Multi-objective genetic algorithm

Robin Devooght

31 March 2010

Abstract

Real world problems often present multiple, frequently conflicting, objectives. The research for optimal solutions of multi-objective problems can be achieved through means of genetic algorithms, which are inspired by the natural process of evolution: an initial population of solutions is randomly generated, then pairs of solutions are selected and combined in order to create new solutions slightly different from the initial solutions. The fittest solutions are kept in the population and are used to generate new solutions. Generations after generations, the initially random population converge towards optimal solutions. The genetic algorithms have the advantage of being easily implemented in any multi-objective problem, and are used in numerous area such as safety systems and electrical network.

In this report, key concepts related to multi-objective optimization problems have been presented, such as the notion of decision variables (the actionable characteristics of the problem), objective functions (the relations between the actionable and the observable characteristics) and Pareto optimality (which formally describe how to compare solutions in a multi-objective problem). The mechanisms of genetic algorithms have been described, including the following: the crossover and the mutation which constitute the mechanism of reproduction, the method used for ranking the solutions in a population and the mechanism of selection.

A genetic algorithm was implemented to solve a two-objective problem consisting in expanding a power transmission network while optimizing the cost and the reliability of the network. A flaw of the genetic algorithm was illustrated by this example: in spite of a clear convergence of the algorithm towards optimal solutions, the genetic algorithm did not cover uniformly the search space, and the most interesting part (the solutions with low cost) was not explored.

Three methods were proposed to guide the convergence of the genetic algorithm towards desired solutions: biasing the generation of the initial population in order to favour the exploration of a certain region of the search space, attributing different weights to the objectives and comparing the solutions according to the weighted sum of the objectives (WP-MOGA), and including minimum and maximum trades-off in the comparison of solutions (G-MOGA). The three methods were tested on the network expansion problem in order to find solutions that imply adding

few new transmission lines. The weighted sum method appeared to be inefficient, while the two other succeeded in guiding the random population towards solutions with few added lines. Biasing the initial population was the simplest to implement and furnished the best results, but the ease of using this method depend on the problem considered, and it might not be implementable in most multi-objectives problems. Contrariwise, the G-MOGA is universal and may be implemented in any kind of multi-objective problem.

The results were analysed through means of the reliability centrality measures, which indicates the relative importance of the nodes in the network, according to the degree of connection of the node, its closeness to the other nodes of the network, its presence in the most reliable paths of the network, and the effect of removing this node to the reliability of the network.

Contents

1 Introduction **5**

2 Important concepts **6**
2.1 Decision variables and objective functions 6
2.2 Pareto optimality 7

3 Genetic algorithms **7**
3.1 Genotype and phenotype 8
3.1.1 Representation of the decision vector 8
3.2 Ranking 9
3.3 Reproduction 9
3.3.1 Crossover 9
3.3.2 Mutation 10
3.4 Simple multi-objective genetic algorithm 10

4 Concrete example: expansion of an electrical network **10**
4.1 Decision vector 11
4.2 Phenotype and objective functions 12
4.3 Initial population 13
4.4 Reproduction 13
4.5 Replacement 13
4.6 Results 13

5 Guided convergence **14**
5.1 Biased initial population 16
5.2 Weighted Pareto-based multi-objective algorithm 18
5.3 Guided multi-objective genetic algorithm 19
5.4 Comparison 22

6 Interpretation of results **23**
6.1 Reliability degree centrality 24
6.2 Reliability closeness centrality 25
6.3 Reliability betweenness centrality 25
6.4 Reliability information centrality 26
6.5 Conclusion 26

7 Conclusion **26**

A Characteristics of the network **32**

B Matlab code **34**
B.1 Objective functions 34
B.1.1 Reliability Efficiency 34
B.1.2 Cost 34
B.2 Initial population 34

B.2.1 Biased initial population 35
B.3 Reproduction 36
B.3.1 Crossover 36
B.3.2 Mutation 36
B.4 Replacement 36
B.4.1 Ranking 36
B.4.2 Replacement 37
B.5 Pareto dominance 38
B.5.1 Pareto dominance with reasonable trades-off 38
B.6 Weighted sum 39

List of Figures

1 Convergence of the genetic algorithm. 14
2 Non-dominated solutions reached by a initial population of 200 elements after 10^5 reproductions. 15
3 Run-time in fonction of the number of non-dominated solutions found. 15
4 Comparison of initial populations 16
5 Non-dominated solutions reached by a initial population (favoring the absence of new transmission lines) of 200 elements after 10^5 reproductions. 17
6 Non-dominated solutions reached with an initial population favoring the absence of new transmission lines in [1]. 17
7 Non-dominated solutions reached by a initial population of 200 elements after 10^5 reproductions by the Weighted Pareto-based method. 20
8 Visualization of the Pareto dominance principle in a two objectives problem, with and without reasonable trade-off. 21
9 Non-dominated solutions reached by the guided multi-objective genetic algorithm with an initial population of 200 elements after 10^5 reproductions. 22
10 Adjacency matrix of the IEEE RTS 96 network 32

1 Introduction

Multi-objective algorithms enable to face optimization problems where multiple criteria have to be simultaneously optimized. Those criteria may be conflicting and the goal of a multi-objective algorithm is to find the solutions that present the best compromise between the optimization of each criteria. For example, when designing a new product, important aspects, such as the production cost and the overall quality, have to be taken into account. Those two aspects should be simultaneously optimized, but they cannot be considered independently because every choice made to improve the quality of the product might have an influence on its production cost, and vice versa.

In those kind of optimization problem, a solution that would offer the optimum in every single criteria rarely exist, but the algorithm can look for the solutions that offer the best trades-off. The best trades-off are the solutions for which no characteristic can be improved without deteriorating other characteristics. Those best solutions form the Pareto front [2].

The set of criteria that has to be optimized is called the objective vector [3]. Those criteria are observable but cannot be directly modified: they are indirectly influenced by a set of actionable characteristics called the decision variables [4]. For example, the resistance of a beam (objective vector) cannot be directly chosen, but it is influenced by its thickness and its shape (decision variable vector).

There are different multi-objective algorithms as shown in [5]. This report will focus on a special kind of multi-objective algorithm: the multi-objective genetic algorithm (MOGA). The reason is that this algorithm is widely used in broad area such as safety systems [6, 7], network optimization [1, 8] or human resources [9]. The genetic algorithm has indeed the advantage of being easy to implement, and of being able to treat non analytical problems. This algorithm is called "genetic" because its principle is based on the theory of evolution. First an initial set of solution is randomly created: it is the initial population. Pairs of solutions are combined in order to create new solutions (imitating the reproduction in nature), and only the fittest solutions according to the different objectives are kept in the population (as in natural selection). Choosing the fittest solutions implies to have a way of comparing solutions. The comparison criterion is the Pareto dominance, which will be explained later. Generation after generation, the population will drive towards the Pareto front.

In [1] MOGA is used to find the cheapest way to improve a network reliability. The goal is to optimize the network according to two criteria: the cost and the reliability efficiency. This report is based on that paper.

The main disadvantages of genetic algorithm are the following:

- the long run-time that is needed to find optimal solutions [10],
- the population may converge towards a limited region of the Pareto front, ignoring solutions that might be interesting.

When the interesting region of the Pareto front is a priori known, both

disadvantages can be improved by guiding the convergence of the population towards the desired solutions. In the case of the network optimization, the cost is considered to be much more important than the reliability efficiency. In order to guide the genetic algorithm towards solutions that imply a low investment, three methods are experienced in [1]:

1. Choosing an initial population that already present a low investment cost,
2. Giving different weight to each objective (cost and reliability efficiency) (WP-MOGA),
3. Setting reasonable trades-off between cost and reliability efficiency (G-MOGA)

The results will then be analyzed through means of the centrality measures described in [11] which are used to indicates the relative importance of a node in a network.

