Data Driven Approach to Enhancing Efficiency and Value in Healthcare

Richard E. Guerrero Ludueña

Part III

Metaheuristic optimisation

Chapter 5

Routing Home Healthcare services

This chapter discusses the problem of routing healthcare professionals to deliver home healthcare services to the patients, and how metaheuristics algorithms can be used to solve this problem.

This chapter is structured as follows: Section 5.1 introduces the Clinical Homecare and the impact of this system of care in National Health Services; Section 5.2 presents the Home Health Care Problem from an Operational Research point of view, and discusses the application of the classic Travelling Salesman Problem to routing healthcare professionals; Section 5.3 presents algorithm and the parameter selection process; Section 5.4 presents the implementation of GA and experimental results; and Section 5.5 summarises the chapter.

5.1 Introduction

Home healthcare is defined as a system of care provided by skilled practitioners to patients in their homes under the direction of a physician [54]. Home healthcare include services such nursing, physical therapy, speed-language therapy, occupational therapy, medical social services, medical visits, house cleaning, home life aides, old people assistance, etc. Examples of home healthcare services are the attention of patients with cancer; palliative care at home; home treatment of long-term conditions, care of patients with early hospital discharge; and care home 'virtual' wards to promote Recovery at home [88, 89, 90].

The UK National Clinical Homecare Association (NCHA), organisation that represents companies and organisations providing clinical homecare services to 275,000 patients in the UK mentioned on their website a growth over 20% year on year [148]. NCHA defined Clinical Homecare as the provision of medicines, supplies and supporting clinical services directly to patients at times and places most convenient to them.

In a report published in 2015 in the UK [89], a group of experts from the public and private healthcare sector concluded that the Clinical homecare helps to address two major challenges for the NHS:

- Faster recovery, better quality of life and less hospital readmissions for patients.
- Reduction of pressure for hospital beds, waiting lists, and the number of hospitalised patients.

The goals of home healthcare services are to help individuals to improve function and live with greater independence; to promote the patient's optimal level of wellbeing; and to assist the patient to remain at home, avoiding hospitalisation or admission to long-term care institutions [54].

The NCHA summarises the benefits of Clinical Homecare as follows [148]:

- *Benefits for patients.* Improved treatment outcomes as a result of faster access to treatment and regular contact and support from healthcare professionals. Choice of location for service. Time and money saving associated with travels and time off work. Control over treatment. Discrete and confidential services.
- *Benefits for clinicians.* Improved treatment outcomes by proactive prescription management and treatment delivered according to agreed pathways and protocols. Fully traceable supply chain. Reduced hospital capacity pressure.
- *Benefits for Commissioners.* Reduced capacity pressure on hospitals, treatment costs, and working capital. Better clinical outcomes as a result of improved adherence to treatments.

Both the NHS Five-year-forward view (FYFV) report published in October 2014 by NHS England and other bodies [154, 136], as the roadmap for NHS between 2015 and 2020, and the Next steps on the NHS FYFV report published in March 2017 [153], anticipates the further extension of clinical homecare services to treat patients in community settings.

5.2 Home Health Care Problem and Travelling Salesman Problem

As described in the Section 5.1, the Home healthcare service consists of visiting patients who required some professional treatment in the home or other location out of acute care settings. Home healthcare are an important component of healthcare systems with a potential to lower the system-wide costs of delivering care and free capacity in overcrowded care settings such as hospitals.

In Operational Research, the planning problem addressed with this service model is known as Home Health Care Problem (HHCP). One of the first papers dealing with this problem was published in 1997 [11], and since then many scholars have conducted research in the application of home healthcare, considering different optimisation criteria, for example: travel time or cost; overtime costs; number of visits; workload balance; and waiting time for patients [133, 132, 3, 157].

The Travelling Salesman Problem (TSP) is one of the most widely studied combinatorial problems in Operational Research (OR) and computer science [44] [123]. The TSP presents the task of finding an optimum path through a set of given locations (cities), such that each location is passed through only once, and the salesman returns to the start location [52]. The Travelling Salesman Problem is one of the famous NP-hard problems, which means that there is no perfect algorithm to solve it in polynomial time. The minimal time required to obtain optimal solution is exponential [114]. Exact algorithms, including enumeration method and branch and bound algorithm are only suitable for small scale problems due to the limitation of time and memory. In contrast, heuristic techniques and intelligent optimisation algorithms such as genetic algorithm, ant colony optimisation and simulated annealing are reliable to find an acceptable solution within reasonable time.

The TSP aims to find the shortest path to visit each city once and only once. Both symmetric and asymmetric TSP can be modelled as a complete graph $G = (N, A)$, where N is a set of n cities, and A is a set of arcs. The cost of distance of each $arc(i, j) \in A$ is represented in d_{ij} in the distance matrix D. The problem is defined as follows:

The variables:

$$x_{ij} = \begin{cases} 1 & \textit{if the } arc(i,j) \textit{ is selected in the path} \\ 0 & \text{otherwise} \end{cases}$$

The objective function:

$$min\ C = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{5.1}$$

The constraints:

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j \in N \tag{5.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i \in N \tag{5.3}$$

$$x_{ij} \in \{0, 1\}, \quad i, j \in N \tag{5.4}$$

$$\sum_{i,j \in S} x_{ij} = |S| - 1, \quad S \subseteq N \tag{5.5}$$

Where $|S|$ is the number of nodes in set S.

Equation 5.1 shows the requirement to minimise the total cost of the tour. Constraints (5.2-5.5) assure that the tour visits every city once and only once.

HHCP can be compared with the TSP with some special constrains (e.g. time windows, fixed number of travelling salesmen, maximum or minimum number of vertices to visit, etc.) on a complete directed graph.

