figure 3.32 and Figure 3.33 and Table 3.37 and Table 3.38 illustrate the behavior of the variable average waiting time per route.

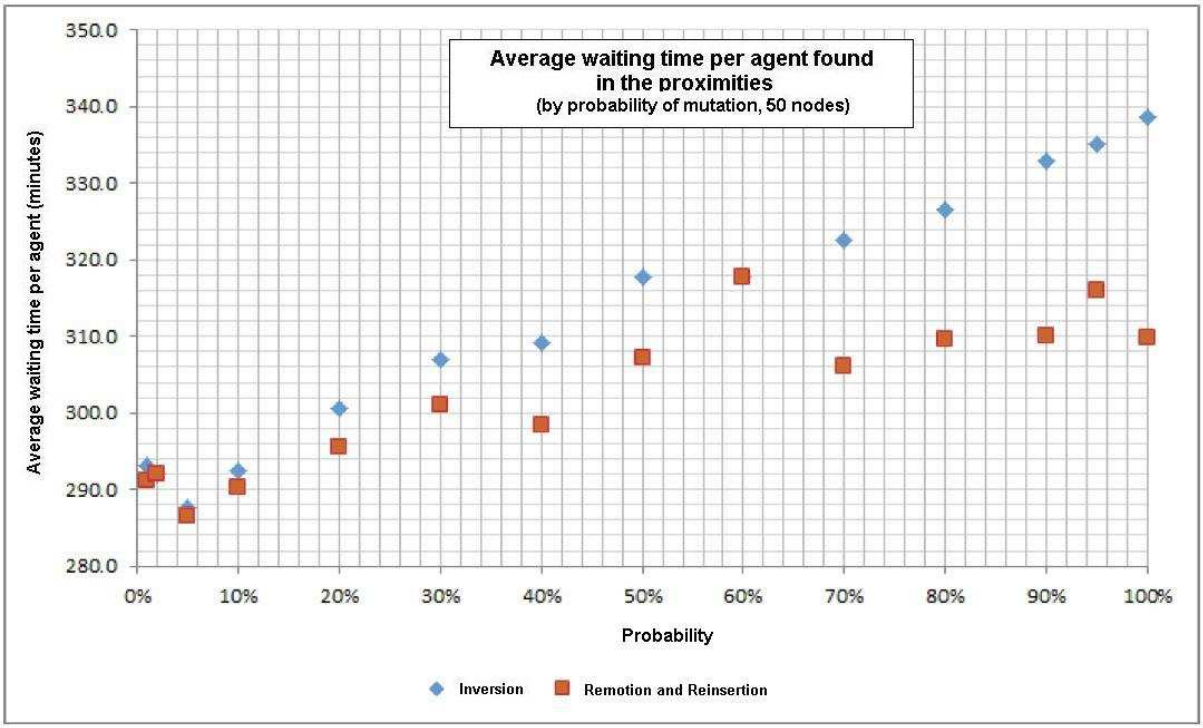


**Figure 3.32 -** Comparison between the mutation operators, variable analyzed: average waiting time per route for 10,000 iterations

| Probability | Optimum | Inversion | Difference | Removal and Reinsert | Difference |
|---|---|---|---|---|---|
| 1 | 135.0 | 293.2±11.4 | 53.96% | 219.2±13.9 | 53.64% |
| 2 | 135.0 | 292.1±13.9 | 53.78% | 292.1±20.5 | 53.78% |
| 5 | 135.0 | 287.6±11.4 | 53.05% | 286.5±11.4 | 52.88% |
| 10 | 135.0 | 292.5±13.9 | 53.84% | 290.2±11.4 | 53.48% |
| 20 | 135.0 | 300.5±16.3 | 55.07% | 295.5±8.5 | 54.32% |
| 30 | 135.0 | 306.9±20.4 | 56.01% | 301.0±0.0 | 55.14% |
| 40 | 135.0 | 309.1±0.0 | 56.33% | 298.5±13.0 | 54.77% |
| 50 | 135.0 | 317.8±20.0 | 57.52% | 307.2±13.0 | 56.05% |
| 60 | 135.0 | 317.7±16.4 | 57.50% | 317.7±16.4 | 57.50% |
| 70 | 135.0 | 322.5±15.4 | 58.14% | 306.0±13.9 | 55.88% |
| 80 | 135.0 | 326.5±11.9 | 58.65% | 309.6±13.9 | 56.39% |
| 90 | 135.0 | 332.8±15.3 | 59.44% | 310.0±14.2 | 56.46% |
| 95 | 135.0 | 335.1±15.4 | 59.71% | 315.9±14.2 | 57.27% |
| 100 | 135.0 | 338.7±18.5 | 60.14% | 309.7±8.5 | 56.42% |

**Table 3.37 -** Comparison between the mutation operators, variable analyzed: average waiting time per route for 10,000 iterations

Analyzing the behavior of the variables for both mutation operators with different probabilities values, it is possible to conclude that the operator Removal-and-Reinsert allows to obtain better results, and, then, will be used to compose the final algorithm to be used in the developed system.

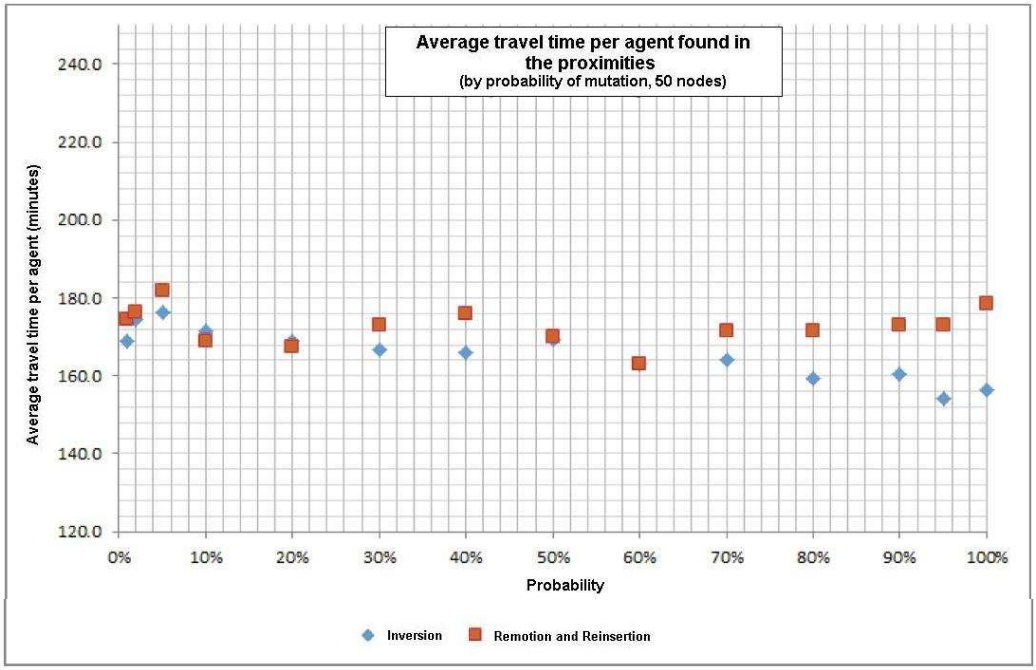


**Figure 3.33 -** Comparison between the mutation operators, variable analyzed: average travelling time per route for 10,000 iterations

| Probability | Optimum | Inversion | Difference | Removal and Reinsert | Difference |
|---|---|---|---|---|---|
| 1 | 217.5 | 168.8±5.9 | 22.39% | 174.4±7.2 | 19.82% |
| 2 | 217.5 | 174.4±7.2 | 19.82% | 176.3±10.2 | 18.94% |
| 5 | 217.5 | 176.2±5.9 | 18.99% | 182.0±5.9 | 16.32% |
| 10 | 217.5 | 171.6±7.2 | 21.10% | 168.8±5.9 | 22.39% |
| 20 | 217.5 | 169.1±8.3 | 22.25% | 167.4±4.4 | 23.03% |
| 30 | 217.5 | 166.9±10.2 | 23.26% | 173.0±7.4 | 20.46% |
| 40 | 217.5 | 166.0±0.0 | 23.68% | 175.8±6.8 | 19.17% |
| 50 | 217.5 | 169.4±10.1 | 22.11% | 170.2±6.8 | 21.75% |
| 60 | 217.5 | 163.0±8.1 | 25.06% | 163.0±8.1 | 25.06% |
| 70 | 217.5 | 164.1±7.6 | 24.55% | 171.6±7.2 | 21.10% |
| 80 | 217.5 | 159.4±5.7 | 26.71% | 171.6±7.2 | 21.10% |
| 90 | 217.5 | 160.6±7.5 | 26.16% | 173.0±7.4 | 20.46% |
| 95 | 217.5 | 154.2±7.7 | 29.10% | 173.0±7.4 | 20.46% |
| 100 | 217.5 | 156.4±9.2 | 28.09% | 178.6±4.4 | 17.89% |

**Table 3.38 -** Comparison between the mutation operators, variable analyzed: average traveling time per route for 10,000 iterations

## 3.5 Dynamic Approach

To the dynamic approach, it was used the local search proposed so that the additional information during the working day could be treated. This way, the additional information can come from the agents or from the services (patients).

In the case of the agent, this can become unavailable to the realization of the work and attendance of the remaining services. Therefore, the associated services to this agent would be available to relocation, i.e. will be treated as reinsertions and will be allocated to other routes or new agents.

In the case of the service, three possibilities will be studied: service cancelation, this will be removed and, in case it is associated to a route, it will be recalculated and the agent will be alerted; edition, in case of edition the previous service will be cancelled (treated as the previous case) and the edited service will be insert; service insertion, in case of insertion, the local search will look for a route to the service be associated and the agent will be alerted. In case a route cannot be found, a new one will be created and the service will be associated to a new agent.

## 3.6 Summary

In this chapter was realized a discussion regarding the algorithm used, and its possible variations and how the different parameters involved affect the its function and results. Initially a small description about the genetic algorithm was done, the canonic model was exposed and the impacts of its application on the proposed problem were discussed. Then, some variations to each step was proposed, and, when possible, at least two possibilities to a specific operation of the genetic algorithm was searched, allowing a comparative analyses of the results.

At the end of the first section of this chapter, the local search proposed and the functioning of the genetic algorithm used were described. The idea is the combined use of this both techniques, to maximize the efficiency of the search for good solutions. At the second section of the chapter, it was presented analytical results regarding the proposed algorithm. A description of the conditions – machines, base of tests, etc. – was given. Each functionality of the algorithm was then tested separately, with many combinations of parameters, aiming to search for the best combination of algorithms and parameters, to achieve better results and performances. With these simulations, some initial expectations were reviewed, in specific the total travelling time that tends to increase with better

solutions. This happens due to the increase of nodes visited by an agent. On the other hand, the number of agents needed and the waiting time decrease, as predicted during the modeling of the problem. Even so, it is important to maintain the objective of minimize the travelling time, to avoid that this time increases beyond the needed.

The results confirmed the viability of the use of the proposed algorithm in the original problem. In some cases, it was possible to obtain results very close to the optimums (and even equals to the optimum values). Using the data obtained during the simulations, the final combination adopted is described on Table 3.39 below.

| **Parameter** | **Configuration** |
|---|---|
| Population size | 100 |
| Evaluation Function | All analyzed variables with same weight 1 |
| Selection | Proportional Selection |
| Crossover | Crossover OX |
| Mutation | 5% - Removal-and-Reinsert |
| Elitism | 10% of size population |
| Local Search | YES |

**Table 3.39 –** Final configuration to be used in the sub-system algorithm

# 4 Web and Mobile Sub-systems

## 4.1 System objective

For the verification of the genetic algorithms operation on systems of Home Care management, it was developed a system that has features, which the routing genetic algorithms can be used to home care, besides the complementary features to guarantee the functioning of the system.

## 4.2 Methodology adopted

The system was developed on components specifically designed in order to maintain the concept of modularity, being able to substitute anyone for another analogous. It was specified tree modules: Web module, Mobile module and Algorithm module. With the aim of support to these tree modules, it was added an infrastructure management function (considered as a fourth module to this project).

## 4.3 Scenarios of use of the system

### 4.3.1 Scenarios of Web sub-system

#### 4.3.1.1 Scenario 1 – First use

The user connects to the system and visualizes an empty list of agents and an empty list of positions to be visited. The user registers the address and other data needed at the base. The user, then, adds agents and clients to be visited, uploading the necessary information.

After the inclusion of data, the user activates manually the execution of the algorithm that defines the routes. The algorithm indicates that the solution was founded and returns to the agents list.

#### 4.3.1.2 Scenario 2 –Traced Routes Visualization

The user connects to the system and has the access to the agents list, previously registered, and a list of patients. The user selects an agent and visualizes his route, in case exits (previously calculated). Once the route was visualized, the user selects a client and visualizes the information, on real time, regarding the selected client.

#### 4.3.1.3 Scenario 3 – Manipulation of Users

The user connects to the system and edits the registered information of an agent and some clients to be visited. The user deletes the registered information of a visit agent and of some clients to be visited. The user adds the information of a new visit agent and new clients to be visited. The edition, removal or inclusion, is treated in real time if there is the need of modifying the agents routes.

#### 4.3.1.4 Scenario 4 – Service Scheduling

The user connects to the system and has access to the patients list. He chooses the option of scheduling of services of attendance and tries to schedule a service for a day on which the patient has already a scheduled service. The system notifies that is not possible to schedule a visit on the day required. The user, then, schedules the visit for a day, on which, there is no other service for the same patient. The system returns a message of success.

### 4.3.2 Scenarios of Mobile sub-system

#### 4.3.2.1 Scenario 1 – Regular Use

The registered agent connects to the system with username and password, requests the transmission of route data regarding the day. The data will be carried on the memory of the equipment. The route will be displayed on the device, the patients information are received when clicking on each node. The agent sends the visit status (ok/not ok), as an alternative, more concrete data about the procedures can be sent.