2 Important concepts

Before introducing the multi-objective genetic algorithms, a formal description of multi-objective problems is needed.

2.1 Decision variables and objective functions

When facing multi-objective problems, it is important to avoid confusion between the actionable characteristics of the problem and the observable characteristics that has to be optimized. The optimization of each observable characteristic is seen as an independent objective. This is where the term "multi-objective" comes from. The observable characteristics might be the life-time, the strength, the iso-entropic efficiency, etc. In general those characteristics cannot be directly chosen, but they are influenced by the value of other characteristics (such as the type of material or the shape of the product) that can be directly and independently modified. Those characteristics are called the decision variables, and the set of all decision variables is called the decision variables vector [4].

The set of all possible decision variables vectors form the search space. The goal of a multi-objective algorithm is to explore the search space and find the decision variables vectors that optimize the observable characteristics. In order to do that, the relation between actionable and observable characteristics has to be known. Those relations are given by the objective function $f_i : E \to \mathbb{R} : U \mapsto f_i(U)$. Where U is the decision variable vector, E the search space and $f_i(U)$ the value of the i^{th} observable characteristic. The objective functions may be non-analytic. The set of the value of all objective function is called the objective vector [3].

2.2 Pareto optimality

Solutions of a multi-objective optimization problem can be hard to compare, because each solution can be the optimal regarding to one objective, but bad solution regarding to other objectives. The most natural way to classify different solutions is to use the concepts of Pareto optimality and Pareto dominance [3].

The best compromises are the solutions for which no objectives can be ameliorated without provoking the degradation of at least one other objective. Those solutions are called Pareto-optimal, and the set of all the Pareto-optimal solution is called the Pareto front. The goal of a multi-objective algorithm is to find the Pareto optimal solutions, or at least solutions closed to the Pareto front. A comparison criterion called Pareto dominance is derived from the principle of Pareto optimality. As defined in [4], a solution X (with decision vector U_x) is said to dominate a solution Y (with decision vector U_y) if:

$$\forall i \in \{1, \ldots, n\} : f_i(U_x) \geq f_i(U_y) \tag{1}$$

$$\exists j \in \{1, \ldots, n\} \mid f_j(U_x) > f_i(U_y) \tag{2}$$

Where $f_i : E \to \mathbb{R} : U \mapsto f_i(U)$, with $i \in \{1, ..., n\}$ are the n objective functions of the optimization problem.

In a subset of the search space, the solutions that are not dominated by any other solutions of the subset regarding to the concept of Pareto dominance are the non-dominated solutions of the subset. The non-dominated solutions of the search space form the Pareto front.

3 Genetic algorithms

The theory of evolution, exposed by Darwin in 1859, is both simple and extremely powerful. To quote Darwin [12]:

> As many more individuals of each species are born than can possibly survive; and as, consequently, there is a frequently recurring struggle for existence, it follows that any being, if it vary however slightly in any manner profitable to itself, under the complex and sometimes varying conditions of life, will have a better chance of surviving, and thus be naturally selected. From the strong principle of inheritance, any selected variety will tend to propagate its new and modified form.

In other words, organisms do not reproduce identical to each others, but with small, random, mutations. If it appears that this mutation represents an advantage in its environment, the mutated organism is more likely than others to survive and to reproduce, and the useful mutation is thus likely to appear in the next generation.

Evolution needs heredity in order to pass characteristics of parents to the children, but also mutation to explore new ways of living. Those two mechanisms

are contradictory, and the efficiency of evolution depends on the balance between them.

The idea behind genetic algorithms is to imitate the mechanisms of the evolution of species, which moves, generations after generations, towards the attributes that best fit their environment. As explained in [10], in multi-objective genetic algorithms, a first set of solutions is randomly created (it is the first generation). Then two mechanisms of evolution are reproduced: a phase of selection, where the best solutions are promoted, and a phase of reproduction, where solutions are combined in order to create a new set of solutions (the second generation). Selection and reproduction are then applied to the new generation and this operation is repeated in order to converge to the Pareto front.

Both selection and reproduction can be achieved by multiple ways, and the convergence of generation towards the Pareto front will depend on the algorithms chosen for both reproduction and replacement. Nevertheless, general observations can be made.

3.1 Genotype and phenotype

When working with genetic algorithm, an analogy for decision variables and objective functions can be made with genetic. In genetic, two concepts are used: the genotype and the phenotype. The genotype is the genetic composition of an organism, i.e. the sequence of amino-acids that constitute his chromosomes. The phenotype is the set of observable characteristics of an organism or an individual: the colour, the size, the shape, etc.

In genetic algorithm, the decision variable vector (i.e. the set of all decision variables) is sometimes called the genotype of solution, and the objective vector is then called the phenotype of the solution [1]. In genetics, the genotype of an organism is not sufficient to predict his phenotype, but in genetic algorithm, the phenotype of a solution depends on its genotype by the mean of the objective functions. A great advantage of genetic algorithm is that objective function does not have to be differentiable. Contrary to conventional methods of optimization, genetic algorithms are applicable even when the gradient of objective function is not known [1].

3.1.1 Representation of the decision vector

In order to be manipulated by the genetic algorithms, the decision variables vector has to be encoded into a computer. In nature, the genotype is coded trough means of four nucleobases (adenine, cytosine, guanine and thymine) whose position and sequence in the ADN influence the phenotype. In genetic algorithms, the decision variables vector is usually binary-coded. The values of the decision variables are translated into a sequence of ones and zeros. Binary-coded genotype are easy to manipulate but does not allow to code continuous decision variables. When a discrete approximation of the search space is acceptable, binary-coded genotype may be used, but sometimes the genotype has to be

real-coded, i.e. translated into a sequence of real numbers [13]. The type of coding influences the mechanisms of the genetic algorithm.

3.2 Ranking

The step of selection implies the choice of a method to compare solutions. The Pareto dominance criterion is used to rank the existing solutions as described in [1]. The ranking method is iterative: the non-dominated solutions of the population are assigned the highest rank, they are then removed from the population, and non-dominated solutions of the new population are assigned to the second rank. The process continues until all solutions are ranked.

Of course, many solutions have the same rank: they are not comparable regarding to the Pareto dominance. Some algorithms add other criterions in order to compare solutions of the same rank, as in the weighted Pareto-based multi-objective genetic algorithm, which will be explained in section 5.2.

3.3 Reproduction

The role of the reproduction step is to generate new solutions from existing ones. For the algorithm to work, the reproduction mechanism has to imitate heredity, i.e. the new solutions as to be similar to the former ones in order to conserve the improvements made generations after generations. The reproduction is composed of two mechanisms, derived from genetic : crossover and mutation.

3.3.1 Crossover

In sexual reproduction, the genes of the offspring are obtained by mixing together the genes of the parents. This mechanism, called crossover, is at the heart of heredity because if the offspring and its parents share a similar genotype, the offspring will probably inherit most aspect of its parents' phenotype.

In genetic algorithm, the crossover mechanism consist in choosing two parents in the population of solutions, and to create new solutions (the offspring) by combining values of the parents' decision vector.

Binary and real-coded algorithms need different implementations of crossover. Considering two decision vectors $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ as the parents, and $x' = (x'_1, x'_2, \ldots, x'_n)$ and $y' = (y'_1, y'_2, \ldots, y'_n)$ as the offspring. In a binary-coded genetic algorithm, the x_i, y_i, x'_i and y'_i are binary numbers, and the cross-over mechanism consists in swapping values of the decision vector in order to create two complementary children. As described in [10], the simplest way of performing a crossover is to randomly choose one bit in the decision vector as the crossover site, and to exchange every bits located after the crossover site. More complex crossover technique exists, such as using multiple crossover sites, or choosing the crossover sites to avoid breaking sequence of bits that seems optimal, but in this report as in [1], the simple one-site crossover will be used.