The HHCP can be defined as a combination of two/three problems:

- Assignment of care services to a sub-population of patients
- Assignment of care workers to specific care services
- Routing the care working team

In this chapter, the focuses is on the routing problem, directly linked with the TSP described above. Next section includes the implementation of a metaheuristic algorithm to solve this problem.

5.3 Proposed solution based on a Genetic Algorithm

In this section, a Genetic Algorithm (GA) approach to solve the problem of routing clinical staff to deliver clinical homecare services is presented. The problem is modelled using the representation described above.

GA was introduced by Holland in 1975 [94], and is a search method for solving combinatorial optimisation problems based on the natural selection process proposed by Darwin, whereby organisms evolve by rearranging genetic material to survive in environments confronting them ('Survival of the fittest'). The algorithm repeatedly modifies a population of individual solutions. GA provide optimal or near optimal solutions for both constrained and unconstrained optimisation problems. GA is part of the Evolutionary Algorithms, a classic example of heuristic search algorithms yet they do not yield exact optimal solutions, but will certainly help to find better optimal solutions when compared with other algorithms within less amount of time.

Several version of GA implemented in Java are presented. The impact of different selection strategies and genetic operators on the performance of the GA is also analysed.

As described in Chapter 2, the GA is a general method for solving *search for solution* problems (as are other evolution-inspired methods, such as evolutionary programming and evolutionary strategies). Figure 5.1 indicates the position of the GA in the family three of optimisation techniques classified by search techniques.

Algorithm 1 presents pseudo-code for a classic GA.

Algorithm 1 *A classic Genetic Algorithm*

```
INITIALISE population with random candidate solutions
EVALUATE each candidate solution
DETERMINE population's average fitness

while termination condition is not true do
  SELECT individuals for the next generation
  RECOMBINE pairs of parents
  MUTATE the resulting offspring
  EVALUATE each candidate solution
end while
```

Each iteration of this process is called *generation*. The entire set of generations is called a *run*. At the end of a run there are often one or more highly fit chromosomes

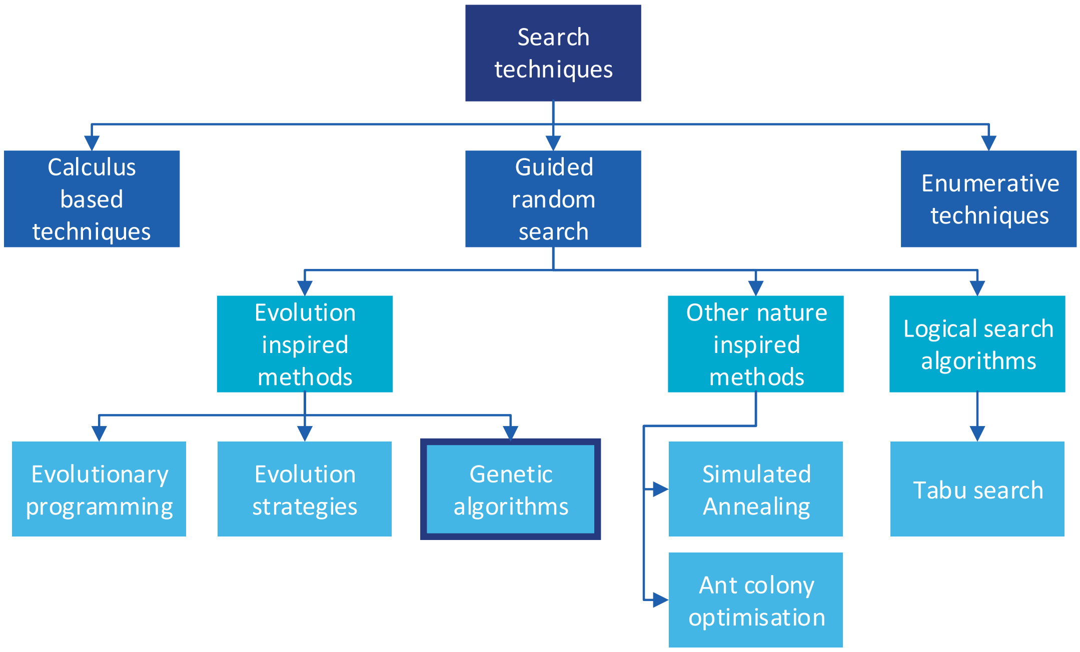

Figure 5.1: Genetic Algorithms. Family three of Optimisation approaches based on search techniques.

in the population. Since randomness plays a large role in each run, two runs with different random-number seeds will generally produce different detailed behaviours.

Procedure described above and summarised at the Figure 5.2, describes the basis for most applications of GAs. There are a number of details to fill in, such as the size of the population and the probabilities of crossover and mutation, and the success of the algorithm often depends greatly on these details. There are also more complicated versions of GAs (e.g., GAs that work on representations other than strings or GAs that have different types of crossover and mutation operators).

The proposed algorithm consists of chromosome representation, generation of initial population, determination of fitness function followed by the crossover and mutation operation. We discuss all these steps in the following sections.

5.3.1 Genetic encoding

Chromosomes were represented using an *order-based representation*. Figure 5.4 shows the this representation, where a possible solution is coded as the sequence of visited cities (or patients).