#### 4.3.2.2 Route Update

The agent receives on his mobile device a warning that a new route is available, he, then, requests an update of the route information, the new route is finally loaded to the mobile device.

#### 4.3.2.3 Scenario 3 - Anomalies

Unable by adverse reason of continue his visits during the day, the agent selects the option Anomaly. The mobile sub-module sends to the web sub-module a signal that the agents cannot continue, starting a procedure of route recalculation, where the patients that were assigned to this agent are redistributed to the others.

#### 4.3.2.4 Scenario 4 – Sending reports

After the realization of a visit, or its non-realization, the user chooses the option corresponding to the report creation, a screen is shown enabling the entry of data regarding the visit or not. The report created is then sent to the web system.

## 4.4 Function requirements

### 4.4.1 Web sub-system

1. Agents management: The sub-system should register the agents, as well as edit their information, disable them (in case they cannot realize the service), reactivate and remove them from the system;
2. Patients management: The sub-system should register the patients, as well as edit their information, disable them (in case they are not available for the service) and remove them from the system;
3. Self-service service management: The sub-system should schedule the services, as well as edit its information (including rescheduling) and remove them from the system (in case of cancelling);
4. Agents list and data consultation;
5. Patients list (complete list and scheduled Patients for one Agent) and consultation of his data;

6. List of self-service service (including services already realized – history) and consultation of data;
7. Manual calculation of route;
8. Route visualization for each Agent graphically on a map;
9. Distinction between different types of Patients on the map, according to his service situation, necessity of route recalculation and if the same is active or not.

### 4.4.2 Mobile sub-system

1. User validation – necessary for access to the app;
2. Route loading: the sub-system realizes the download of the route (with the order of the patients to be visited) calculated for the user associated to the app in execution;
3. Visualization (graphically on a map) of the route allocated to an Agent associated to the app;
4. Visualization of the patients information;
5. Sending the information about the care Service realized with success or not (feedback);
6. Sending the visit Report from each Patient visited;
7. Sending the notice of self-deactivation, in casa the Agent cannot complete the route.

## 4.5 Non-function requirements

### 4.5.1 Usability and Performance

The Mobile system should have a fast response time, because it is used on field by the agents. This time is subjected to the internet speed of the mobile devices and to the data plan hired by the company that uses the system. The controls should be of easy use (big buttons). The Web system, because it is accesses by an unique manager of the system, has exclusively the commitment of being fast (again according to the internet speed used).

### 4.5.2 Availability and Reliability

The database should be available to the agents, as well as the algorithm module, due to the constant need of the agent to download the route, send feedbacks and reports and due to the dynamic module in case there were unexpected changes on the routes during the day. The database should have consistent information and its operations should be atomic, due to the fact that the agents can access the route information at any instant. Therefore, at the moment of update of these data, the changes of each route should be released at the same moment.

### 4.5.3 Security and Limitations

The Mobile sub-system should have the minimum security of validation of user. However, for the Web sub-system this need doesn't exist since it is accessed by a unique manager/supervisor of the system. The entire system (including the Web, Mobile and Algorithm sub-system) should always have access to internet, without which there is no possibility of access to the API of Google Maps, preventing then the visualization of maps, location of point at the map, request of address data (geocoding), assembly of routes and exchange of information between sub-systems.

## 4.6 System specifications

### 4.6.1 System architecture

### 4.6.2 Classes diagram

#### 4.6.2.1 Web Sub-system

The diagram of Figure 4.2 defines the classes of the Web Sub-system:

#### 4.6.2.2 Algorithm Sub-system

The diagram of Figure 4.3 defines the classes of the Mobile Sub-system:

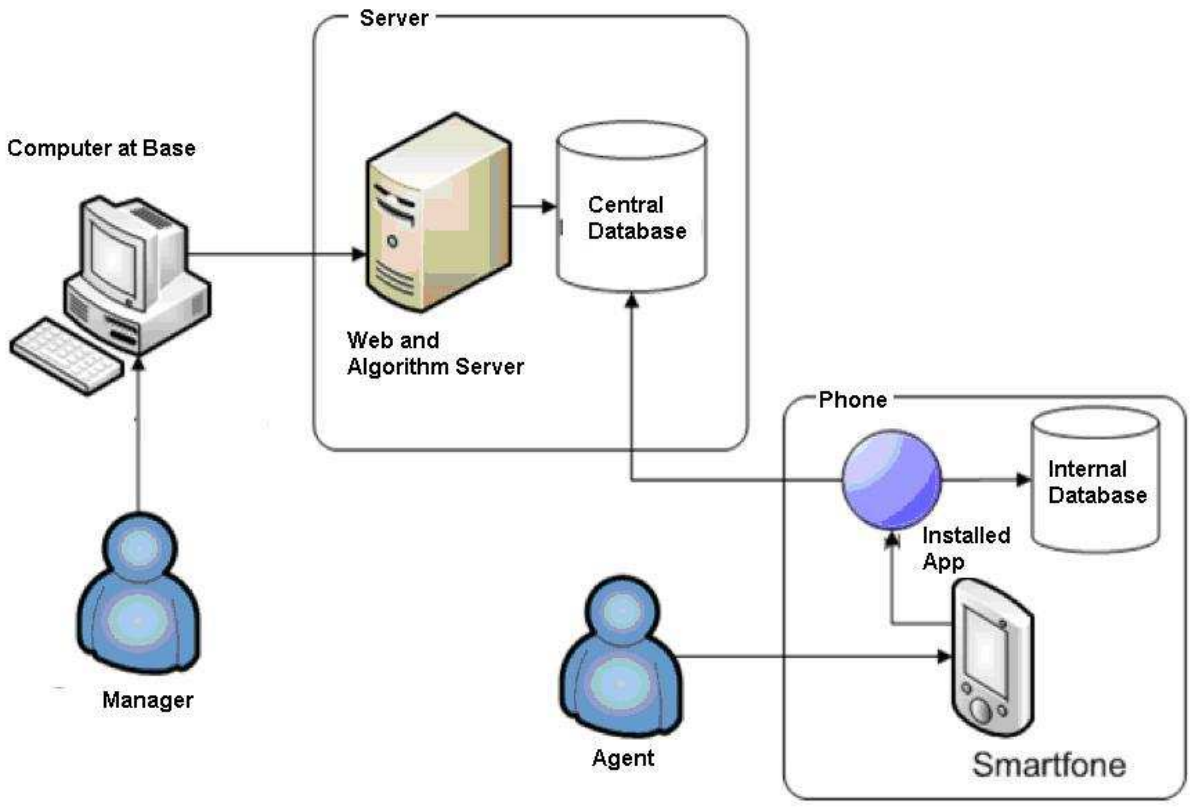


**Figure 4.1 –** Architecture of the Complete System

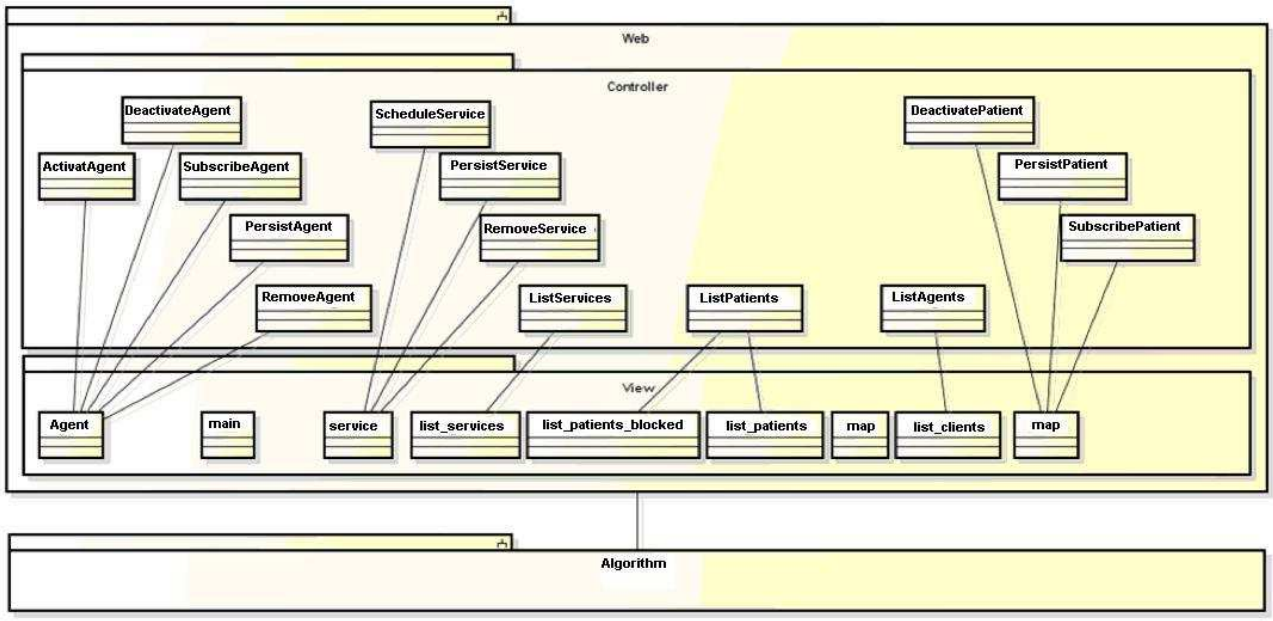


**Figure 4.2 –** Classes diagram of the Web Sub-system

#### 4.6.2.3 Algorithm Sub-system

The diagram of Figure 4.4defines the classes of the Algorithm Sub-system:

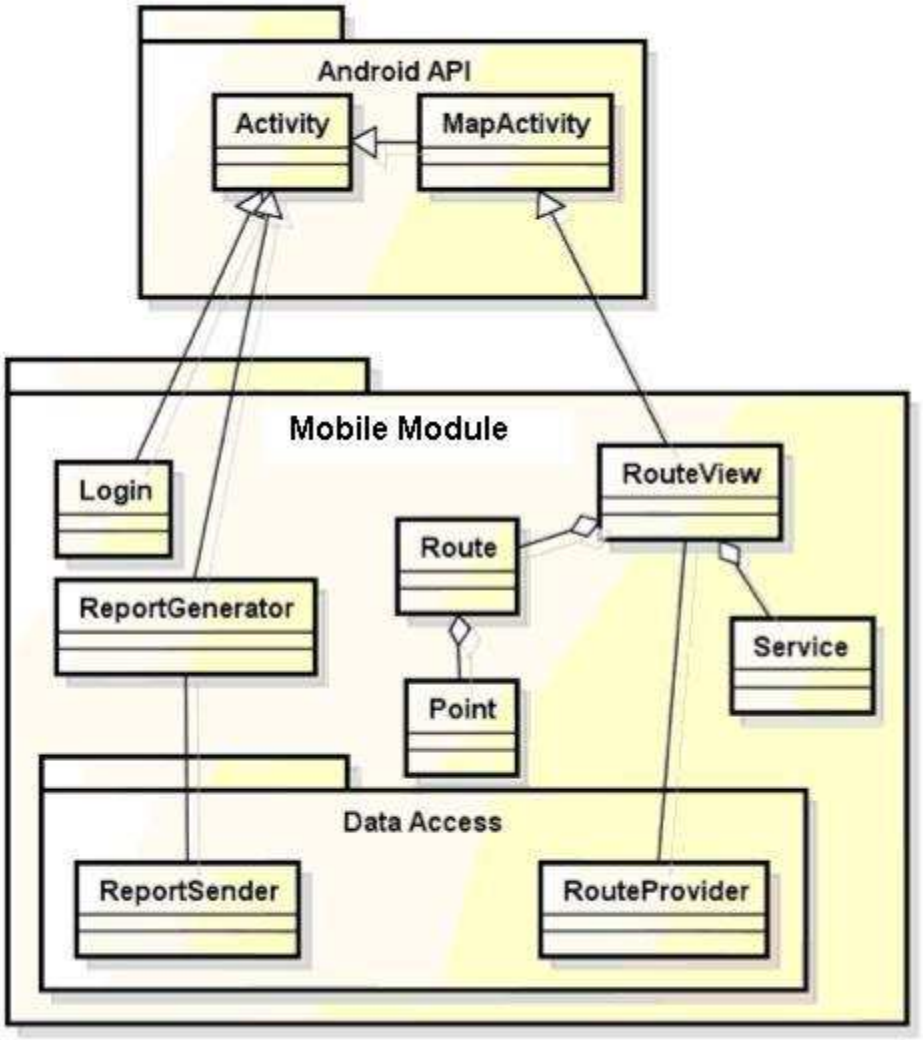


**Figure 4.3 –** Classes diagram of the Mobile Sub-system

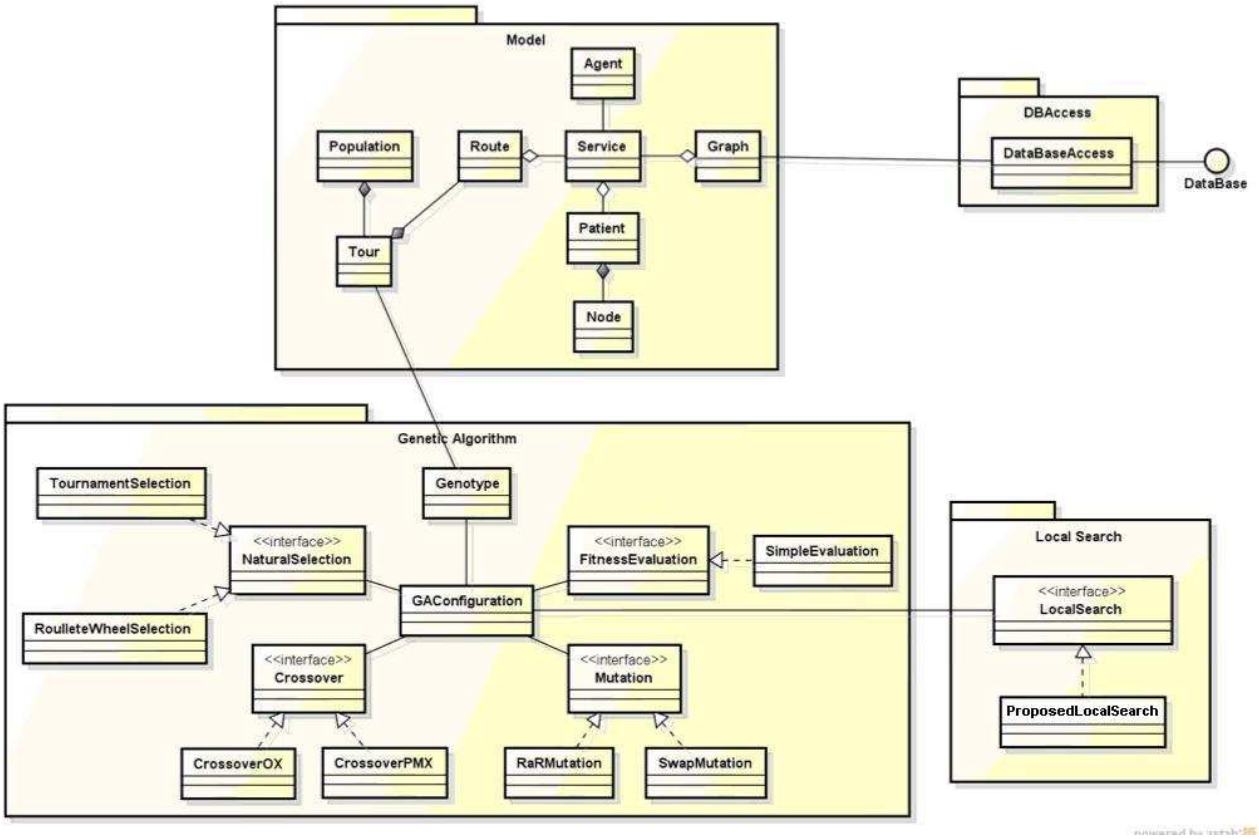


**Figure 4.4 –** Classes diagramof the Algorithm Sub-system

### 4.6.3 Database Model

On Figure 4.5 and Figure 4.6 are illustrated the Entity Relationship Diagram (ERD), with focus on its entities and its relationships, and the database scheme, on which is possible to analyze the entities attributes.

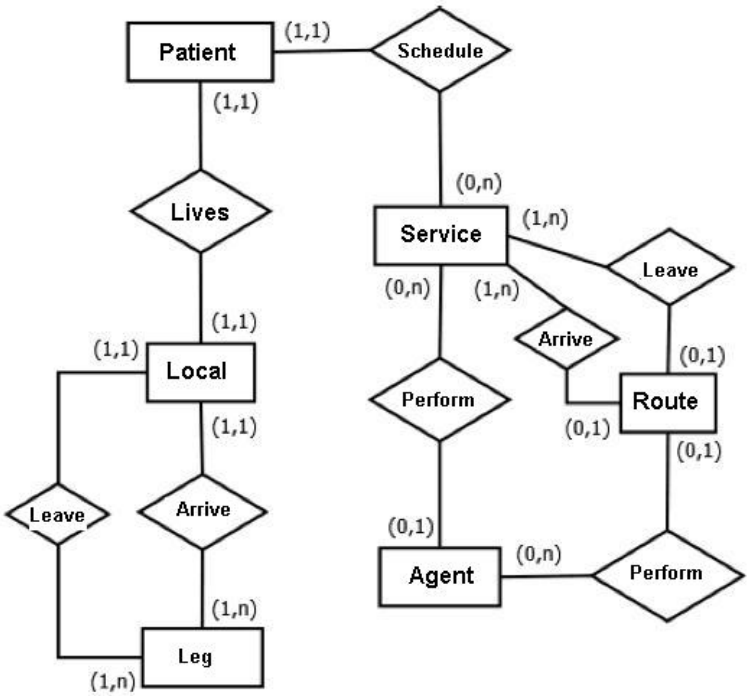


**Figure 4.5 –** Entity Relationship Diagram (ERD) of the project

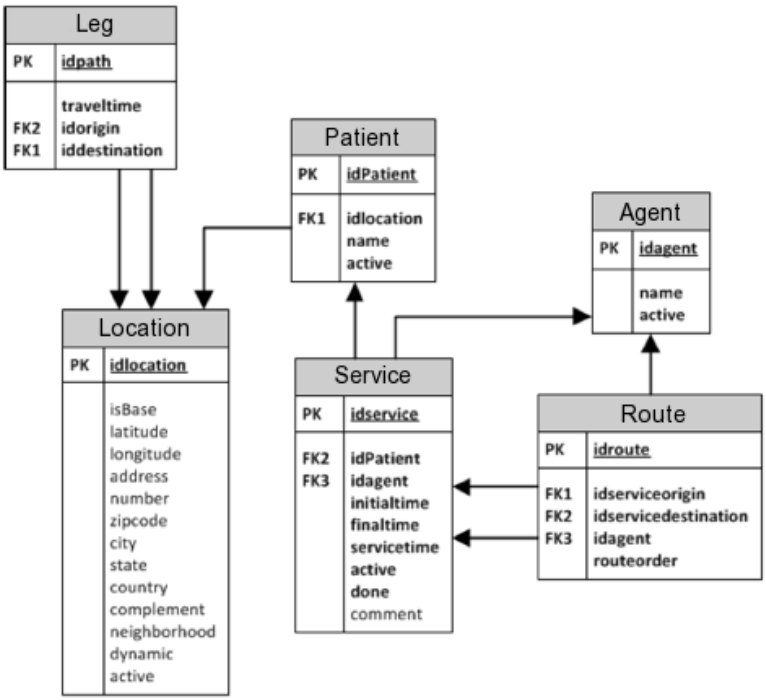


**Figure 4.6 –** Scheme of the database of the project

## 4.7 Implementation

### 4.7.1 Infrastructure

A server was provided at the Software Factory Laboratory from Escola Politécnica, running Windows 7. The manager SystemManagerDatabase used was MySQL and, to realize the versioning of files and documents, it was used the SVN. Both services were hosted on the server so that these services were available through internet. It was necessary to have installed a client software, to access the database and to use the versioning. For the access of the base it was used the dbForge and for the versioning it was used the Tortoise SVN.

### 4.7.2 Technology

The Web sub-system was developed in Java, using the appliance Eclipse (version Helios) with server Apache-Tomcat (version 6.0). Additionally, there was an application programming interface (API) from Google Maps for the maps and routes visualization and obtaining the address data (including latitude and longitude). The Mobile sub-system was developed in Java, using the appliance Eclipse (version Helios), using a development tool for the Android platform. The mobile devices and the emulator used to the tests use the operational system Android and this also has access to the API from Google Maps. All the development had as bases the version 2.1 (Éclair) from the operational system Android, this was the stable version of the system at the initial moment of development. By uncertainties in relation to the compatibility, the version of the operational system was maintained, even after the launch of the new version. Other than that, the integration between the mobile module and the database at the server was done through the scripts PHP running on protocol HTTP, for security issue, for the integration with the maps server (Google Mas). It was also used the protocol HTTP, with information reception in XML. The algorithm sub-system, both static and dynamic, was also developed using language Java and appliance Eclipse (version Helios). The code developed was distributed as a file .jar to the Web sub-system, providing access to the classes and methods necessary to the use of the routing functionalities and entities from the database.

## 4.8 System screens

### 4.8.1 Screen results – Web Sub-system

#### 4.8.1.1 Agents

Figure 4.7Figure 4.8,Figure 4.9 and Figure 4.10 show the health agents in the system:

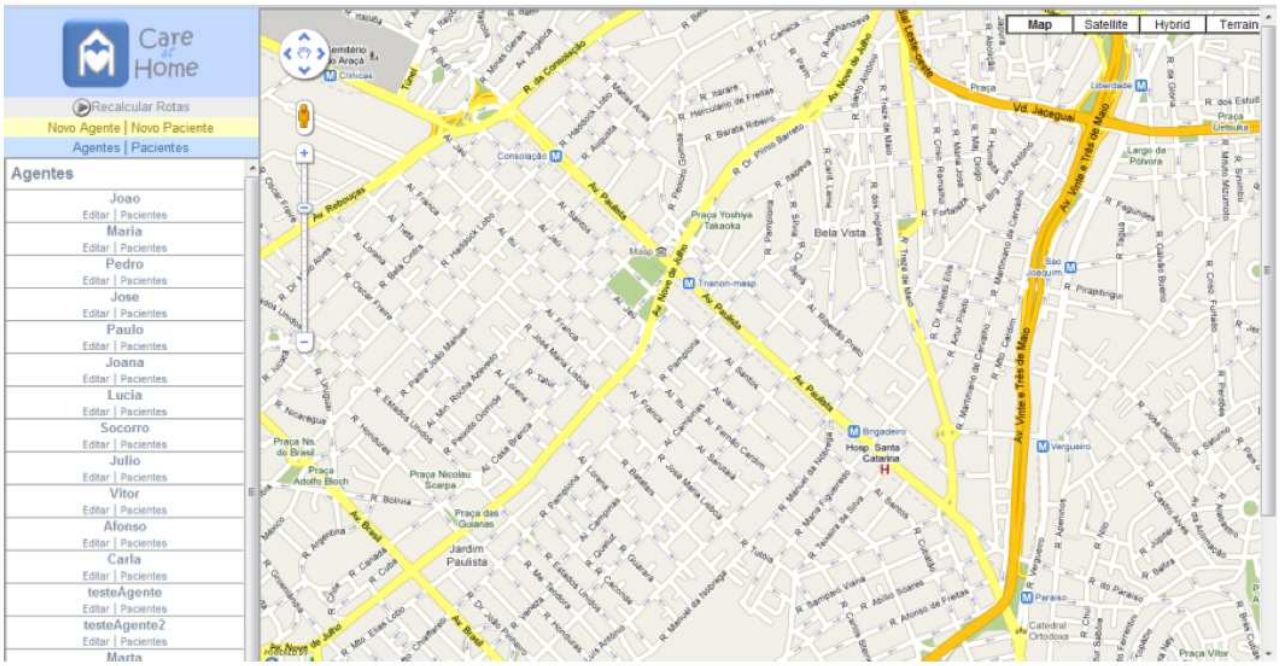


**Figure 4.7 –** Screen 001 – List of agents

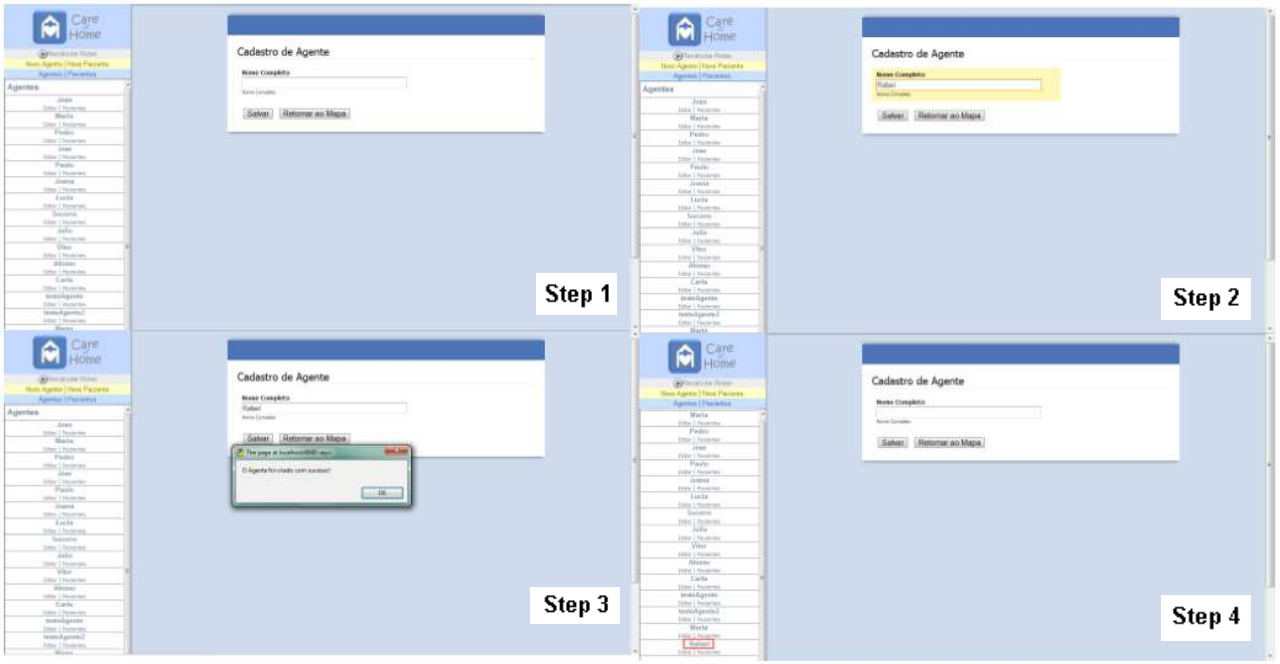


**Figure 4.8 –** Screen 002 – Agent creation steps

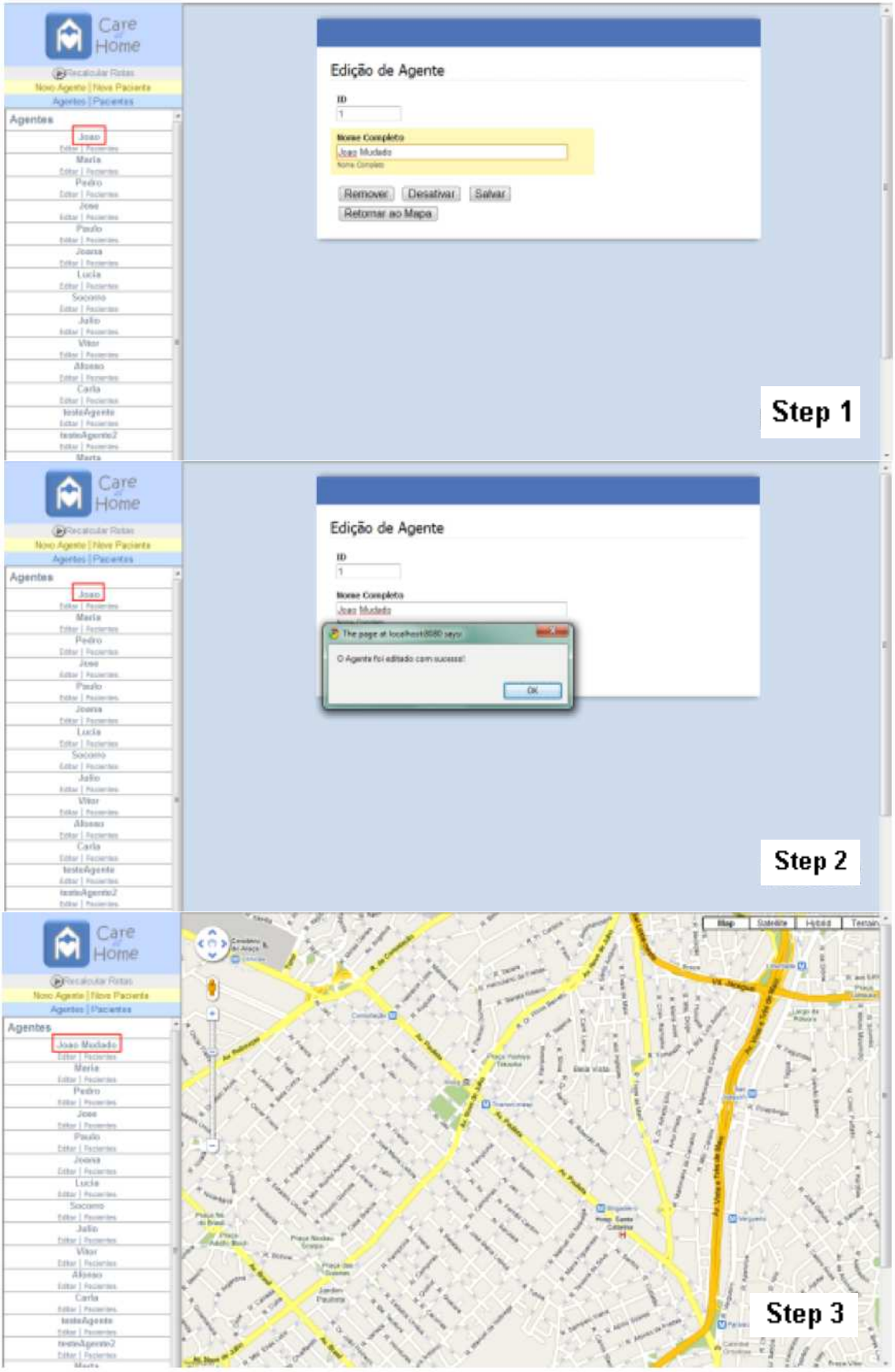


**Figure 4.9 –** Screen 003 – Agent edition steps

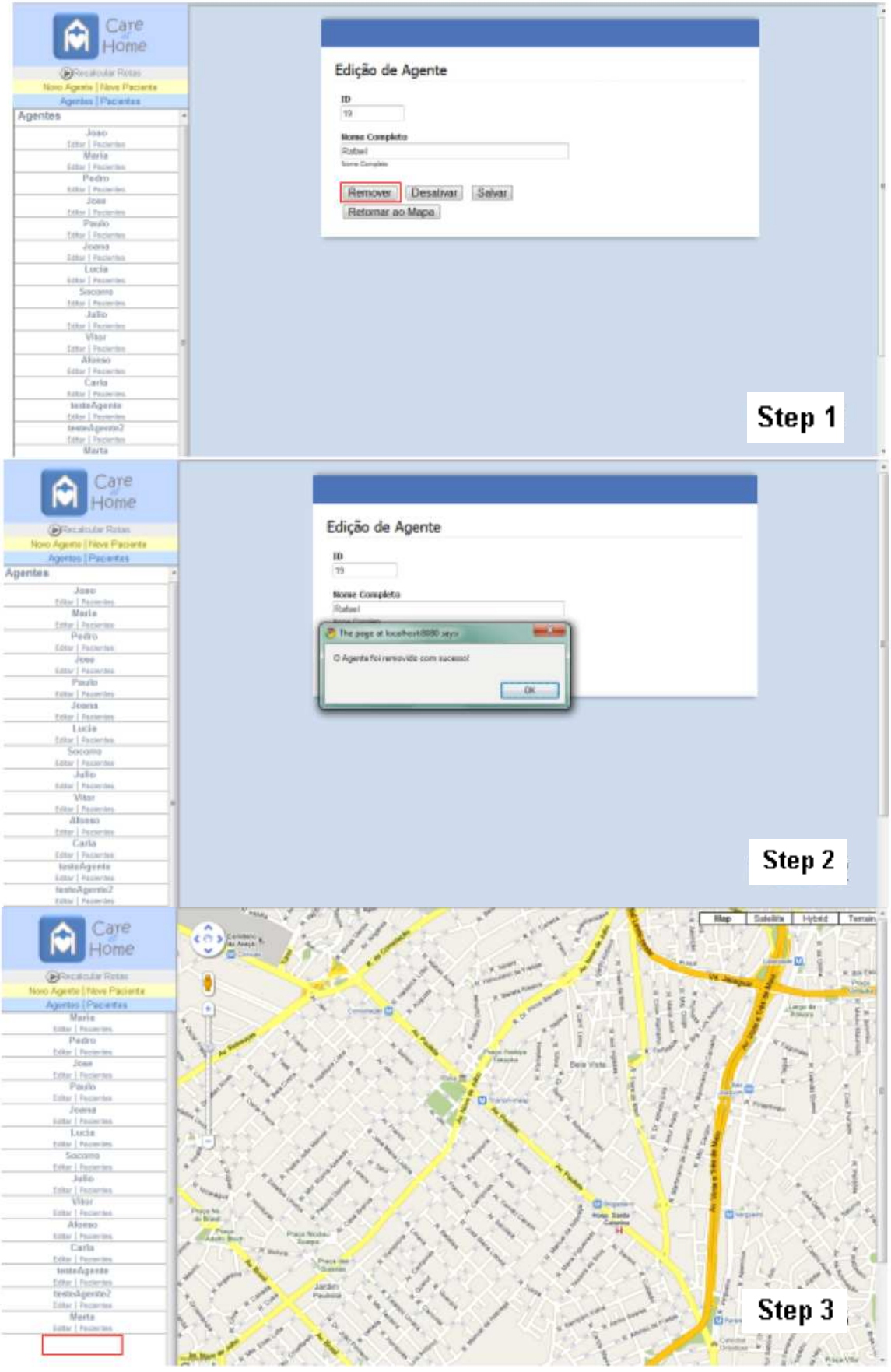


**Figure 4.10 –** Screen 004 – Agent removal steps

### 4.8.1.2 Patients

Figure 4.11,Figure 4.12 and Figure 4.13 show the patients in the system:

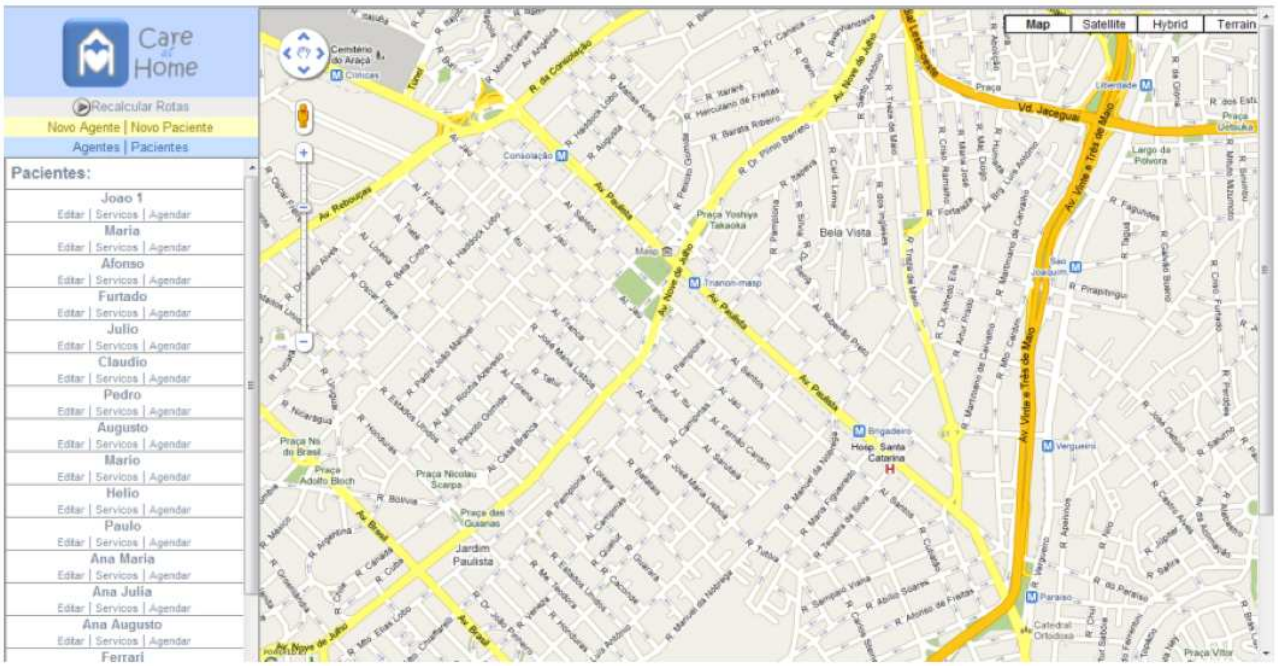


**Figure 4.11 –** Screen 005 – List of patients

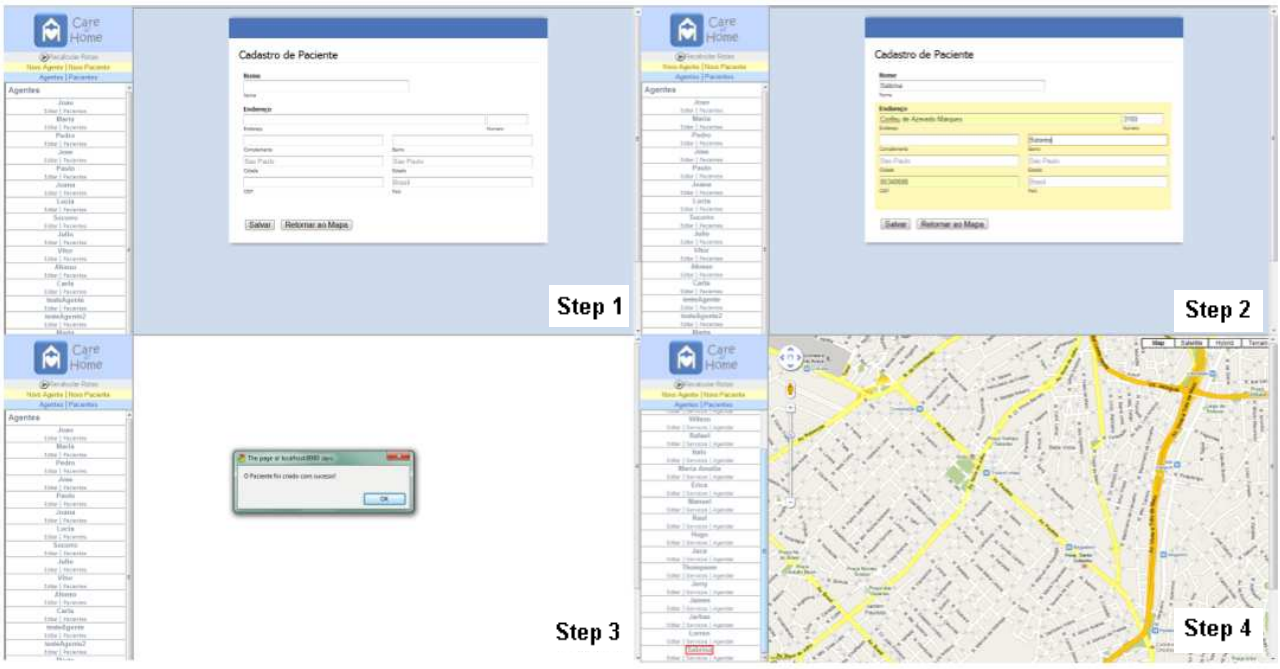


**Figure 4.12 –** Screen 006 – Patient criation steps

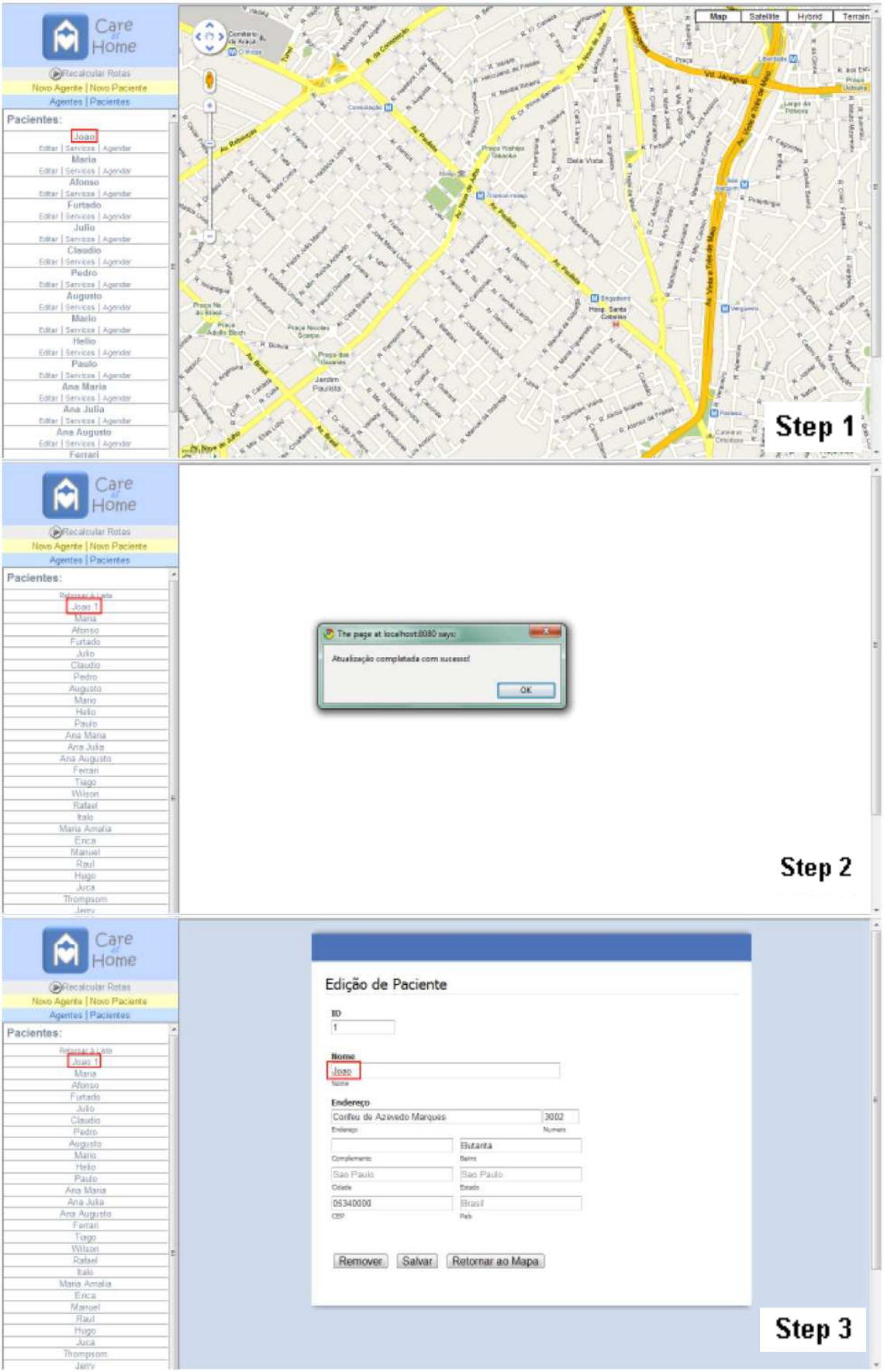


**Figure 4.13 –** Screen 007 – Patient edition steps

#### 4.8.1.3 Services

Figure 4.14, Figure 4.15, Figure 4.16 and Figure 4.17 show the customer service in the system:

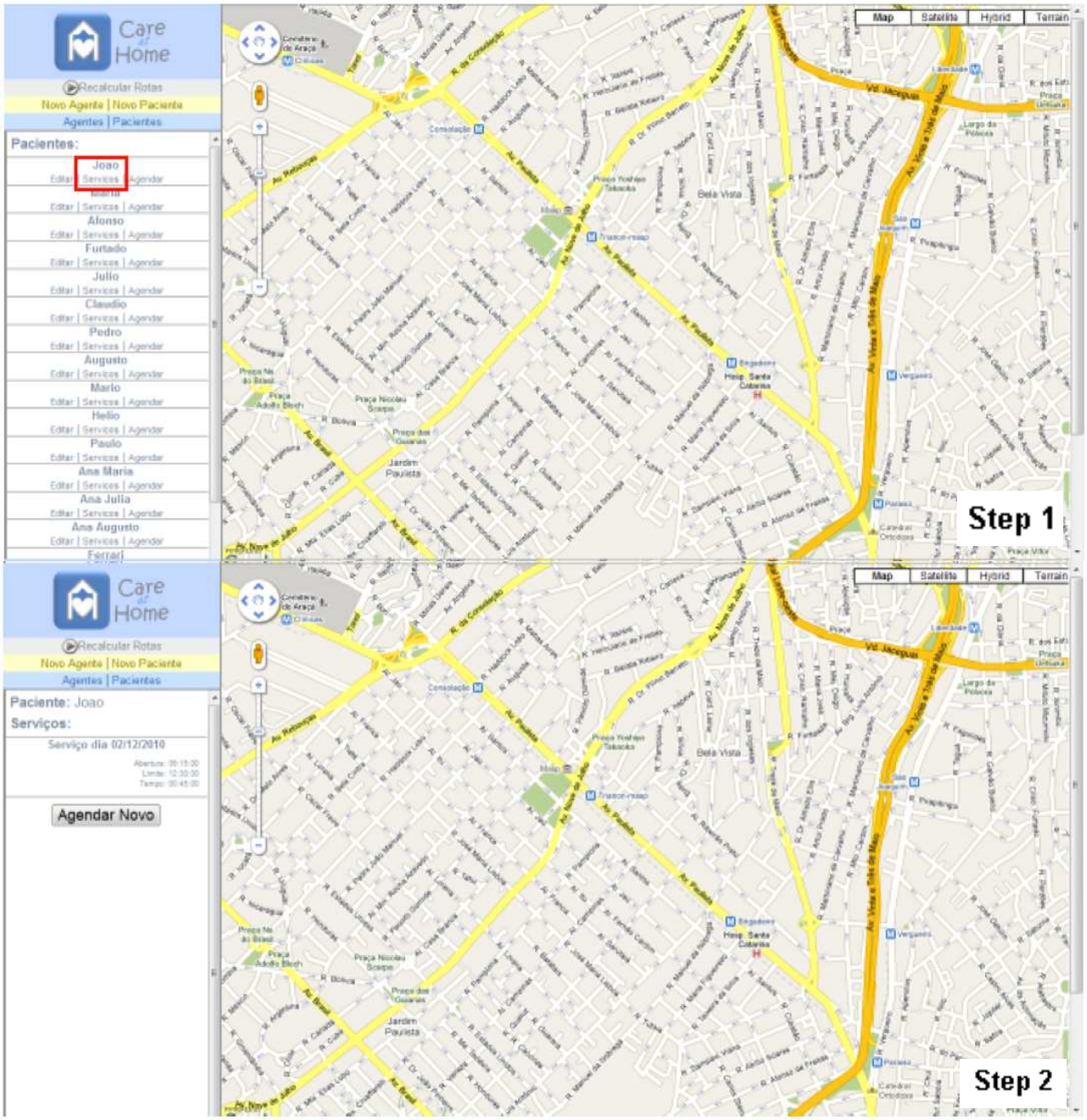


**Figure 4.14 –** Screen 008 – List of patients services

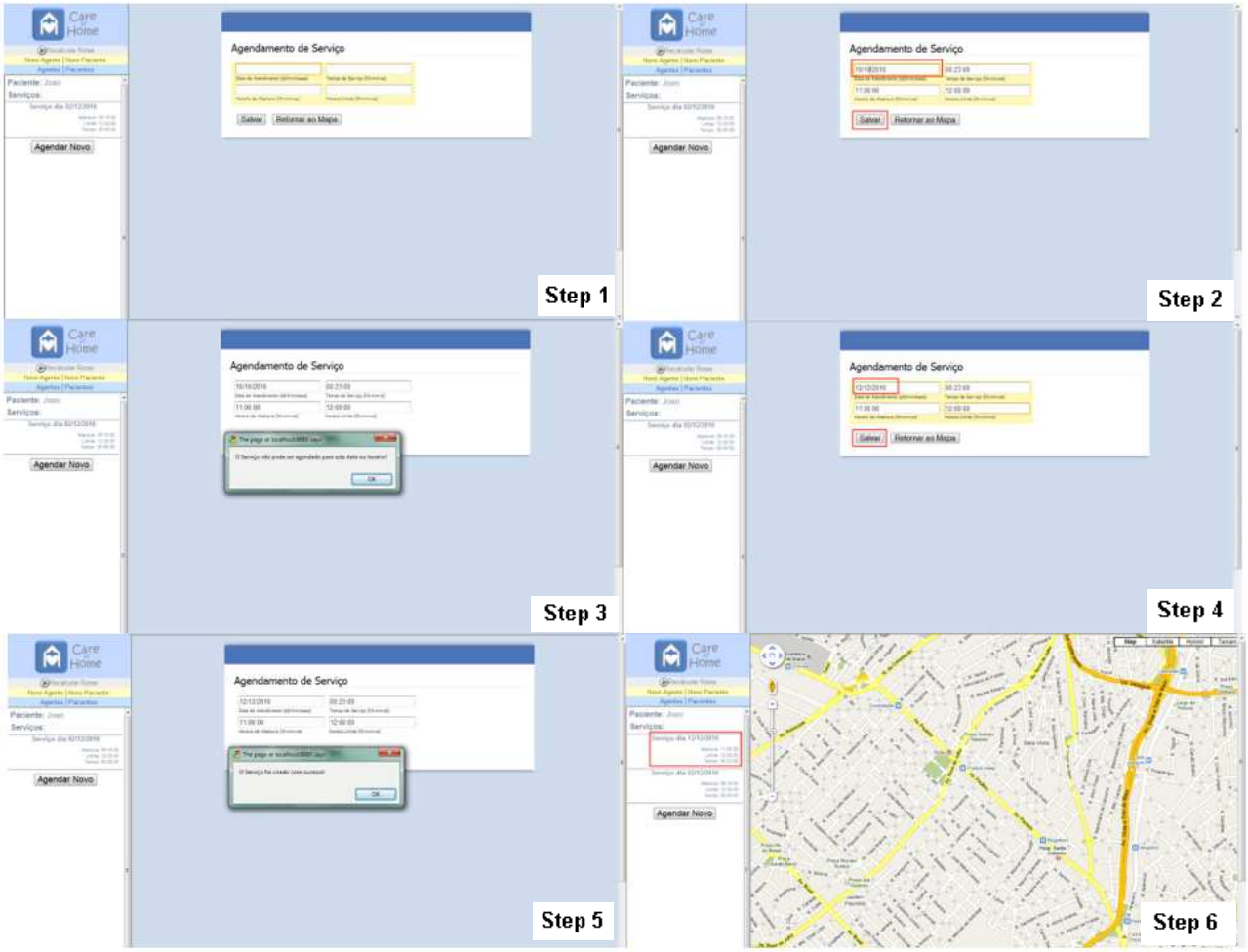


**Figure 4.15 –** Screen 009 - Service creation steps

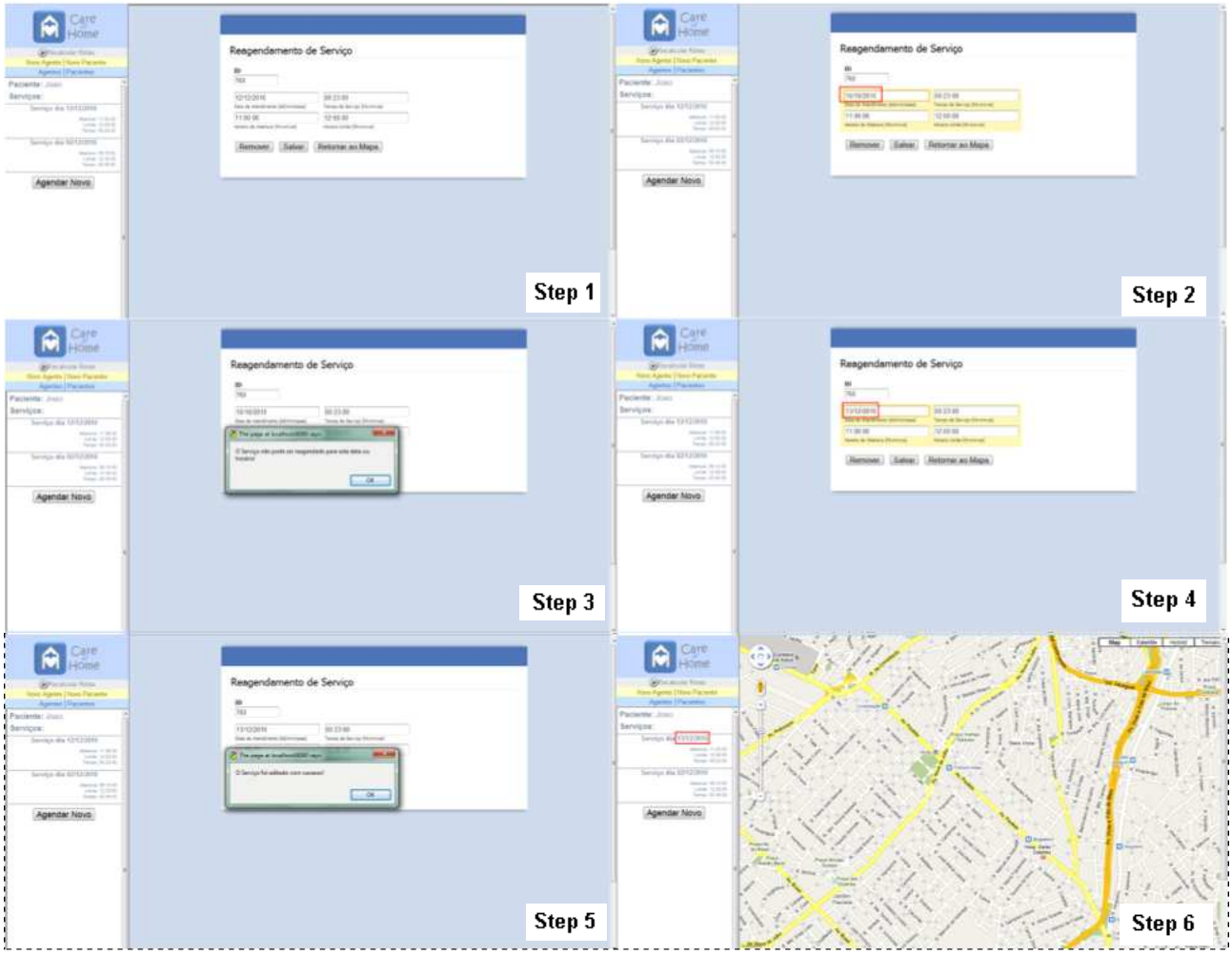


**Figure 4.16 –** Screen 010 – Service edition steps

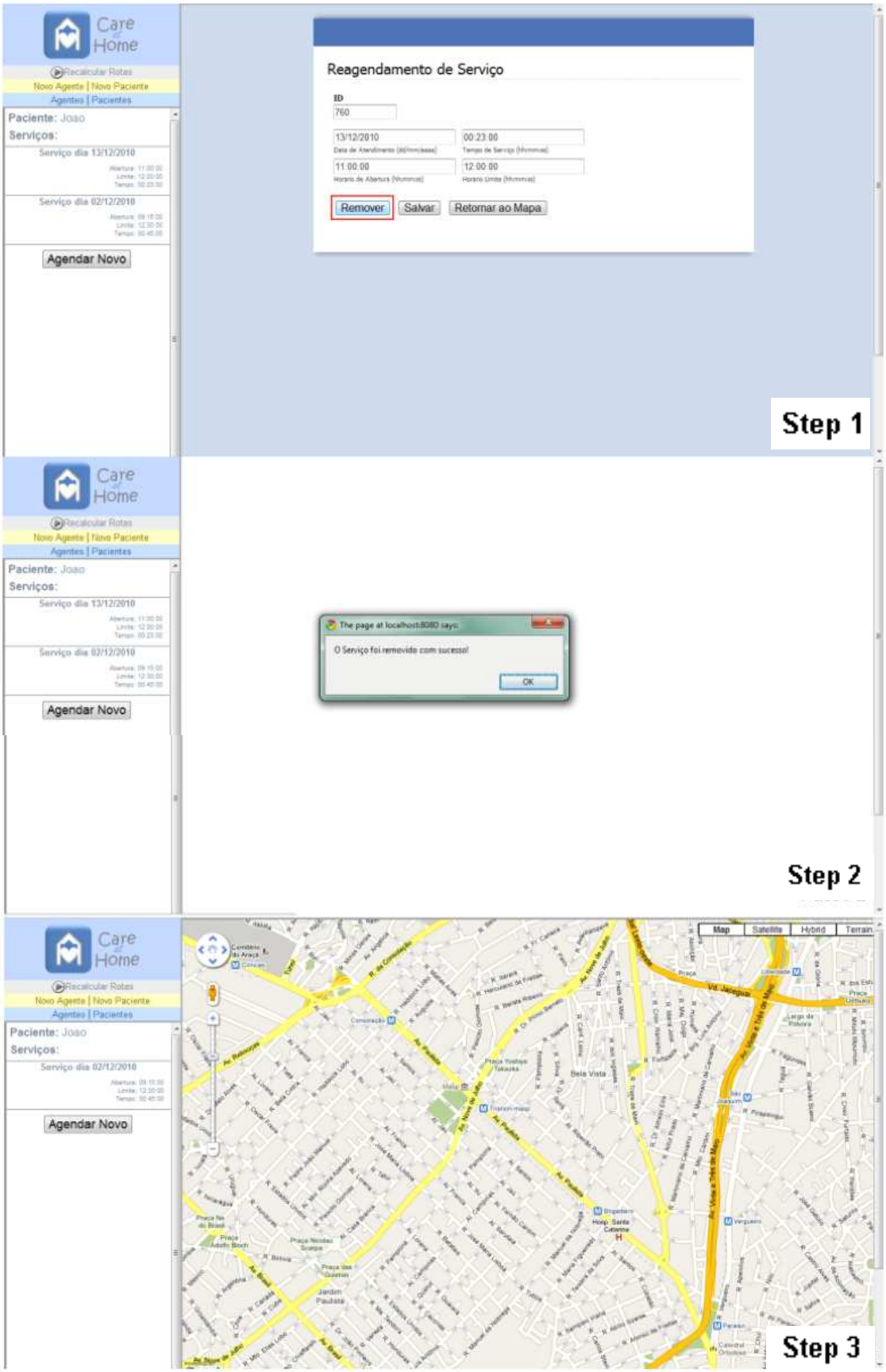


**Figure 4.17 –** Screen 011 – Service removal steps

#### 4.8.1.4 Routes

Figure 4.18, Figure 4.19, Figure 4.20 and Figure 4.21 show the routes in the system:

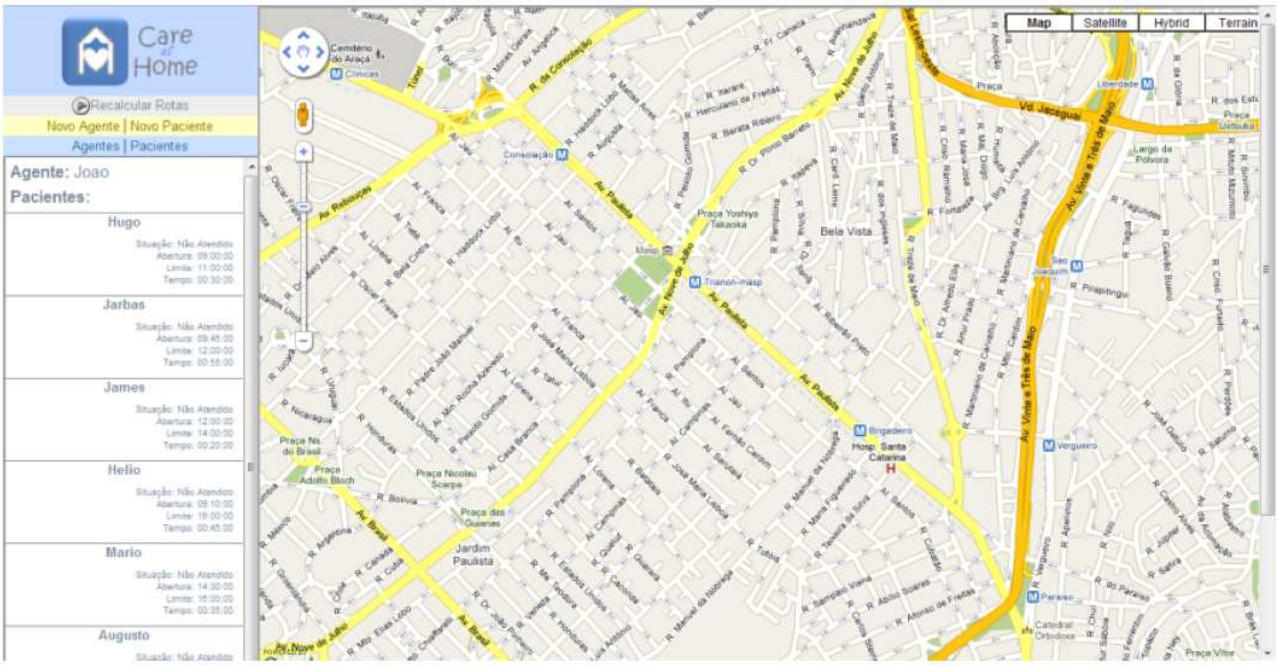


**Figure 4.18 –** Screen 012 – List of patients per agents

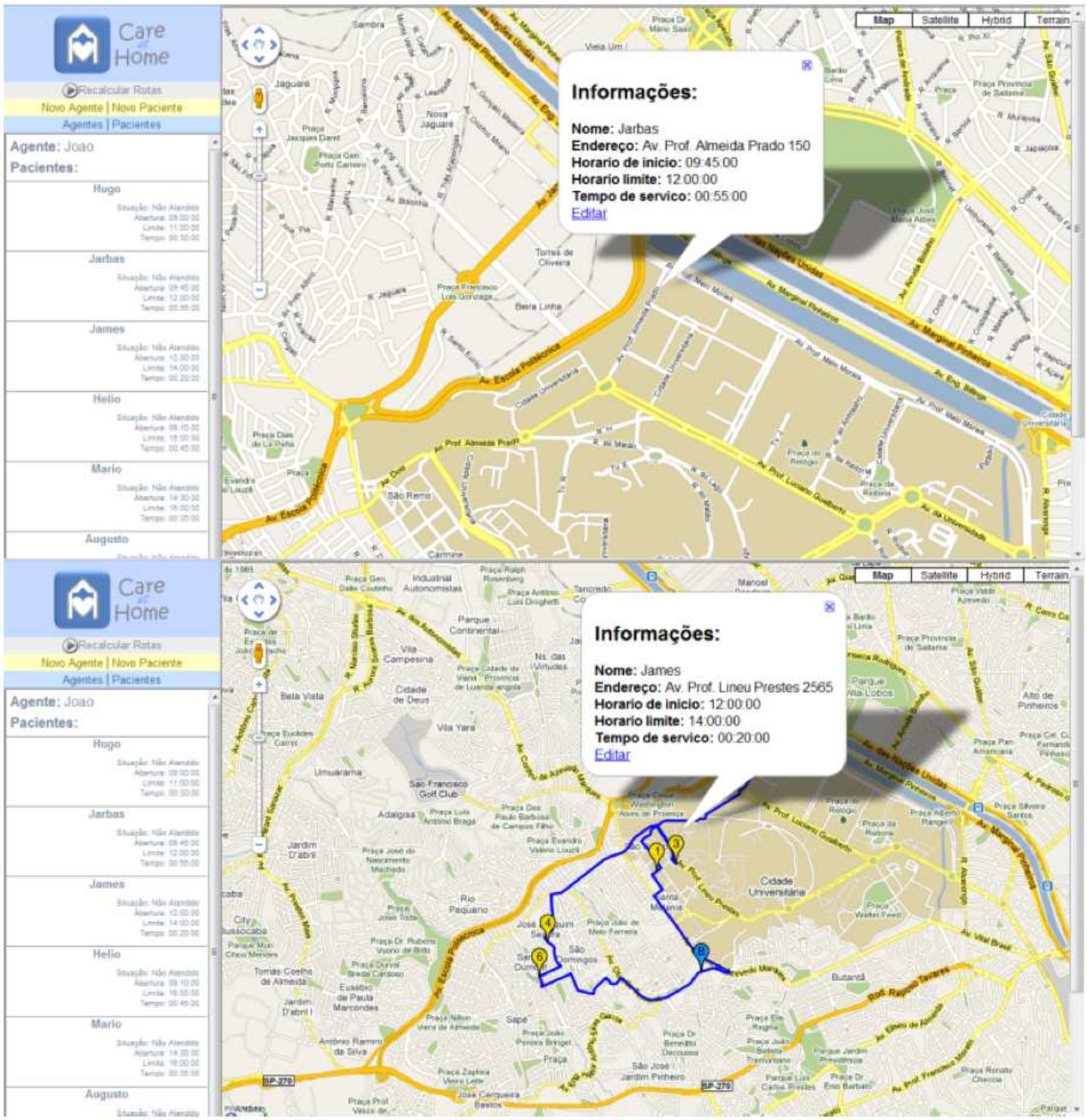


**Figure 4.19 –** Screen 013 – Service in route or not visualization

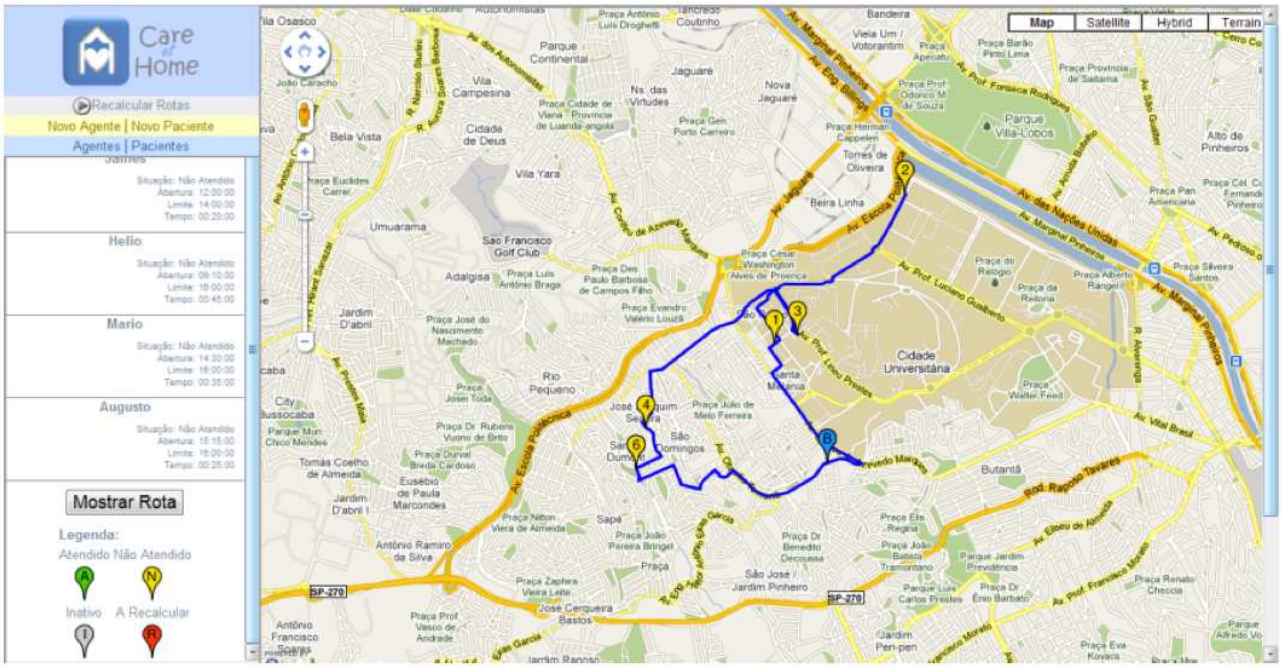


**Figure 4.20 –** Screen 014 – Visualization of route and legend

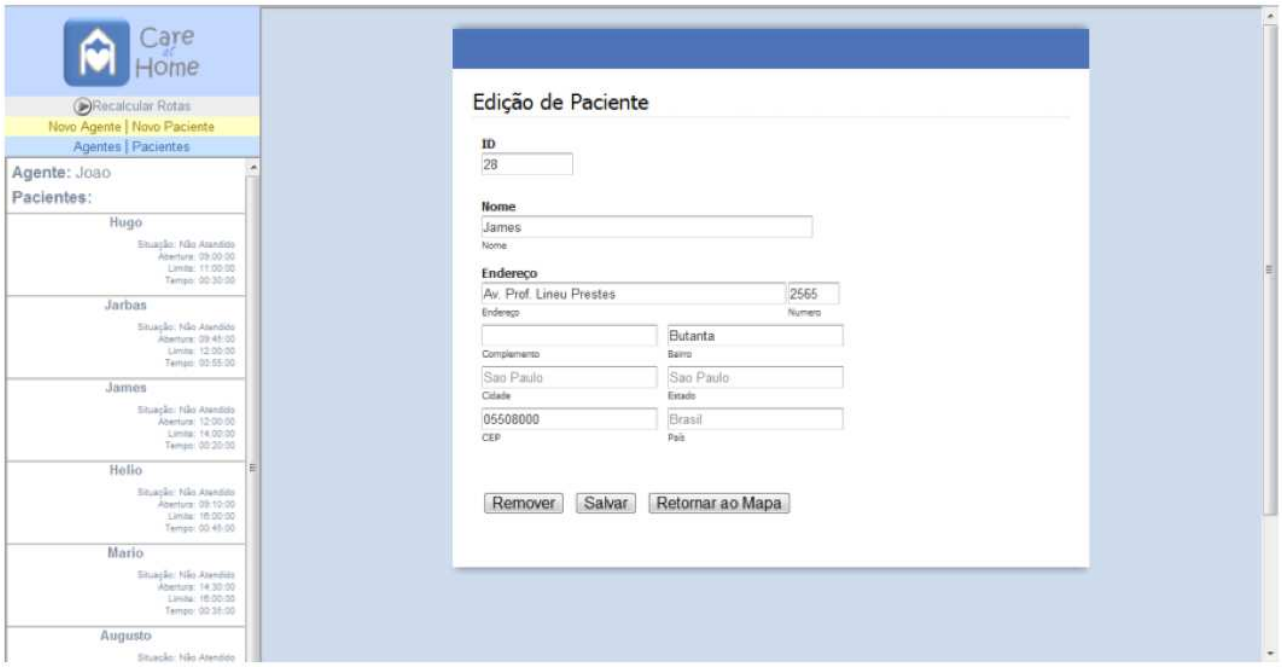


**Figure 4.21 –** Screen 015 – Description of location

### 4.8.2 Screen results – Mobile Sub-system

#### 4.8.2.1 Regular use of the system

Figure 4.22 andFigure 4.23 show an example of use of the system:


**Figure 4.22 –** Screen 016 – Authentication of the agent

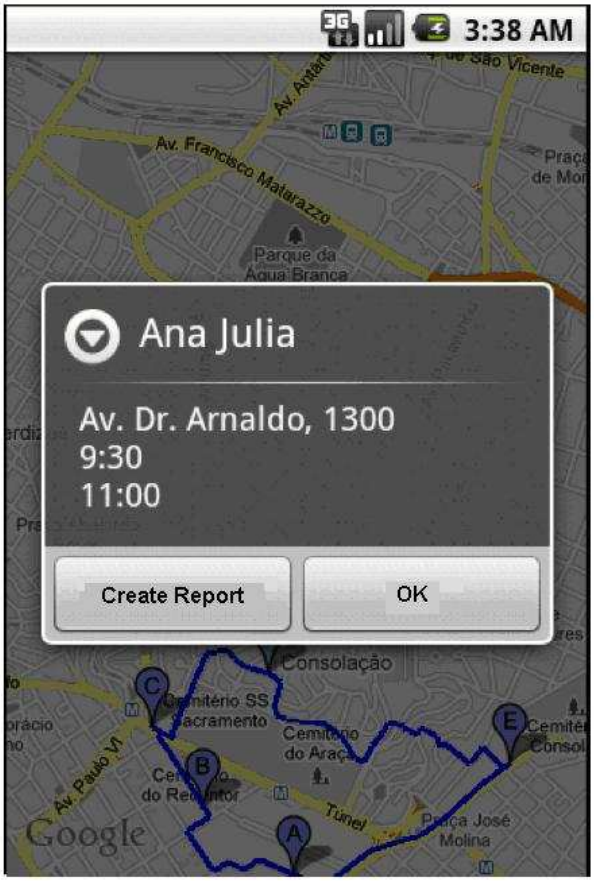


**Figure 4.23 –** Screen 017 – Visualization of patient information

#### 4.8.2.2 Route update

Figure 4.24 shows an example of route update:

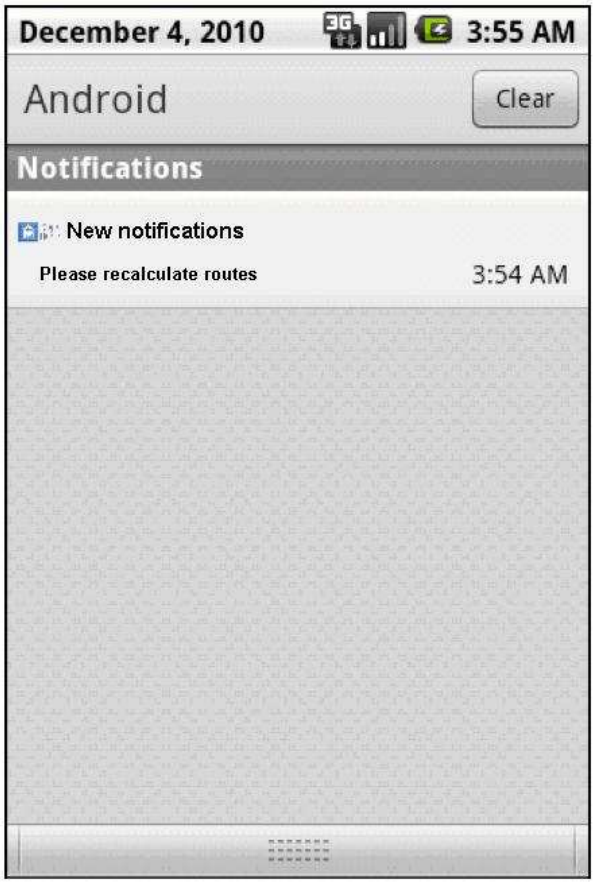


**Figure 4.24 –** Screen 018 – Notification of existence of new route information

#### 4.8.2.3 Anomaly information

Figure 4.25 shows an example of anomaly:

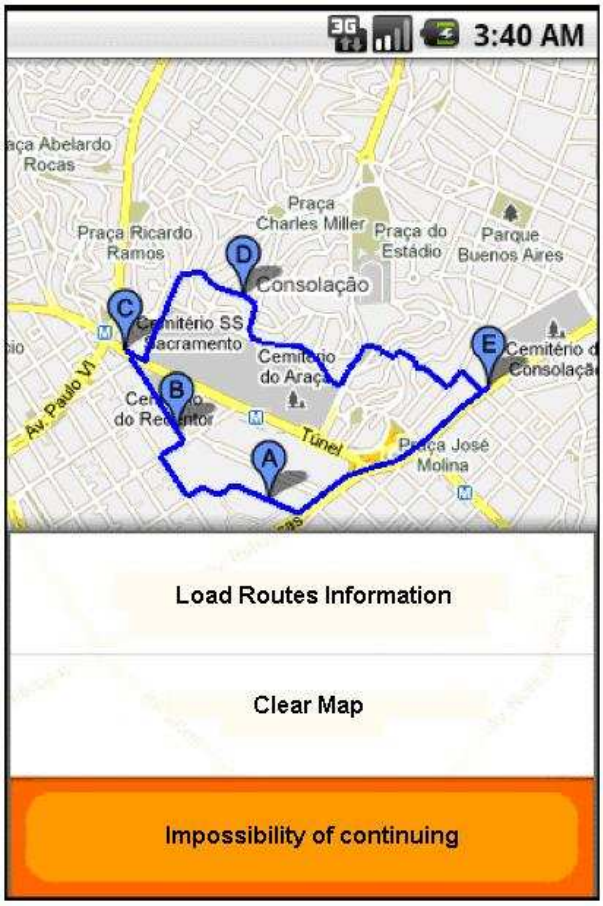


**Figure 4.25 –** Screen 019 – Information of impossibility of agent to continue the visits

#### 4.8.2.4 Report generation

Figure 4.26 shows an example of a visit report:

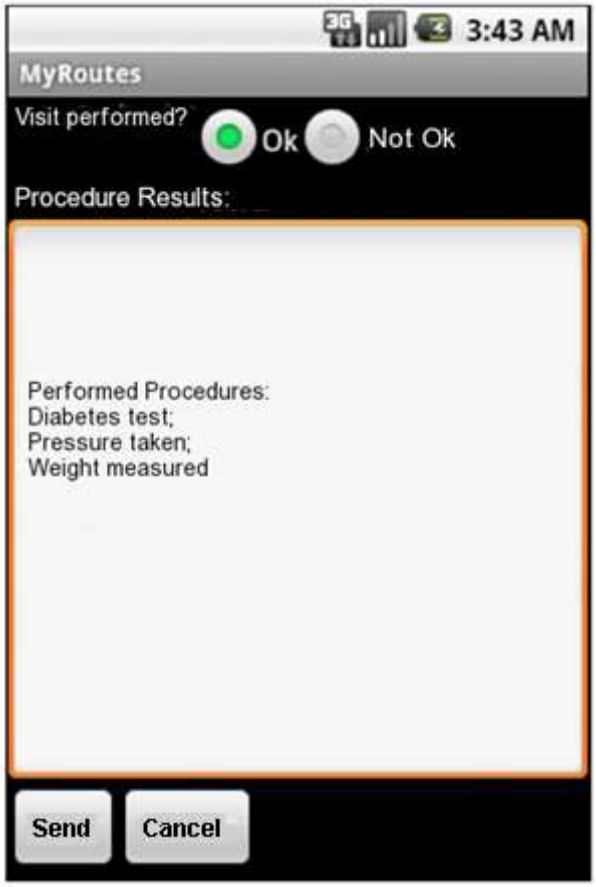


**Figure 4.26 –** Screen 020 – Agent creates the report informing about the visit

## 4.9 Tests and Results

### 4.9.1 Test cases – Web Sub-system

Table 4.1 shows the tests done for the Web Sub-system:

| Tests | Target functionality | Scenarios | Results |
|---|---|---|---|
| Entries | Base, Agent, Patient and Service Entries | - File a base into the Database;<br>- File Agents into the Database;<br>- File Patients into the Database;<br>- File services into the Database. | OK |
| Lists | Agents, Patients and Services lists | - Show Agents;<br>- Show Patient List;<br>- Show Patient List for a specific Agent; | OK |

| | | | |
|---|---|---|---|
| | | - Show Blocked list of Patients;<br>- Show patient Service List (present, past and future) | |
| Patient Status | Patient List | - Show status and service information of a Patient to a designed Agent in the list. | OK |
| Show routes | Route Visualization for an Agent, Differences of Patients by Status | - Show numbered Route, patient base and color differentiation according to Status on the map. | OK |
| Patient Information | Visualization of Patient Information | - Show information window on the map (about Patient geological local) with Patient address data. | OK |
| Patient Edition | Edition of | - Change Patient Name;<br>- Change Patient Address;<br>- In case the Patient Address if of limits from the original local, change status to “Recalculate”. | OK |
| Service Edition | Edition of | - Change date of Service;<br>- Change Service Time;<br>- In case there is another Service for the same date, or this is previously the current date, not allow it. | OK |
| Agent Removal | Agent Removal | - Remove Agent from the Database;<br>- Put all non-visited Patients as “Recalculate”. | OK |
| Patient Removal | Patient Removal | - Remove Patient from the Database. | OK |
| Service Removal | Service Removal | - Remove Service from the Database;<br>- In case the Service date is previously the current date, not allow it. | OK |
| Agent Deactivate | Agent Deactivate | - Deactivate Agent;<br>- Put all non-visited Patients as “Recalculate”. | OK |
| Agent Reactivate | Agent Reactivate | - Activate Agent. | OK |
| Calculate manual route | Calculation of Manual Routes | - Recalculate Agents Routes | OK |

**Table 4.1 –** Functional tests of the Web module

### 4.9.2 Test cases – Algorithm Sub-system

The trials were realized together with the tests described on 4.8.1 Web Sub-system, since it is a module used by the same type of system. However, tests and results rolling theoretical and practical aspects can be found on Chapter 3.

### 4.9.3 Test cases – Mobile Sub-system

The Mobile Sub-system tests were realized to guarantee the module functionality and were based on the collected cases. In cases where the functionality was a start point to a complex series of routines, some unitary tests were realized.