In real-coded genetic algorithm, the x_i, y_i, x'_i and y'_i are real numbers, and the children can be obtained by applying this rule [13]:

$$x'_i = \alpha_i x_i + (1 - \alpha_i) y_i \tag{3}$$

$$y'_i = \alpha_i y_i + (1 - \alpha_i) x_i \tag{4}$$

Where α_i is a now a random real comprise in a bounded interval (e.g. [-0.5 1.5]). However this definition is not the only possibility, as explained in [13].

3.3.2 Mutation

The mutation mechanism consist in randomly modifying some values of the children's decision vector. The probability of mutations has to be carefully chosen, because a too low rate of mutations limits the range of solution explored by the genetic algorithm, but a too high rate of mutations transforms the genetic algorithm in a random search.

In binary-coded algorithms, the mutation consists in giving a certain probability for each bit of the genotype for changing from one to zero and vice-versa. In real-coded algorithms the elements of the decision variables vector cannot simply change from one value to another. The mutation for real-coded algorithms is thus more complex than for binary-coded algorithms, and numerous implementations exist, as described in [14].

3.4 Simple multi-objective genetic algorithm

In [4] is presented a genetic algorithm that will be used as a base for the implementation of the three guiding methods. It is one of the simplest implementation of MOGA, but has proved to be efficient [15]. This method proceeds as follow:

1. Randomly create an initial population of solution vectors.
2. Reproduction: two parents are randomly selected in the population and are crossed to obtain two children.
3. Replacement: the four solutions are sorted according to their rank, and the two best are kept in the population.
4. Repeat the the step 2 and 3 on the new population until a maximum number of computation is reached.

4 Concrete example: expansion of an electrical network

In order to illustrate the use of genetic algorithms, a simplified problem of network expansion will be considered. To keep the problem simple, only two characteristics of the network will be optimized:

- the reliability efficiency, which indicates the average reliability of the connection between each nodes of the network,
- the cost of the network.

The goal of the problem will be to find where to add new transmission lines in the existing IEEE RTS 96 network in order to maximize the reliability efficiency of the network, while minimizing its cost.

The transmission network system IEEE RTS 96 described in [1] is a network consisting in N=24 nodes, interconnected by K = 34 links. The network is composed of two sub-network working at different voltage (138 kV and 230 kV) linked by 230/138 kV transformers. The list of the nodes, and the voltage of each node is given in table 4 in appendix A. The network can be represented by a NxN matrix, called the adjacency matrix [16], where the aij element is 1 when there is connection between the nodes i and j, and 0 when there is no direct connection. The adjacency matrix is shown in appendix A. The failure rate λ_{ij} is the average number breakdown per year. If the failure probability is uniformly distributed over time, the probability that the connection between the nodes i and j is still working after time T is given by $p_{ij} = e^{-\lambda_{ij}T}$ [17]. The failure rate λ_{ij} and the reliability p_{ij} (calculated for T = 1 year) of each links is given in table 5 in appendix A.

The tests were performed using Matlab. The code is visible in appendix B.

4.1 Decision vector

The Decision vector is the genotype of the solution: it indicates which lines are added, and what are their failure rates. The failure rate is a priori a real number. To simplify the treatment only three possible failure rates will be considered, allowing using a binary representation of the decision vector, rather than a real representation. The three failure rates are chosen as the maximum, the minimum and the mean values of the failure rates of existing connections in [1]:

- $\lambda_1 = 0.2267$ outages/yr
- $\lambda_2 = 0.374$ outages/yr
- $\lambda_3 = 0.54$ outages/yr

The failure rate of the connections between nodes 9, 10, 11 and 12 are inferior to 0.2267 outages a year, but has not been considered here because they are not transmission lines, but transformers between 138kV and 230kV nodes. For the expansion of this network, only new transmission lines between nodes at the same voltage will be added. The number of new possible connections is thus 107. Each new connection will be represented in the decision vector by two bits, and will be interpreted as follow:

- 00 means no new line

- 01 means a new transmission line with λ_1 failure rate
- 10 means a new transmission line with λ_2 failure rate
- 11 means a new transmission line with λ_3 failure rate

The decision vector is thus a 214 bits vector where the bits 2*i-1 and 2*i code the failure rate or the absence of the i^{th} new line.

4.2 Phenotype and objective functions

The objective vector is the phenotype of a solution, i.e. the set of characteristics that need to be optimized. In this case, the characteristics are the reliability of the network and the cost of the expansion.

Calculate the real cost of the expansion would be very difficult, but it can be considered that the cost of a new line will be inversely proportional to its failure rate, which lead to the following definition of the network cost given in [1], in an arbitrary monetary unit :

$$C = \sum_{i,j \in N, i \neq j} \frac{1}{\lambda_{ij}} \tag{5}$$

This definition leads to a cost of 602 for the initial network, and of 1074 for the network where every new line has been added with the maximum reliability. Those costs represent the maximum and the minimum costs of the network. In [1], the minimum and maximum cost found are 332 and 804. Those differences are unexplained, but do not have significant impact, since the absolute value of the cost does not have a real meaning.

If γ_{nm} is a path between the nodes n and m, the reliability of the path is the product of the reliability of the links composing the path γ_{nm}. The reliability efficiency ϵ between two nodes is defined as the reliability of the best path between those two nodes [1].

$$\epsilon_{nm} = max_{\gamma_{nm}} \left(\prod_{ij \in \gamma_{nm}} p_{ij} \right) \tag{6}$$

The reliability efficiency of the network can then be defined as [1]

$$E = \frac{1}{N(N-1)} \sum_{n,m \in N, n \neq m} \epsilon_{nm} \tag{7}$$

The optimization consists in finding which transmission lines to add in order to minimize the cost C and maximize the reliability efficiency E.

A Matlab code for computing the reliability efficiency and the cost of a network is shown in appendix B.1.

4.3 Initial population

The initial population is simply obtained by creating decision vectors where each bit has an equal chance to be 0 or 1. The initial population has to comport enough elements to ensure the diversity of the solution. The number of elements in the initial population has to be carefully chosen. The initial population must include enough elements to be a representative sample of the search space, but a big population will slow down the convergence of the algorithm. In the following tests, the number of chromosomes in the initial population will be 200, as used in [1]. This value was probably found by trial and error.

A Matlab code for generating the initial population is shown in appendix B.2.

4.4 Reproduction

Two parents are picked in the population. One of the 214 bits of the decision vector is randomly chosen to be the crossover site. Two new decision vectors are obtained by swapping the bits value after the crossover site in the decision vectors of the parents. Then, for each bit of the two new decision vectors, an inversion of the bit value has a 0.001 probability to be performed, as used in [1].

The mutation probability and the number of crossover sites are kept low to ensure that the offspring's objective vectors are similar to the parents' objective vectors.

A Matlab code for crossover and mutation is shown in appendix B.3.

4.5 Replacement

The replacement consists in choosing which decision variables vectors have to be kept in the population. The four decision vectors implied in the reproduction (the two parents and the two children) are sorted according to their rank (section 3.2) and the two best decision variables vectors are kept in the population.

The process of reproduction may lead to duplicate solutions. To avoid the rapid decay of diversity of the population, the two best solutions are compared before replacement to ensure that they are not identical. In theory, two or more different genotype could have the same phenotype (the opposite is not true), but in practice, it is a common approximation to consider that different phenotypes imply different genotypes [15]. This is a very useful approximation when duplicate solution are looked for, because the size of the decision vector is much bigger than the size of the objective vector, it is consequently faster to identify duplicates by comparing the objective vectors than the decision vectors.