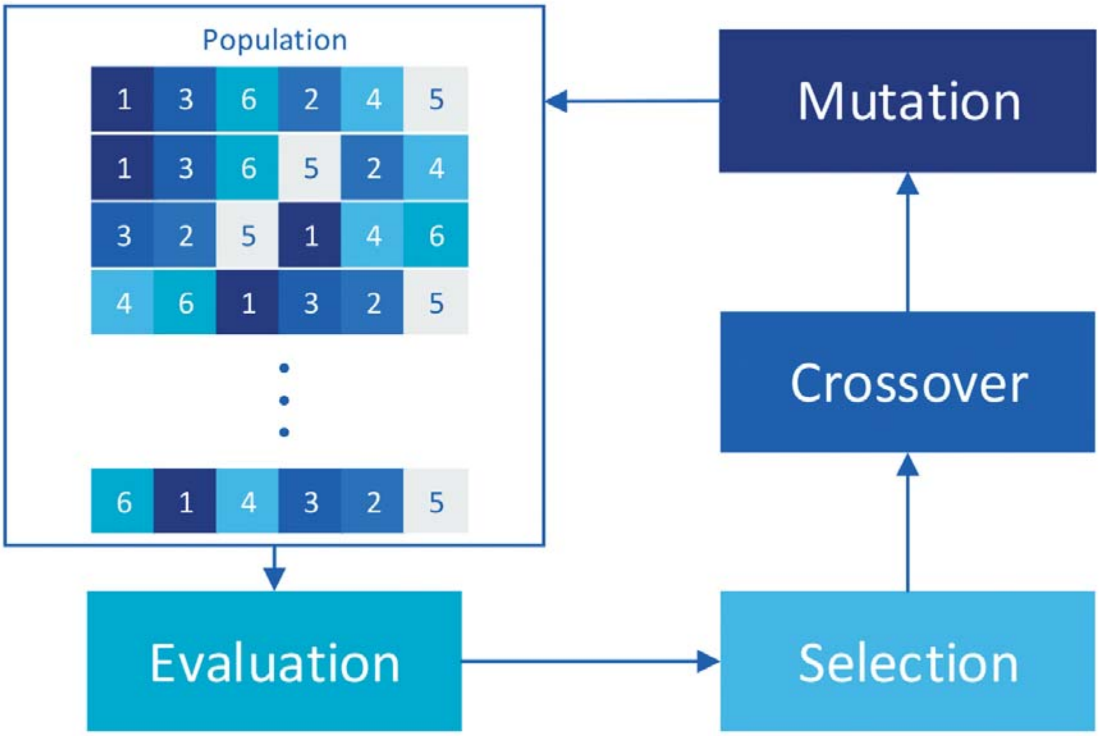

Figure 5.2: Evolution flow of Genetic Algorithms.

Chromosome C was represented as a permutation of cities (or patients) $(c_1c_2...c_n)$. For instance, $C = 136245$ represents the path $1 \rightarrow 3 \rightarrow 6 \rightarrow 2 \rightarrow 5 \rightarrow 1$.

5.3.2 Initial population

The initial population is a randomly generated set of chromosomes in which each chromosome is a collection of numbers which represent a path.

The initial population was created by using the encoding representation described above. A random population of 1,000 individuals was generated with the same path for all of them $(1 \rightarrow 2 \rightarrow 3... \rightarrow n)$. A permutation based on *Order changing*- where two numbers are randomly selected and exchanged - was applied $4n$-times to all of them. The same population size was used through all the process. Initial population generation algorithm is shown in Algorithm 2.

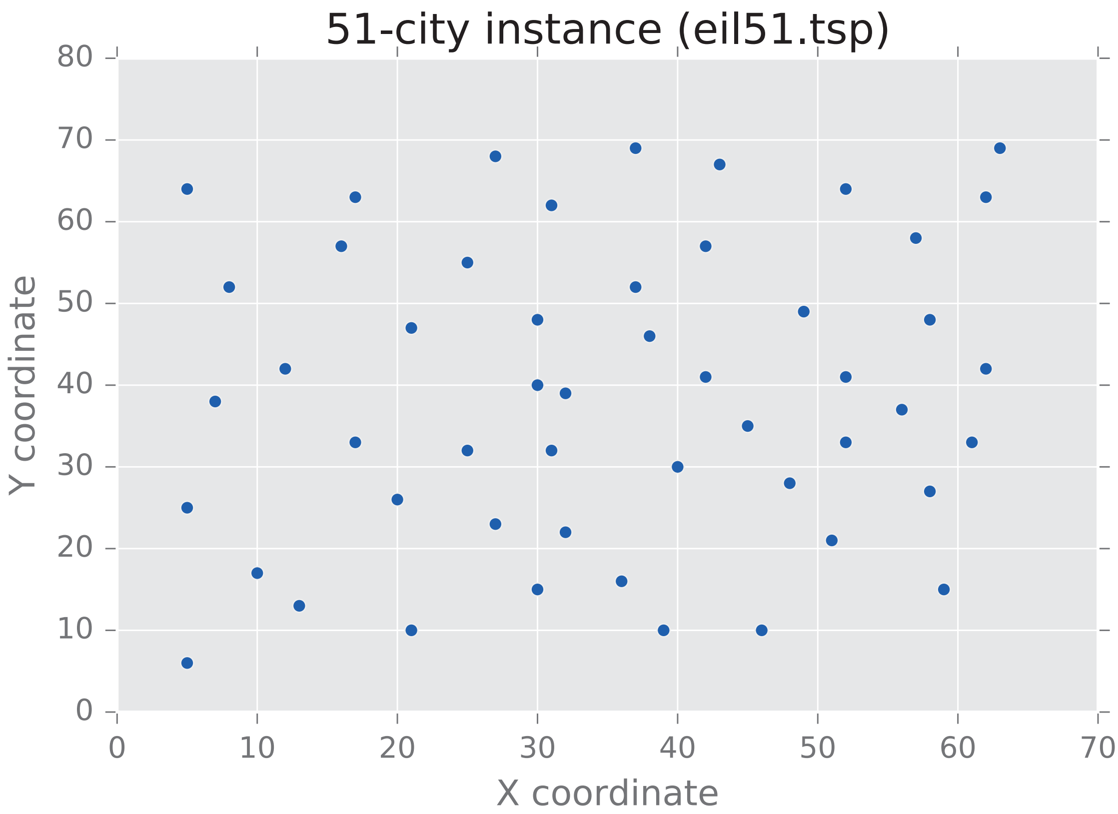

Figure 5.3: TSP instance with 51 cities to visit (eil51.tsp) from TSPLIB library.