Table 4.2 shows the unitary tests and cases realized, with a small description of scenario that it was conducted.

| Tests | Target functionality | Scenarios | Results |
|---|---|---|---|
| Receiving route information | Integration with Google Maps | - Given two geographic points (latitude and longitude), find the route between them;<br>- Given N geographic points, find the route between the first and last passing by the others. | OK |
| Show route on top of map | Route trace on Android's Interface | - Given any route, show it on the map. | OK |
| Receiving BD route | Integration with BD | - Given an agent name, search on BD the points where he should pass. | OK |
| Marker Color Change | Visual Feedback of report generation | - Change marker color (shown on map) according to the value of a Boolean variable. | OK |

**Table 4.2 –** Functional tests of the Mobile Sub-system

### 4.9.4 Integration Tests

Table 4.3 shows the integration tests done:

| Tests | Target functionality | Scenarios | Results |
|---|---|---|---|
| Visualize Data | - Web: EntryAgent, EntryPatient and EntryService;<br>- Algorithm: DAOAgent, DAOPatient, DAOService, Agent, Patient and Service. | - Receive Agent data;<br>- Receive Patient data;<br>- Receive Service data. | OK |
| List formation | - Web: AgentList, PatientList and ServiceList;<br>- Algorithm: DAOAgent, DAOPatient, DAOService, DAOLocation, Agent, Patient, Service and Node. | - Receive Agents list;<br>- Receive Patient list;<br>- Receive Service list. | OK |
| Save Patient | - Web: PersistPatient;<br>- Algorithm: DAOPatient, DAOService, DAOLocation, Patient, Service and Node. | - Saved information on Database. | OK |
| Remove Patient | - Web: RemovePatient;<br>- Algorithm: DAOPatient. | - Remove patient from database. | OK |
| Remove Service | - Web: RemoveService;<br>- Algorithm: DAOService. | - Remove Service from database; | OK |
| Deactivate Agent | - Web: DeactivateAgent;<br>- Algorithm: DAOAgent, DAOService, Agent and Service. | - Parameter isActive as false on Database;<br>- Services as isDynamic as true. | OK |
| Activate Agent | - Web: ActivateAgent;<br>- Algorithm: DAOAgent and Agent. | - Parameter isActive as true on Database. | OK |

**Table 4.3 –** Functional tests of iteration

## 4.10 Difficulties faced

The main difficulties faced were related to the mobile technology used. The first difficult concerns the routes exhibition on the map, the API Google Maps for Android 2.1 do not allows the direct route exhibition, given two points. Therefore, it was necessary that the information were loaded directly on Google Maps website, which offers the possibility of download the information from an xml file. From this file is possible to extract the relevant route information and to do the manual tracing on the map.

The second difficult refers to the access to the central database. For security reasons on information change, the direct access to the database is not advisable on mobile

devices. The proposed solution uses a PHP script, localized on the server. This script is accessed by HTTP and has direct access to the central database. It then fives, though HTTP, the information related to the route. The same process can be done for the report sending.

## 4.11 Summary

On this section, it was discussed the relative aspects to the implementation of the complete system, including the algorithm described on the previously chapter and access interfaces: one manageable, based on a web platform; and one for the agents, based on a mobile Android platform. Many market technologies were used to the implementation viability; those technologies were detailed during the section. The functionality and scenarios of use were listed and described, as well as the tests realized. On this phase, the tests are related to the functionality and not to the routing results founded, that were evaluated on Chapter 3. Even considering the technological difficulties faced, the implementation was considered successful.

# PART IV

# FINAL CONSIDERATIONS

# 5 Final Considerations

## 5.1 Achieved objectives

The objective initially proposed to develop a system that allowed an automatic calculation of the scheduling and routing of health agents and nurse for patients was accomplished. Moreover, for the development of the system, knowledge in the area of Artificial Intelligence and Operational Research was studied, in order to the modeling problem be realized and to techniques of solution approximation be proposed and developed. However, some aspects, see section 5.2, could not be treated in the desired depth, leaving it for future works.

## 5.2 Future works

Some desirable characteristics of the system had to be disregarded due to the implementation deadline. Those characteristics remain as complementary work to this project. It can be cited in special the non-existence of the agent's specialization and the homogeneity of patients. It is noticeable that in a real case those aspects are key to the function of the real business, and so a practical system should obligatory content these requirements.

Another aspect to be improved is the possibility of preference or dislike of an agent by the patient. It is known that in the medical sector the relation between health agent and patient is a differential on the medical treatment, therefore a system that reinforces this bound is desirable. However, the total link between an agent and a patient would prevent the possibility of distribution optimization of agents by patients, because of that the system should be done in a carefulmanner, given preference to the couples but not its obligation. Other than that, personal problem between patients and agents can occur during the treatment, fact that counts in favor of the exclusion of visit possibility to a patient by an agent.

One more relevant aspect is the preferential service, in case exist a service that demands a faster treatment from the health agent; those should be passed in from of the others. However, should not be part of the system hospital emergency cases, which should be treated in a conventional manner.

Last, the increase of test numbers and performance improvements of the algorithms used during the dynamic execution of the system are relevant aspects to be accounted for.

Especially, for the dynamic execution, it is suggested the adoption of adaptive techniques (38). In the case of the genetic algorithm modeling, it was initially studied the proposals named on (39), however there was no time to finalize it, leaving this line or research as a future work. On the static aspect, even though many and comprehensive tests were realized, not all of the parameters were evaluated, due to the multiple combination possibilities resulting from the options of available operator.

## 5.3 Conclusions

The use of medical home service has been shown to be a tendency on developed countries as a way of decreasing the amount of people in hospitals. The method can be applied to less severe cases, with long treatments and that do not necessitate continue monitoring throughout the day. To those cases, a health agent can visit the patient on his house, giving him better conditions, e.g. more comfort and less expose to hospital infections.

This type of service has been satisfactory on countries like Denmark and it starts to be adopted in Brazil, already with public subsidy. On this project, it was proposed a solution for the optimal allocating problem of patients and health agents, aiming to maximize the number of patients visited by the agent, decreasing the traveling time and the number of agents necessary to the round.

The developed system is based on a mix approach of genetic algorithms and local search. The genetic algorithms were adapted from its canonical form to a viable form to the mentioned problem. The results obtained by these algorithms were satisfactory, approaching, in many cases, to the optimal solutions on the realized tests.

Finally, the complete system was created, with a web management module, which allows to the inclusion and exclusion of agents and patients and a mobile module for the exhibition of route information to the agent during the visits. The algorithm was incorporated with the web sub-system module, once it has no direct function to the user.

Even with the imposed restrictions to the aim of the project, its implementation was satisfactory. The system is functional and the results obtained are beyond the initially expected. Moreover, the system performance, even though it is not optimal, is sufficient for the requirements imposed by the application.

# REFERENCES


1. **DOHN, A. e. a.** The Home Care Crew Scheduling Problem. *1st International Conference on Applied Operational Reseach.* Yerevan, Armenia: s.n., 2008.

2. **APPLEGATE, D. e. a.***The Travelling Salesman Problem: A Computational Study.* Princeton, USA : Princeton University Press, 2006. ISBN 0691129932.

3. **TOTH, P. and VIGO, D.** The vehicle routing problem. Philadelphia, USA: *Society for Industrial and Applied Mathematics.* 2001. ISBN 0898715792

4. **KOHL, N. e. a.** 2-path cuts for the vehicle routing problem with time windows. *Transportation Science,* INFORMS Institute for Operations research and the management Sciences, Linthicum, Maryland, USA*.* 1999, Vol. 33, pp. 101-116. ISSN 1526-5447

5. **PILLAC, V, GUÉRET, C. and MEDAGLIA, A.***Dynamic Vehicle Routing Problems: State of The Art and Prospects.* Nantes, France: Ecole des Mines de Nantes, 2010.

6. **RUSSEL, S. J. and NORVIG, P.***Artificial Intelligence: A Modern Approach.* Upper Saddle River, New Jersey, USA: Prentice Hall, 2010. ISBN 9780132071482

7. **KIRKPATRICK, S. C., GELLATT, D. C. and VECCHI, M. P.** Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics,* Springer Netherlands, Amsterdam, Holand*.* 1983, Vol. 34, pp. 975-986. ISSN 0022-4715

8. **CERNY, V.** A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications.* 1985, Vol. 45, pp. 41-51. ISSN 0022-3239

9. **GLOVER, F. e. a.***New advances and applications of combining simulation and optimization.* In: [S.I.]: IEEE Computer Society, 1996.

10. **GLOVER, F. and LAGUNA, M.** Tabu search. In: SHARDA, R. et al. (Ed)*Metaheuristic Procedures for Training Neutral Networks.* New York, New York, USA: Springer US, 1997, (Operations Research/Computer Science Interfaces Series, Vol. 36), pp. 53-69. ISBN 978-0-387-33416-5

11. **GENDREAU, M.***An introduction to tabu search.* In: [S.I.: s.n.] 2003.

12. **DORIGO, M.***Optimization, learning and natural algorithms.*Dipartimento di Elettronica, Politecnico di Milano. Milan, Italy: 1992.

13. **LIU, W. et al.** Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem. Xi'an, China:*Industrial Electronics and Applications,* 2009*. ICIEA 2009.* 4th IEEE Conference on, 2009.

14. **JUNJIE, P. and DINGWEI, W.** An Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem. Beijing, China : *ICICI 06 Proceedings of the First International Conference on Innovative Computing, Information and Control.* 2006. Vol. 1, pp. 210-213.

15. **MULLEN, R. J. et al.***A review of ant algorithms. Expert Systems with Applications,* Maryland Heights, MO 63043, USA, 2009. pp. 9608-9617. Vol. 36. ISSN 0957-4174.

16. **VIANA, B. B. et al.** Utilização de autômatos adaptativos na simulação de inteligência artificial de uma colônia de formigas. *Trabalho de Conclusão de Curso da Escola Politécnica da USP - Engenharia da Computação.* São Paulo, SP, Brazil: s.n., 2006.

17. **DARWIN, C.** On the origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for life. [S.I.]: John Murray, 1959.

18. **HOLLAND, J. e. a.** Adaptation in Natural and Artificial Systems. Cambrigde, Massachusetts, USA: MIT Press, 1975. ISBN 978-0262581110

19. **SHARAPOV, R. R. and LAPSHIN, A. V.** Convergence of genetic algorithms. Lenina 51 Yekaterinburg 630083, Russia: Pattern regognition and image analysis, 2006, Vol. 16, pp. 392-397.

20. **ANDERSON, M. B.** Genetic Algorithms in Aerospace Design: Substantial Progress, Tremendous Potential. Rhode-Saint-Genèse, Belgium: *RTO-EN-022 Intelligent Systems for Aeronautics.* 2002.

21. **TOEMEH, R. and ARUMUGAN, S.** Breaking Transposition Cipher with Genetic Algorithm. Kaunas, LÇituania: *Electronics and Electrical Engineering.* 2007, Vol. 79.

22. **GARG, P., SHASTRI, A. and AGARWAL, D.***An Enhanced Cryptanalytic Attack on Knapsack Cipher using Genetic Algorithm.* Penang, Malaysia: Proceedings of world academy of science, engineering and technology, 2006. pp. 355-358. Vol. 12.

23. **CLARK, D. E.***Evolutionary Algorithms in Molecular Design.* New York, New York, USA: John Wiley and Sons, 2008. (Methods and principles in medicinal chemistry) ISBN 9783527301553.

24. **CHAMBERS, L. D.***The practical Handbook of Genetic Algorithms: Applications Second Edition*. Boca Raton, Florida: Chapman&Hall/CRC, 2000. (The practical handbook of Genetic Algorithms, Vol. 1). ISBN 978-1584882404.

25. **HAUPT, R.L. and HAUPT, S. E.** Practical Genetic Algorithms. Second Edition. Storrs, Connecticut, USA: John Wiley and Sons, 2004. ISBN 978-0-471-45565-3

26. **OMAR, M., BAHARUM, A, and HASAN, Y. A.** A job-shop scheduling problem (JSSP) using genetic algorithm (GA). Penang, malaysia: *Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and Applications,* 2006.

27. **BRANCHINI, R. M.** Busca Tabu para o problema de roteamento dinâmico de veículos com janelas de tempo. Campinas, Brazil : Thesis presented to Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, 2005.

28. **GOLDBERG, D. E. and MILLER, B. L.** Genetic algorithms, tournament selection and the effects of noise.*Complex Systems*, Champaign, Illinois, USA, 1995. pp. 193-212. Vol. 9.

29. **BACK, T.***Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* New York, New York, USA: Oxford University Press, 1996. ISBN 0-19-509971-0.

30. **LOPES, S.***Biologia Volume Único.* Sao Paulo, SP, Brazil: Editora Saraiva, 2004.

31. **OLIVER, I. M., SMITH, D. J. and HOLLAND, J. R. C.** A study of permutation crossover operators on the travelling salesman problem. In: *Proceedings of the Second International Conference on Genetic Algorithms and their applications.*[S.I.]: L. Erlbaum Associates Inc, 1987, pp. 224-230. ISBN 0-8058-0158-8

32. **DOMINIK, T. G.***Genetic Algorithms Reference, Volume 1 Crossover for single-objectiver numerical optimization problems.* Lomianki, Poland: Tomasz Gwiazda, 2006. ISBN 978-8392395836.

33. —. *Genetic Algorithms Reference, Volume II Mutation operator for numerical optimization problems.* Lomianki, Poland: Tomasz Gwiazda, 2007. ISBN 978-8392395843.

34. **GENDREAU, M., LAPORTE, G, and POTVIN, J.-Y.** Metaheuristics for the Vehicle Routing problem. Montréal, Québec, Canada: Les Cahiers du Gerad, G-98-52, 1998.

35. **SAFE, M. et al.** On Stopping Criteria for Genetic Algorithms. *Lecture Notes in Computer Science.* São Luis, Maranhão, Brazil: s.n., 2004. pp. 405-413.

36. —. On Stopping Criteria for Genetic Algorithms. *Lectures Notes in Computer Science.* Sao Luis, Maranhão, Brazil: s.n., 2005. pp. 405-413.

37. **BOCK, F.** An algorithm for solving travelling-salesman and related network optimization problems. Saint Louis, Missouri, USA: manuscript non published, presented on the 14th ORSA National Meeting, 1958.

38. **NETO, J.J.** Adaptive rule-driven devices - general formulation and case study. Pretoria, South Africa: *CIAA 2001 Sixth International Conference on Implementation and Application of Automata*, 2001. pp. 234-250.

39. **LOPES, V. D.** Proposta de Integracao entre tecnologias adaptativas e algoritmos genéticos. São Paulo, SP, Brazil:*Dissertation presented to Escola Politécnica da USP to obtain the title of master in Engineering.* 2009.