A Matlab code for ranking and replacement is shown in appendix B.4.

4.6 Results

This algorithm has been used to perform 10^5 reproduction on an initial population of 200 elements. Figure 1 compare the initial population (randomly

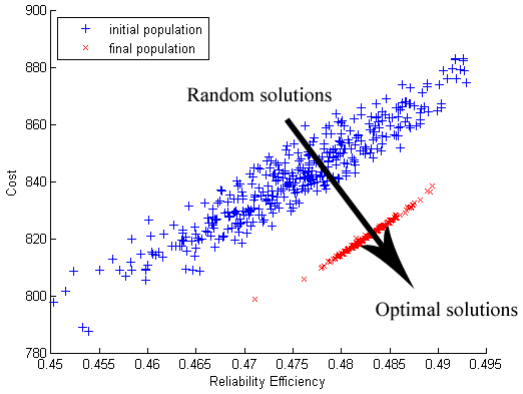

Figure 1: Convergence of the genetic algorithm.

generated) with the final population. A clear shift towards solutions with low cost and high efficiency can be observed, as expected from the algorithm. Figure 2 shows the non-dominated solutions found, as well as the initial network (representing the minimum cost) and the network obtained by adding the best transmission line between each nodes (resulting in the most reliable network). The run led to 85 Non-dominated solutions, with an average of 78 new connections by solution. Yet installing 78 new transmission lines implies a too big investment to be a practical solution. The interesting solutions are those that improve the network efficiency for a small number of new connections. The genetic algorithm can be modified to guide the convergence towards the solutions with few new transmission lines. Three different approaches will be presented here: act on the initial population (section 5.1), indicate an order of preference in the different objectives (section 5.2) and modify the concept pareto dominance (section 5.3).

5 Guided convergence

The genetic algorithms share the advantages of natural selection (simplicity and convergence), but also its flaws: long run-time and non uniform exploration of the search space.

Figure 3 shows the time needed to reach a certain number of non-dominated solutions in a simple, two-objective, problem. The data are obtained from [15], and it can be observed that the run time rises dramatically when the number of non-dominated solutions increases.

Another disadvantage of GA is that premature convergence may lead to a partial exploration of the Pareto front, ignoring possibly interesting solutions.

On the other hand, often only few solutions of the Pareto front are rele-

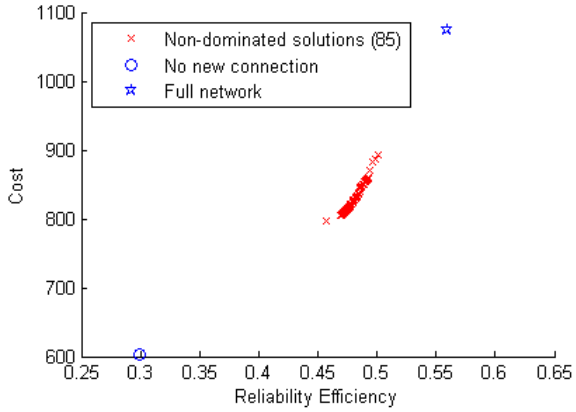

Figure 2: Non-dominated solutions reached by a initial population of 200 elements after 10^5 reproductions.

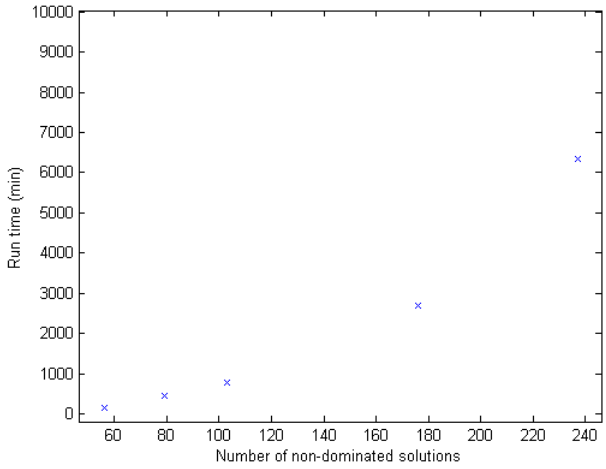

Figure 3: Run-time in fonction of the number of non-dominated solutions found.

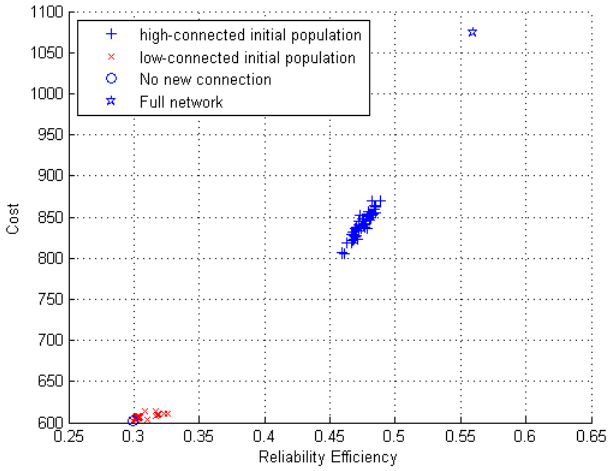

Figure 4: Comparison of initial populations

vant. For example, an optimum in quality can represent an unrealistic cost of production. Those unrealistic solutions are the fruit of useless calculation time, which could be spare by guiding the convergence towards the relevant part of the Pareto front. Three methods will be tested that permit to promote a certain area of the Pareto front, without having to choose a precise range of acceptable solutions.

The first method is to bias the generation of the first set of solutions. The second method, called weighted Pareto optimization method (WP-MOGA), allow the decision maker to attribute different weights to each objective. The third method, called guided multi-objective genetic algorithm (G-MOGA), consist in choosing minimal and maximal trade-offs between pairs of objectives by modifying the concept of Pareto dominance.

The implementation and results of each method will be described and compared after the precise description of a simple MOGA.

5.1 Biased initial population

The high connected networks obtained via the preceding algorithm reflect the characteristics of the initial population. Indeed, for the generation of the initial population, an identical probability is given for the presence of a new transmission line with a failure rate λ_1, λ_2, λ_3, or for the absence of a new line. The result is a 0.75 probability of adding a new line. As there are 107 possible new lines, the initial population will have an average of 80 new lines, which is similar to the average number of new lines in the non-dominated solutions (78).

As the initial population seems to have a great influence on the results of

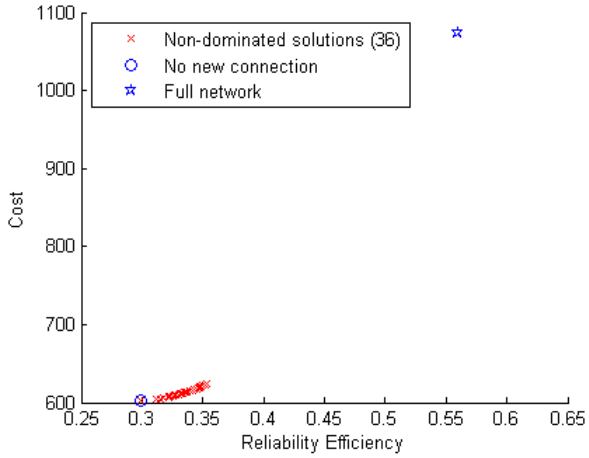

Figure 5: Non-dominated solutions reached by a initial population (favoring the absence of new transmission lines) of 200 elements after 10^5 reproductions.