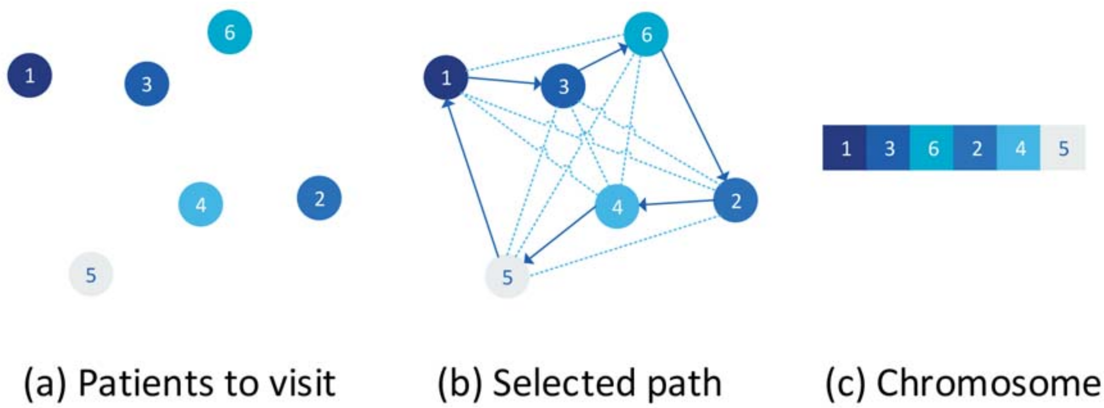

Figure 5.4: Order-based representation: the chromosome (possible solution) is coded as the sequence of cities (or patients) to visit. In (a), the set of patients to visit are shown. In (b), a path selected to visit all patients is shown. In (c), the associated chromosome is represented as the sequence of the visited patients.

Algorithm 2 *Initial Population*

```
InitialPopulation = [ ];

for all chromosome members of population do
  for all genes of chromosome do
    gene_i = i
  end for
  SWAP genes randomly selected
end for
```

5.3.3 Estimation of fitness function

The fitness value of a chromosome represents its level of quality on the basis of the objective. In the proposed implementation, the goal is to select the shortest tour. is the criterion of individuals' quality and it is associated with the objective function. Equation 5.6 shows the definition of fitness function, in which $d_{c_i c_{i+1}}$ is the distance between c_i and c_{i+1}, and $\sum_{j=1}^{n-1} d_{c_i c_{i+1}} + d_{c_n c_1}$ is the tour distance.

$$fitness = \left(\sum_{j=1}^{n-1} d_{c_i c_{i+1}} + d_{c_n c_1} \right)^{-1} \tag{5.6}$$

5.3.4 Selection and genetic operators

To the selection of parents to be reproduced to the next generation, the fitness was compared using a *k-tournament selection*, with $k = 2$. Two individuals were randomly selected and then the individual (or chromosome) with greater fitness was selected into the next generation.

After several experiments, the mutation rate that consistently generated the best solutions was selected. An order changing permutation encoding was used as a mutation operator, with a probability of 0.0001. Invert-section permutation was used as for crossover with a probability of 0.0001 of crossover. Figure 5.5 shows the effect of this two genetic operators in a chromosome.

5.4 Experimental results

Several experiments were done to compare the performance of the implementations of GA, using different initialisation techniques, population size, genetic parameters,

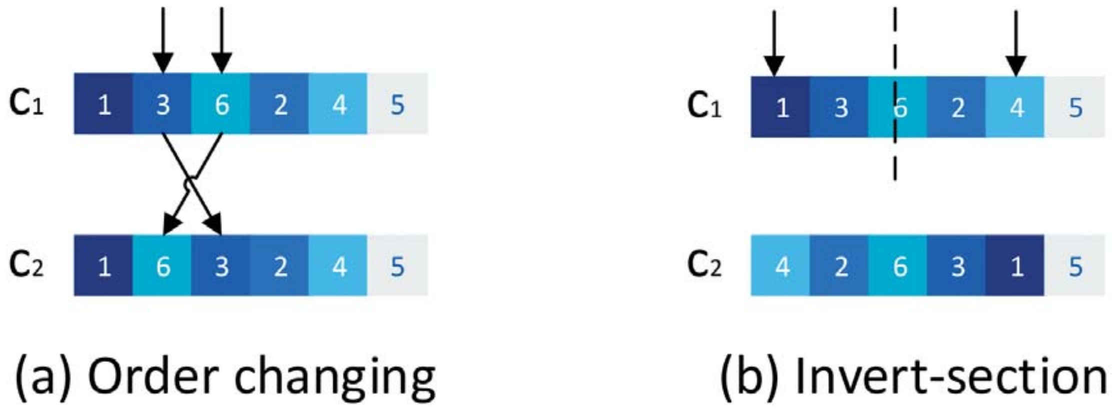

Figure 5.5: Genetic Operators for mutation and crossover. (a) shows the effect of the *Order changing* mutation operators over the chromosome C_1, selecting the nodes 3 and 6. (b) shows the effect of the *Invert-section* operator used as a crossover.

and processes to create generate the offsprings, with instances extracted from TSLIB [175].

The GA was coded in Java on a computer with 2.90 GHz CPU and 16 GB physical RAM. Each case run 50 times, and the average value was collected for analysis.

In the first implementation (GA01), the instance *eil51.tsp* (see Figure 5.3) was solved using the genetic encoding and genetic operators in the same manner as described above. The process of evolution ends after 50,000 generations.