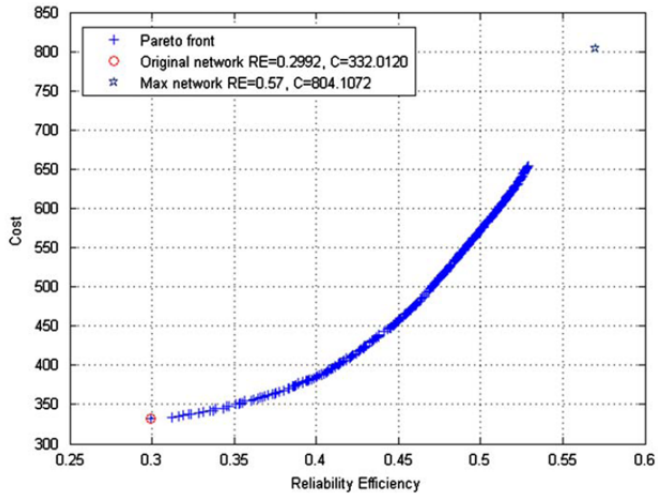

Figure 6: Non-dominated solutions reached with an initial population favoring the absence of new transmission lines in [1].

the research, the generation of the initial population could be biased to explore a preferred region of the search space. This is not always feasible, because the relations between decision and objective vectors can be very complicated. In this problem however, there is a direct relation between the number of new lines and the cost of the network expansion. The research can be easily guided towards low connected networks by giving a high probability of having no new connections in the initial population.

A Matlab code used for generating a biased initial population in shown in appendix B.2.1. Figure 4 shows the initial populations generated with an even probability for the strings 00, 01, 10 and 11, or with a probability of 0.99 for the string 00 and an even probability for the strings 01, 10 and 11. Those two initial populations are clearly different, and lead to the exploration of two distinct area of the search space.

Figure 5 shows the non-dominated solutions found after performing 10^5 reproductions on an initial population of 200 elements. 36 non-dominated solutions where found, with an average of 5 new transmission lines.

The research was efficiently guided, demonstrating a strong dependence between the initial population and the explored area of the search space. However, a similar test was made in [1], resulting in a much weaker convergence, as shows figure 6. This difference may come from variation in the implementation of the genetic algorithm.

Anyway, it must be kept in mind that acting on the initial population, although very simple and straightforward here, can become really challenging as the objective functions grow in complexity. Two other methods will be studied, both presenting the advantage of being applicable regardless of the problem considered.

5.2 Weighted Pareto-based multi-objective algorithm

During the step of replacement, parents and their children are compared according to the Pareto dominance principle. It is a natural way of comparing solutions, but it has the disadvantage of leaving most of the solution incomparable. Indeed, as soon as one solution as a higher cost, but also a higher reliability than another solution, those solutions cannot be ranked using the Pareto dominance principle. The idea of weighted Pareto-based multi-objective algorithm (WP-MOGA) is to use another ranking method to compare non-dominated solutions, favouring the preferred objective.

As described in [4], a weight $w_i, i \in \{1, 2, \ldots, n\}$ is assigned to the n objectives, representing the relative importance of each objective. For each solution, the weighted sum $\sum_{i=1}^{n} w_i f_i(U)$ is calculated, where the f_i are the n objective functions, and U is the decision variables vector of the solution. Non-dominated solutions are then compared according to this weighted sum. It is important to notice that the Pareto dominance principle is still the most important comparison criterion, and is used before comparing the weighted sum of the objective values. If the solutions where only compared according to the weighted sum, the problem would not be multi-objective anymore, but would rather be a new

single objective optimization problem where the new objective is defined as the weighted sum of former objectives.

In the network optimization case, the two objective functions are the cost C and the reliability efficiency E. In order to advantage improvements that requires a low investment, the cost will be assigned a weight of 0.9, while the reliability efficiency will be assigned a weight of 0.1, as in [1].

Using weighted sum of the objectives can only be done after normalizing the objective values. Indeed, giving a weight of 0.99 to an objective whose value range from 1 to 5 and a weight of 0.01 to an objective comprise between 10^5 and 10^7 will still promote the second objective because of the scaling effect. The normalized objective values are usually defined as follow:

$$f_i^* = \frac{f_i - f_i^{min}}{f_i^{max} - f_i^{min}} \tag{8}$$

Since the cost has to be minimized and not maximized, the definition 8 is not appropriate. In order to normalize the cost while promoting the minimum cost, the following definition will be used for the normalization:

$$C^* = \frac{C^{max} - C}{C^{max} - C^{min}} \tag{9}$$

Normalizing can be difficult if the exact range of the objective value is not known. It can be achieved by using the effective range method [4] which dynamically normalizes the objectives value using the range defined by the minimum and the maximum values of the objective for the solutions found so far.

In this case however, the possible ranges of the objectives' value are known: the minimum in cost and reliability are given by the original network, and the maximum in cost and reliability are given by the network where each pair of nodes have been linked with the best transmission line. The reliability is comprised between 0.2992 and 0.5598, while the cost ranges from 602.4 to 1074 in the arbitrary monetary unit.

A Matlab code computing the weighted sum of the cost and reliability efficiency of the network is shown in appendix B.6. The non-dominated solutions obtained with this method are shown in figure 7. The guiding towards low-connected network was not efficient: the average number of new link in the non-dominated solutions is 44, which is still unrealistically high, despite a clear shift towards low-connected network. It appears that adding a complementary comparison criterion to the Pareto dominance principle is too weak to guide the convergence. The next method will directly modify the concept of Pareto dominance to more efficiently guide the evolution of the population.

5.3 Guided multi-objective genetic algorithm

The WP-MOGA tried to guide the population towards the preferred area of the Pareto front by translating the relative importance of objectives in weight. However, it may be difficult to choose the weight of each objective since this notion is a little bit abstract. Indicating reasonable trades-off between each

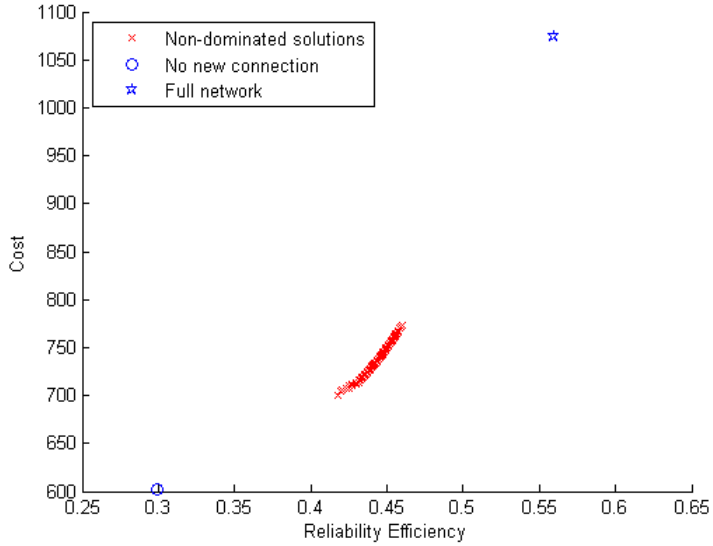

Figure 7: Non-dominated solutions reached by a initial population of 200 elements after 10^5 reproductions by the Weighted Pareto-based method.

objective may be easier. A reasonable trade-off between the objectives A and B, is the minimum improvement in A that worth a unitary decrease in B, and vice versa. For example, increasing the speed of a car 10 km/h may worth spending one more litter of gasoline every hundred kilometers, but spending fifty more litter to win another 10 km/h would be unrealistic.