Figures 5.6 and 5.7 demonstrated the convergence process after 50,000 generations of the GA, in a run for an instance with 51 cities or patients to visit, average and best solution for each generation are presented, as well as the optimal solution extracted from the TSPLIB website [175]. Figure 5.7 shows the performance of the algorithm for the 250-first generations, with a convergence to a population with a small variability between individuals after 10 generations.

Figure 5.8 shows the best solution for the first generation randomly generated, the best solution found after 50,000 generations, and the optimal solution. The initial solution has a length of 1,358 units, and the best solution of the GA was 435. Optimal solution reported at the TSPLIB website for this problem is 426.

Different seeds are used to generate random numbers involved in the initialisation, selection, reproduction, and replacements operations of a GA. Random numbers are used to generate the initial population, to select the parents in each generation, to create new offspring through the application of mutation and crossover operators, etc. Figure 5.9 demonstrates the impact of the random seed in the performance of

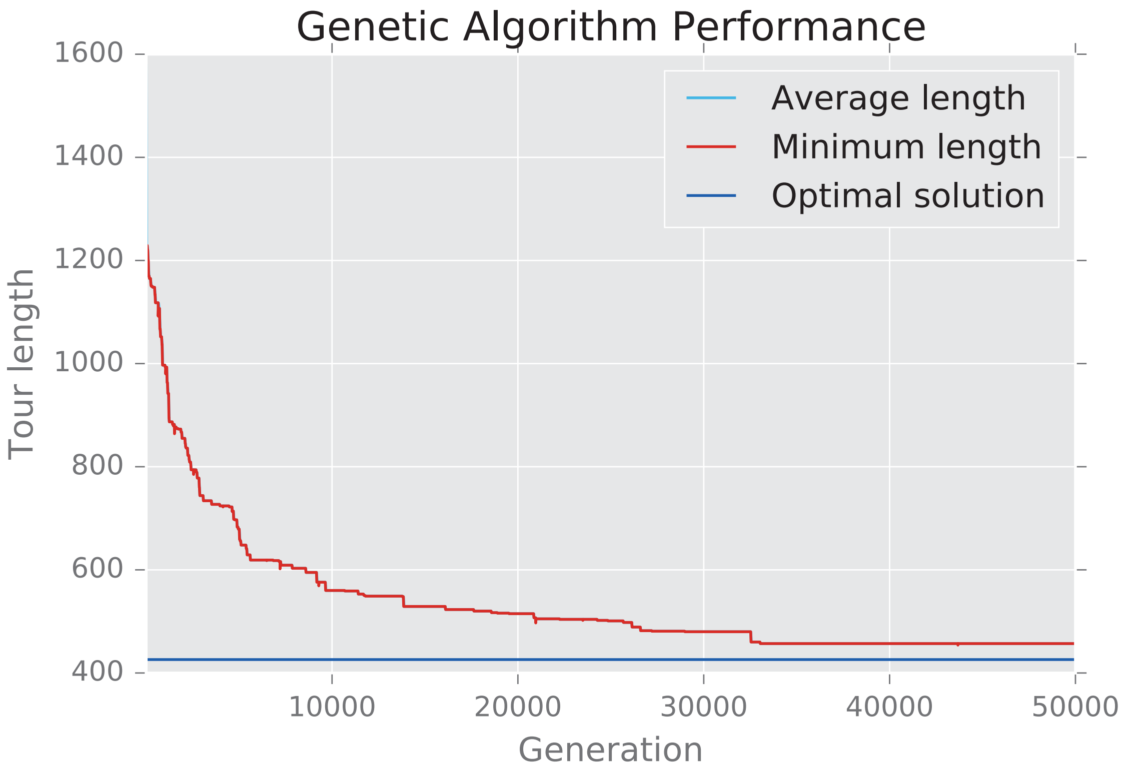

Figure 5.6: Performance of the Genetic Algorithm: graph shows the average and best solution in each generation. Optimal solution extracted from TSPLIB website is also included.

the GA. The graphs shows the results of 20 runs with the same parameters and with different seeds. Lines represents the minimum, average, and maximum tour length.

Figures 5.6, 5.7, and 5.9 shown the GA performance with pre-defined parameters such population size, crossover rate, and mutation rate (selected experimentally).

Below we present an evaluation of the combined impact of those parameters in the performance (tour length), and in the execution time. Figure 5.10 shows the impact of *mutation rate* and *population size* for different *crossover rates* on the tour length. Figure 5.11 shows the impact on the execution time.

As Figure 5.10 illustrates for six datasets, the surfaces defined by these points do not vary significantly from one crossover rate to another. We observed two main trends. Tour length decrease with the population size. Impact of mutation rate decreases with population size. Interestingly, a good performance is observed A difference of less than 5% with the optimal solution (426) is observed without relation with crossover nor mutation rates, when a GA is implemented with population over