As explained in [18], the principle of Pareto dominance may be modified to incorporate the reasonable trade-off. In a two-objective optimization problem where the objective functions O1 and O2 have to be maximized, if a unitary decrease in objective O1 worth minimum an improvement of a_{21} units in objective O2, and a unitary decrease in objective O2 worth minimum an improvement of a_{12} units in objective O1, then a solution A dominates (or is equal to) the solution B if the two following inequalities are satisfied:

$$O1(A) + a_{12}O2(A) \geq O1(B) + a_{12}O2(B) \tag{10}$$

$$O2(A) + a_{21}O1(A) \geq O2(B) + a_{21}O1(B) \tag{11}$$

In the plan (O1, O2), the equation 10 define a half-plan delimited by the line of slope $-\frac{1}{a_{12}}$ passing through the point (O1(A), O2(A)). The equation 11 define a half-plan delimited by the line of slope $-a_{21}$ passing through the point (O1(A), O2(A)). Those two lines delimit the area dominated by the solution A. Figure 8 illustrate de difference between the initial Pareto dominance and the new dominance criterion.

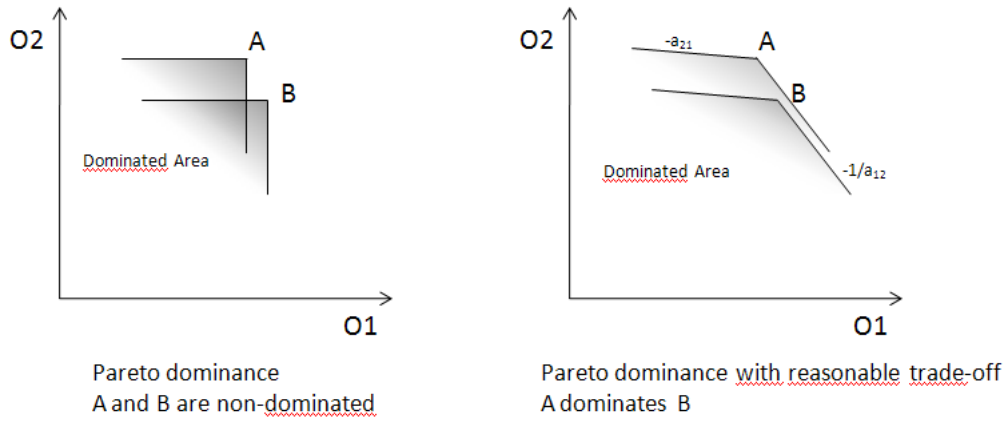

Figure 8: Visualization of the Pareto dominance principle in a two objectives problem, with and without reasonable trade-off.

In the network optimization problem, the first objective is the reliability efficiency E, and the second objective is the cost C. As the cost has to be minimized and not maximized, the definition of dominance has to be slightly modified, A dominating B iff:

$$E(A) - \frac{C(A)}{a_{12}} \geq E(B) - \frac{C(B)}{a_{12}} \tag{12}$$

$$C(A) - a_{21}E(A) \leq C(B) - a_{21}E(B) \tag{13}$$

In order to guide the convergence towards solution with a low cost, the trade-off parameters will be the following: $a_{12} = 331.3157$ and $a_{21} = 0$. Those trade-off have been found by trial and error in [1]. This means that any improvement in the cost is valuable, regardless of the decrease in reliability ($a_{21} = 0$), but that a increase of 331.3157 of the cost has to be compensate by an improvement of at least 1 in the reliability efficiency.

A Matlab code for comparing two solutions according to the Pareto dominance principle, with or without reasonable trades-off, is given in appendix B.5. Figure 9 shows the non-dominated solutions found after performing 10^5 reproductions on an initial population of 200 elements. The computation led to 11 non-dominated solutions. The guiding was really efficient, leading an initial population with an average of 80 new lines to final population with an average of 3 new transmission lines. This method can be used in any multi-objective problem, however, the number of trades-off that have to be chosen increase rapidly with the number of objective ($(n^2 - n)/2$ trades-off for n objectives [4]) and it may become difficult to chose all of them.

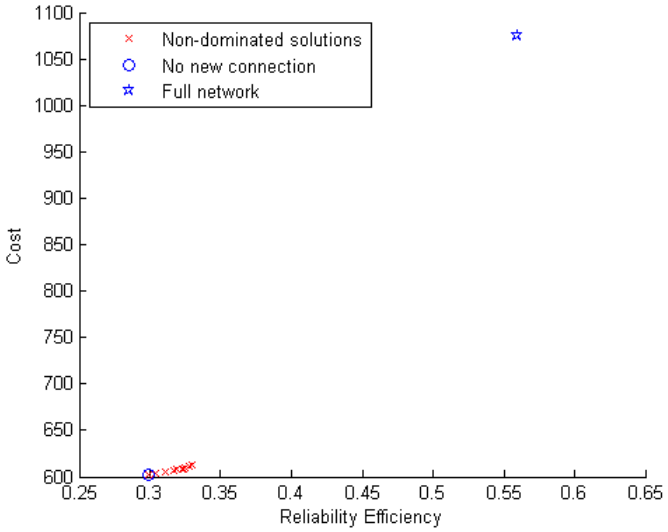

Figure 9: Non-dominated solutions reached by the guided multi-objective genetic algorithm with an initial population of 200 elements after 10^5 reproductions.

5.4 Comparison

In the search for optimal extension of a transmission network, three methods were tested to guide the convergence of the evolutionary algorithm towards solutions that implies a low investment cost:

1. Biasing the generation of the initial population by favouring solutions with few new transmission lines because they represent a low investment cost.
2. Using the WP-MOGA which compares the solutions according to the weighted sum of the objectives values (reliability efficiency and cost) with a higher weight given to the cost.
3. Using the G-MOGA which modifies the concept of Pareto dominance by including reasonable trades-off between cost and reliability efficiency.

The WP-MOGA showed little efficiency in guiding the population towards solution with a low cost. The biased initial population as well as the G-MOGA efficiently drove the population towards the desired region of the Pareto front, and they both have advantages and disadvantages.

- Acting on the initial population was the easiest way to guide the convergence in the case of the network extension, but this method may be

$p\{00\} = 0.99$; $p\{10\} = 0.0033$; $p\{01\} = 0.0033$; $p\{11\} = 0.0033$		G-MOGA	
Reliability Efficiency	Cost	Reliability Efficiency	Cost
0.3120	333.9	0.3072	337.6
0.3162	334.7	0.3168	339.4
0.3190	336.4	0.3186	339.4
0.3192	336.5	0.3187	339.4
0.3223	337.4	0.3193	339.4

Table 1: The five solutions on the Pareto front obtained by the MOGA search with $p\{00\} = 0.99$ and by the G-MOGA.

Nb of lines	failure rate	Connected nodes	Reliability efficiency	Cost
1	λ_3	(11,21)	0.3120	333.9
1	λ_2	(12,21)	0.3162	334.7
1	λ_1	(12,21)	0.3190	336.4
2	λ_2, λ_3	(11,16), (11,18)	0.3192	336.5
2	λ_2, λ_2	(11,16), (12,17)	0.3223	337.4

Table 2: The five solutions of lowest cost on the Pareto front obtained by the MOGA search with $p\{00\} = 0.99$.

difficult to apply to other multi-objective problems. Indeed, in this case, a clear relation existed between the number of new transmission lines (which is a decision variable) and the cost (the objective), but in other problems the objective function, which link the decision variables to the objectives, are more complex and it may be difficult act on the initial population.

- The guided-MOGA has the advantage of being usable in any kind of problem, regardless of the complexity of the objective functions, but in optimization problems with more than two or three objectives, the great number of trade-off that has to be chosen can make implementation of the algorithm difficult.