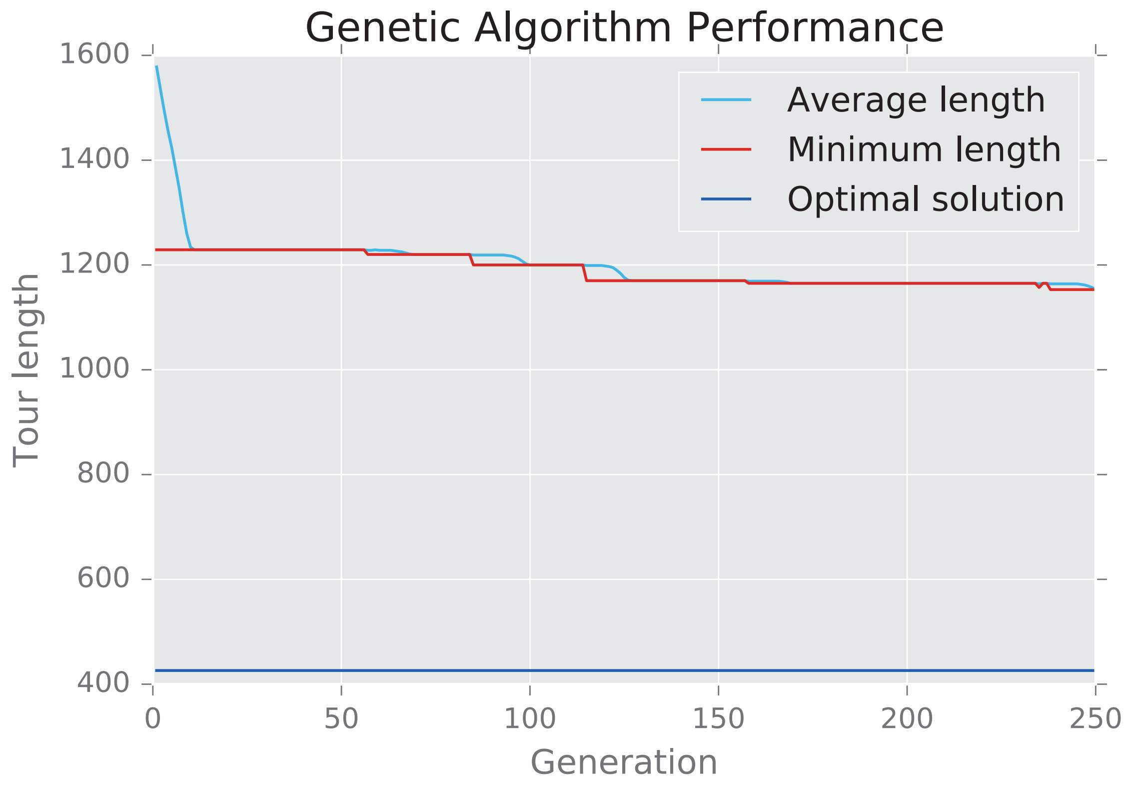

Figure 5.7: Performance of the Genetic Algorithm for the 250-first generations: graph shows the average and best solution in each generation. Optimal solution extracted from TSPLIB website is also included.

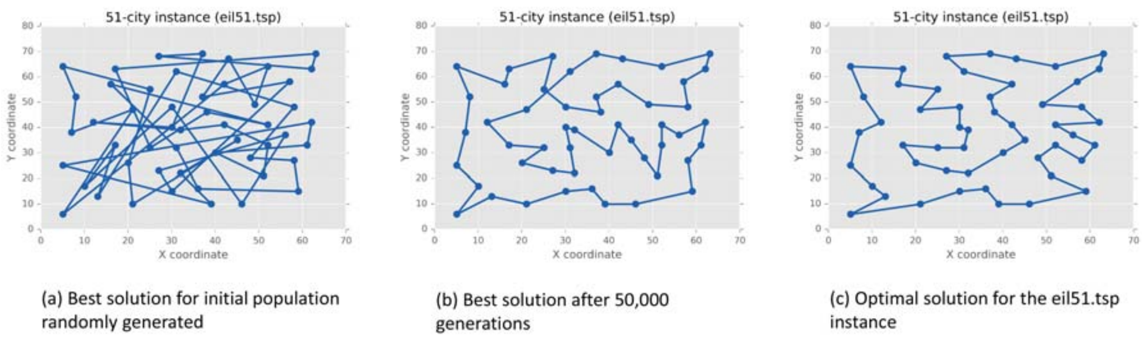

Figure 5.8: TSP instance with 51 cities to visit (eil51.tsp) from TSPLIB library. Random solution, best solution with GA, and optimal solution.