6 Interpretation of results

Both guided multi-objective genetic algorithm and genetic algorithm in which the initial population favours the absence of new lines found solutions with a low investment cost. The five solutions of lowest cost found by those two methods are shown in table 1 from [1]. The solutions found with $p\{00\} = 0.99$ have a lower cost and a higher reliability efficiency than those found with the G-MOGA, they present thus a greater interest.

The five best solutions added one or two transmission lines and present clear similarities, as seen in table 2. The new lines are always located in the 230kV

part of the network. The reliability efficiency of the sub-network composed of the node with a 230 kV voltage is 0.3269 while the reliability efficiency of the low-voltage sub-network is 0.3958. It is logic to see that the first improvements are made in the less reliable part of the network.

The nodes 11, 12, 16 and 21 appear in most solutions. The importance of those nodes in the network can be evaluated using four parameters called Centrality Measures [11]:

1. The reliability degree centrality C^{rD} which depends on the number of immediate neighbours of the node and on the reliability of the lines connected to the node.
2. The reliability closeness centrality C^{rC} which measures the average closeness of the node to all the other nodes of the network.
3. The reliability betweenness centrality C^{rB} which gives importance to nodes that are traversed by a lot of most reliable path.
4. The reliability information centrality C^{rI} which indicates the effect of removing a certain node on the global reliability of the network.

Those four centrality measures are detailed hereafter.

6.1 Reliability degree centrality

Intuition indicates that a node is more important if it has numerous connections with other nodes. The reliability degree centrality count the number of transmission lines leaving the node, taking into account the reliability of those links. The C^{rD} of the node i is defined in [11] as:

$$C_i^{rD} = \frac{k_i \sum_{j \in G} p_{ij}}{(N-1)^2} \tag{14}$$

Where k_i is the degree of the node i, i.e. the number of transmission lines connected to the node i. Since the maximum degree of a node in a network composed of N nodes is N-1, $\frac{k_i}{(N-1)}$ is the normalized degree of the node i.

p_{ij} is the reliability of the line connecting nodes i and j, as defined in section 4 and G is the set of node of the network. $p_{ij} \in [0,1]$ with $p_{ij} = 1$ meaning perfectly reliable edge, which implies:

$$\sum_{j \in G} p_{ij} \in [0, N-1] \tag{15}$$

$$\Rightarrow \frac{\sum_{j \in G} p_{ij}}{N-1} \in [0,1] \tag{16}$$

The reliability degree centrality is thus normalized in the interval $[0,1]$. The running time required for computing C^{rD} for all nodes is $O(N)$ [11]. The five nodes with the highest reliability degree are, in a decreasing order: 10, 9, 11, 12 and 16.

6.2 Reliability closeness centrality

The reliability closeness centrality assumes that a node is important if it has reliable connexion with every other nodes of the network. ϵ_{nm} was defined in section 4.2 as the reliability of the most reliable path between the nodes n and m. d_{nm}, defined as the inverse of ϵ_{nm}, can be interpreted as the distance between nodes n and m. This is not a topological distance, since there are no information about the position of different nodes, but a distance in term of reliability. The shortest distance is obtained when every edge of the path linking n to m has a reliability of 1. If there is no connection between two nodes, the distance becomes infinite. The C^{rC} of the node i is defined in [11] as:

$$C_i^{rC} = \frac{N-1}{\sum_{j \neq i} d_{ij}} \tag{17}$$

C^{rC} is normalized in the range $[0, 1]$. The running time required for computing C^{rD} for all nodes is $O(N^3)$ [11].

The five nodes with the highest reliability degree are, in a decreasing order: 11, 9, 10, 3, 12. Those nodes connect the high voltage part of the network with the low voltage part, and thus naturally occupy a central position in the network, which explain their high value of C^{rC}.

6.3 Reliability betweenness centrality

With the notion of most reliable path between two nodes comes the idea that a node is important if it is a part of many of the most reliable paths in the network. The C^{rB} of the node i is calculated by finding the most reliable path between each pairs of nodes (except node i) and counting the amount of time the node i is traversed by one of these paths. The maximum amount of most reliable paths that may traverse a node is $\frac{(N-1)(N-2)}{2}$ (there are N-1 nodes considered and thus $\frac{(N-1)(N-2)}{2}$ path to connect them). The C^{rB} is normalized in the range [0, 1] by dividing it by $\frac{(N-1)(N-2)}{2}$ (or by $(N-1)(N-2)$ if each path is counted twice, as in equation 18). The C^{rB} has to take into account the fact that multiple equivalent paths may exist between two nodes. If n_{jk} is the number of most reliable paths between the nodes j and k and $n_{jk}(i)$ the number of most reliable paths between the nodes j and k that traverse the node i, the influence of finding the node i in a most reliable path between the nodes j and k is weighted $\frac{n_{jk}(i)}{n_{jk}}$. The reliability betweenes centrality of the node i is defined in [11] as:

$$C_i^{rB} = \frac{1}{(N-1)(N-2)} \sum_{j,k \in G, j \neq k \neq i} \frac{n_{jk}(i)}{n_{jk}} \tag{18}$$

The running required for computing C^{rB} for all nodes is $O(N^3)$ [11]. The five nodes with the highest reliability degree are, in a decreasing order: 16, 9, 14, 24, 10. Nodes 11 and 12 are found in the sixth and ninth position.

Rank	C^{rD}	C^{rR}	C^{rB}	C^{rI}
1	11	11	16	16
2	12	12	14	11
3	16	14	24	12
4	21	16	11	15
5	15	24	15	24
6	17, 23	23	12	14
7		13	21, 23	23
8	13	20		19
9	24	19	13, 19, 20	20
10	20	17		21

Table 3: Reliability centrality measures of the high-voltage nodes

6.4 Reliability information centrality

Finally, an interesting way to indicate the importance of a certain node in a network is to observe the effect of removing this node on the global reliability of the network. The reliability information centrality is defined in [11] as:

$$C_i^{rI} = \frac{E[G] - E[G'(i)]}{E[G]} \tag{19}$$

Where G'(i) is the network obtained by removing the node i from the original network G. E[G] is the reliability efficiency of the network G, as defined in section 4.2. Since E[G'(i)] is always inferior or equal to E[G] and superior to zero, the C^{rI} is normalized in the range [0, 1].

The running required for computing C^{rB} for all nodes is $O(N^4)$ [11]. The five nodes with the highest reliability degree are, in a decreasing order: 9, 10, 16, 11, 8. Nodes 12 is found in the seventh position.

6.5 Conclusion

The new nodes were added in the high-voltage part of the network, which has lower reliability efficiency than the low-voltage part of the network. The nodes of the high-voltage part were ranked according to the centrality measure. The ranks are shown in table 3. The nodes 11 and 12 have high rank for each centrality measure. This is partly due to their role as transformer between the low and high voltage parts of the network, which gives them a central position in the network.

7 Conclusion

In this report, the multi-objective optimization problems have been presented. Important concepts related to multi-objective problems such as decision vari-

ables and objective functions have been described in section 2.1. The notion of Pareto optimality and Pareto dominance have been presented in section 2.2 as a way of comparing solutions in a multi-objective problem.

The genetic algorithms, which are used to solve multi-objective optimization problems, were presented in section 3. Genetic algorithms are inspired by the natural process of evolution: a initial population of solutions is randomly generated, then those solutions are combined in order to create new solutions (section 3.3); the best solutions are kept in the population, and the process continues until optimal solutions are reached.