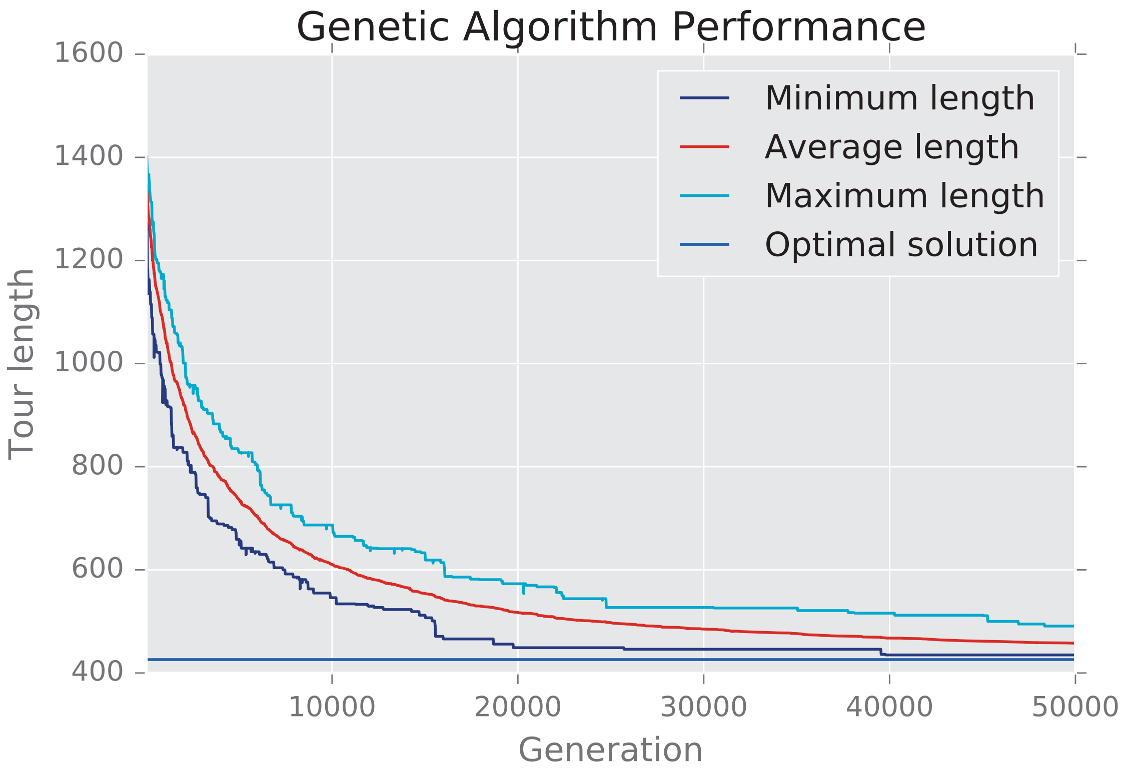

Figure 5.9: Performance of the Genetic Algorithm: graph shows the impact of the random seed in the performance of the algorithm. Minimum, average and maximum tour length after 20 runs with the same parameters and with different random numbers.

1,000 chromosomes. Figure (f) shows the effect of a crossover rate = 1 (i.e., the crossover operator is applied to all the chromosomes selected for the next generation), providing additional randomness when the algorithm is applied to an small population.

In Figure 5.11, execution time remains constant (near zero) until the population size grows to a value between 100 and 1,000, after that, the execution time grows exponentially with the population size in most of the cases. The surfaces defined by these points vary significantly from one case to another when the population size is over 1,000 chromosomes. The choice of the mutation rate does not have a clear impact on the execution time (See for example (b) and (f)).

Overall, this analysis suggest that not only the best pair crossover and mutation rates are important, but mainly the population size to find a good solution without sacrificing execution time. It is particularly important when implementing Genetic

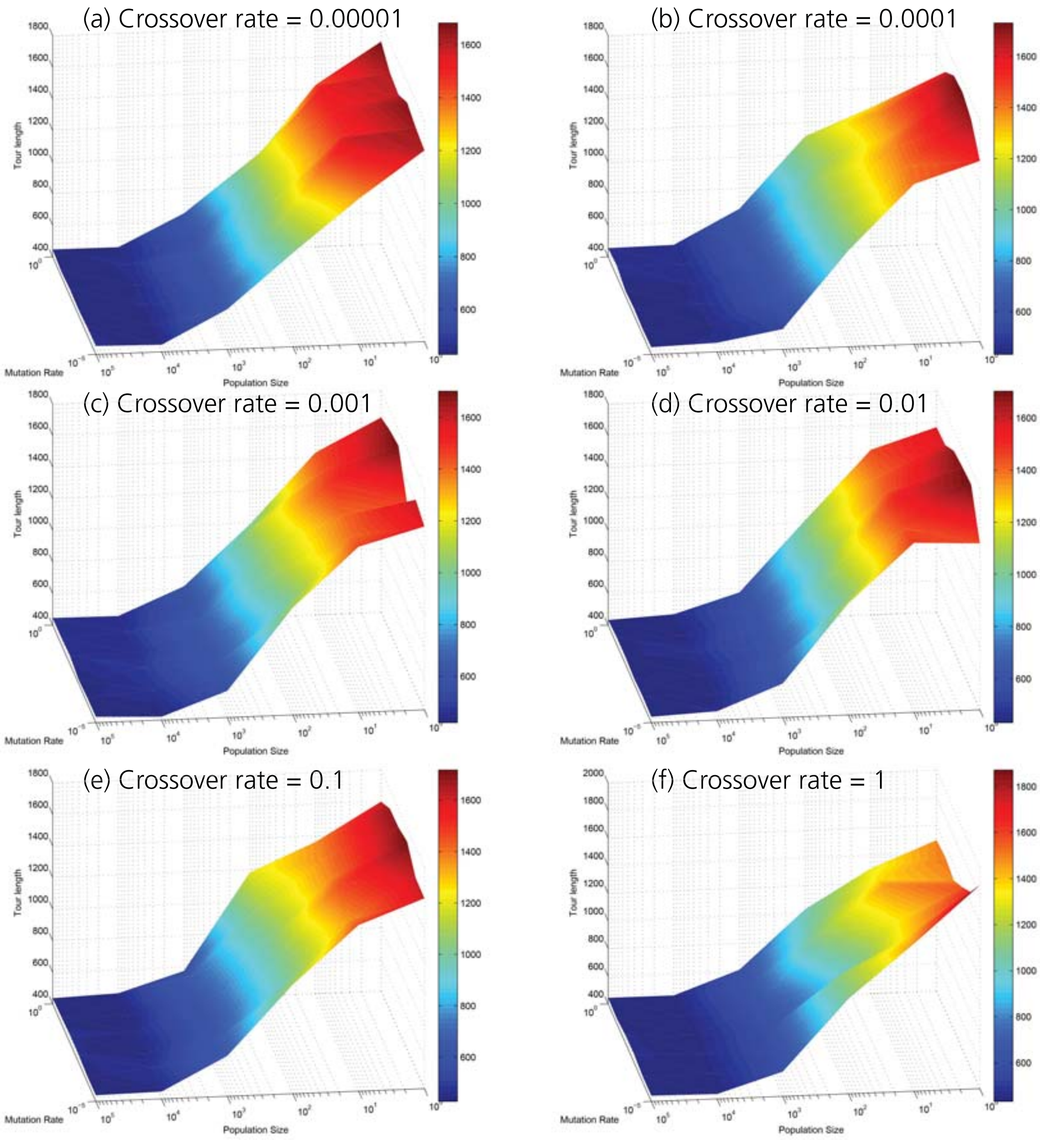

Figure 5.10: Impact of parameters on the Genetic Algorithm performance: x-axis represents the population size (log); y-axis represents the mutation rate (log); and z-axis shows the tour length.

Algorithm to solve real world problems, as is the case of the home care routing problem analysed in this chapter.

5.4.1 Conclusions

This chapter described how Metaheuristic optimisation can be used to solve complex problems associated, for example, with scheduling or routing in healthcare. Particu-

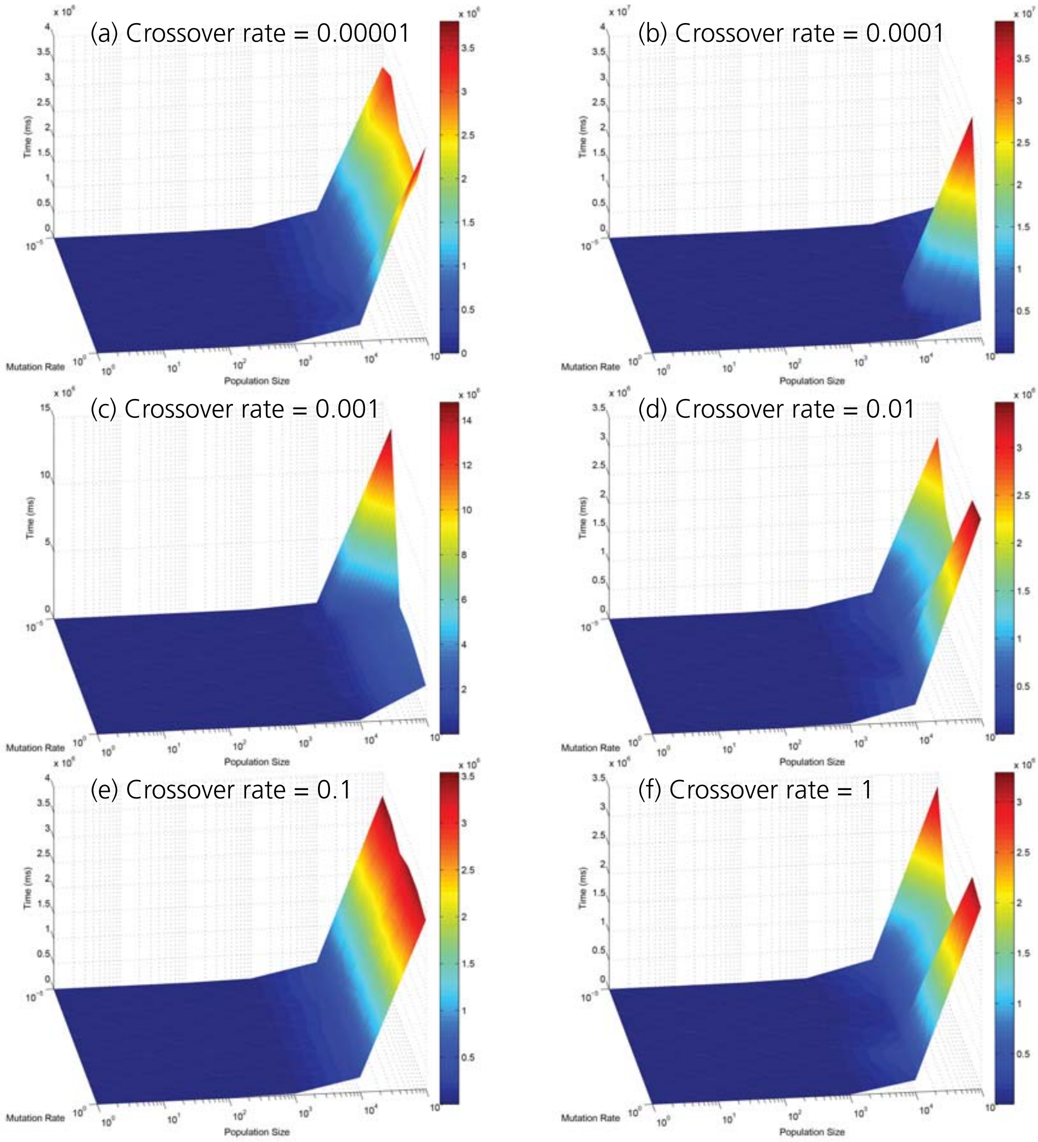

Figure 5.11: Impact of parameters on the execution time: x-axis represents the population size (log); y-axis represents the mutation rate (log); and z-axis shows the execution time.

larly, Genetic Algorithm was implemented to solve a simplified version of the Home Care Healthcare Problem, focusing on the problem of routing healthcare professionals to deliver home healthcare services to the patients.

The basics of optimisation and the Travelling Salesman Problem were presented. Genetic Algorithm was introduced in order to shed new light on the impact of parameters selection in its performance and execution time. An theoretical case was solved using a benchmarking dataset.

Several commercial options are currently available to solve this and much more complex problems related with scheduling and routing healthcare professional, nevertheless, in this chapter we described an alternative approach using only open source tools.

Future research can consider the solution of a problem with multi-agents, or additional constraints as time-windows for the patient visit, work-load balance between clinical staff, or patients preferences. Additional metaheuristic can also be analysed.

Metaheuristic optimization deals with optimization problems using metaheuristic algorithms. This approach can provide a useful solution to tackle complex problem in healthcare, and through this, enhancing efficiency and value. Metaheuristic algorithms remains an open field of research for which many questions are still left unanswered, even regarding well-established methods.

5.5 Chapter Summary

Section 5.1 introduced the problem of routing healthcare professionals to deliver home healthcare services to the patients. Section 5.2 presented the Home Health Care Problem and the Travelling Salesman Problem. Section 5.3 described a solution based on a Genetic Algorithm (GA), described the algorithm and the process of parameter selection. Section 5.4 presented the implementation of GA and the experimental results. Finally, Section 5.4.1 summarised the conclusions and future research.