A genetic algorithm was use to study the extension of a power transmission network composed of 24 nodes and 34 edges operating at two different voltage levels: 138 kV and 230 kV. The objectives were to maximize the reliability efficiency and to minimize the cost, as described in section 4.2. The advantages and the flaws of genetic algorithms are summarized hereafter:

- The algorithm is easy to use, and can be implemented in almost any kind of multi-objective optimization problems.
- The algorithm converges towards the Pareto front.
- Genetic algorithm requires an important computation time, especially when the objective functions become complex and when a great amount of optimal solutions are looked for.
- The search space is not uniformly covered. In the case of the network extension, the solutions found by the genetic algorithm had an average of 80 new transmission lines, and solutions with few new lines were absent from the results. Yet solutions with few new lines are the most realistic because they represent a reasonable investment.

Three methods were presented to guide the convergence of the population towards the preferred solutions:

1. Biasing the generation of the initial population by favouring solutions with few new transmission lines because they represent a low investment cost. (section 5.1)
2. Using the WP-MOGA which compares the solutions according to the weighted sum of the objectives values (reliability efficiency and cost) with a higher weight given to the cost. (section 5.2)
3. Using the G-MOGA which modifies the concept of Pareto dominance by including reasonable trades-off between cost and reliability efficiency. (section 5.3)

Those methods were tested on the network extension problem in order to find solutions that represented a low investment cost. It appears that the WP-MOGA had little influence on the convergence of the population. Contrariwise,

modifying the initial population and setting minimal and maximal trade-off between cost and reliability efficiency both efficiently guide the population towards low cost solutions. The two methods present distinct advantages and flaws described in section 5.4. Modifying the initial population is easier and very efficient, but is only applicable to certain simple optimization problems. The G-MOGA is implementable in any kind of multi-objective problem, but tuning the different trades-off may be a complicated task, especially when numerous different objectives are taken into account.

The results obtained thanks to the genetic algorithm were then analyzed using the centrality measures described in section 6. Four parameters, called reliability degree centrality, reliability closeness centrality, reliability betweenness centrality and reliability information centrality, were used to indicate the relative importance of each node in the network.

1. The reliability degree centrality C^{rD} depends on the number of immediate neighbours of the node and on the reliability of the lines connected to the node.

2. The reliability closeness centrality C^{rC} measures the average closeness of the node to all the other nodes of the network.

3. The reliability betweenness centrality C^{rB} gives importance to nodes that are traversed by a lot of most reliable path.

4. The reliability information centrality C^{rI} indicates the effect of removing a certain node on the global reliability of the network.

The optimal solutions connected new transmission lines to the nodes 11 or 12, which have a high importance in the network according to the centrality measures. It also appears that every new links were located in the high-voltage part of the network, which have a lower reliability efficiency than the low-voltage part.

The analysis of the optimal extension of a power transmission network system was here greatly simplified in order to illustrate the genetic algorithms and the methods for guiding the research of solutions. The model of the network neglected important characteristics as noticed in [1]:

- the "electrical" length of a path which depends on the difficulty of transmission,

- the transmission of power is determined by physical laws and does not necessarily follow the most reliable paths,

- the direction of the power flow should be taken into account,

- the nodes represent different components with given capacities that should be considered in the analysis of the network.

Moreover, the improvement of the network does not necessarily implies adding new transmission lines, but could be performed by improving the existing installations. However, this simple example permit to illustrate the genetic algorithms and to study the possible ways of guiding the research of optimal solutions.

References

[1] F. Cadini, Enrico Zio, and C.A. Petrescu. Optimal expansion of an existing electrical power transmission network by multi-objective genetic algorithms. *Reliability Engineering and System Safety*, 95:173–181, 2010.

[2] Ajith Abraham and Lakhmi Jain. Evolutionary multiobjective optimization. In Lakhmi Jain, Xindong Wu, Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 1–6. Springer Berlin Heidelberg, 2005. 10.1007/1-84628-137-7_4.

[3] Eckart Zitzler, Lothar Thiele, Marco Laumans, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance assessment of multiobjective optimizers: ananalysis and review. *IEEE Trans Evol Comput*, 2003.

[4] Enrico Zio, P. Baraldi, and N. Pedroni. Optimal power system generations cheduling by multi-objective genetic algorithms with preferences. *Reliability Engineering and System Safety*, 94:432–444, 2009.

[5] Carlos Coello Coello. Recent trends in evolutionary multiobjective optimization. In Lakhmi Jain, Xindong Wu, Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 7–32. Springer Berlin Heidelberg, 2005. 10.1007/1-84628-137-7_2.

[6] P. Giuggioli Busacca, M. Marseguerra, and E. Zio. Multiobjective optimization by genetic algorithms: application to safety systems. *Reliability Engineering System Safety*, 72(1):59 – 74, 2001.

[7] Marzio Marseguerra, Enrico Zio, and Luca Podofillini. A multiobjective genetic algorithm approach to the optimization of the technical specifications of a nuclear safety system. *Reliability Engineering System Safety*, 84(1):87 – 99, 2004. Selected papers from ESREL 2002.

[8] M. A. Abido. A novel multiobjective evolutionary algorithm for environmental/economic power dispatch. *Electric Power Systems Research*, 65(1):71 – 81, 2003.

[9] Chi-Ming Lin and Mitsuo Gen. Multi-criteria human resource allocation for solving multistage combinatorial optimization problems using multiobjective hybrid genetic algorithm. *Expert Systems with Applications*, 34(4):2480 – 2490, 2008.

[10] Lech Józwiak and Adam Postula. Genetic engineering versus natural evolution: Genetic algorithms with deterministic operators. *Journal of Systems Architecture*, 48(1-3):99 – 112, 2002.

[11] F. Cadini, Enrico Zio, and C.A. Petrescu. Using centrality measures to rank the importance of the components of a complx network infrastructure. *Proceedings of CRITIS'08*, 2008.

[12] Charles R. Darwin. *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life*. London: John Murray, 1859.

[13] P. Kaelo and M.M. Ali. Integrated crossover rules in real coded genetic algorithms. *European Journal of Operational Research*, 176(1):60 – 68, 2007.

[14] Kusum Deep and Manoj Thakur. A new mutation operator for real coded genetic algorithms. *Applied Mathematics and Computation*, 193(1):211 – 213, 2007.

[15] Christine L. Mumford-Valenzuela. A simple approach to evolutionary multiobjective optimization. In Lakhmi Jain, Xindong Wu, Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 55–79. Springer Berlin Heidelberg, 2005. 10.1007/1-84628-137-7_4.

[16] Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Physical Review Letters*, 87(19), 2001.

[17] Yves De Smet. *MATH-H-204 Calcul des probabilités et statistiques*.

[18] Guidance in evolutionary multi-objective optimization. *Advances in Engineering Software*, 32(6):499 – 507, 2001.

A Characteristics of the network

Node	Voltage
1 to 10	138 kV
11 to 24	230 kV

Table 4: List of nodes and their voltage

$$a_{ij} = \begin{bmatrix}
0. & 1. & 1. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
1. & 0. & 0. & 1. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. \\
0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0 \\
0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 1. & 1. & 0. & 0. & 0. & 1. & 0. & 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 1. & 1. & 0. & 1. & 0. & 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 1. & 0. & 1. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & 1. & 0. & 1. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 1. & 0. & 0. & 1. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 1. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 1. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 1. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 1. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 1. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. \\
0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.
\end{bmatrix}$$

Figure 10: Adjacency matrix of the IEEE RTS 96 network

From node	To node	λ_{ij}	p_{ij}
1	2	0.24	0.786627861
1	3	0.51	0.600495579
1	5	0.33	0.718923733
2	4	0.39	0.677056874
2	6	0.48	0.618783392
3	9	0.38	0.683861409
3	24	0.02	0.980198673
4	9	0.36	0.697676326
5	10	0.34	0.711770323
6	10	0.33	0.718923733
7	8	0.30	0.740818221
8	9	0.44	0.644036421
8	10	0.44	0.644036421
9	11	0.02	0.980198673
9	12	0.02	0.980198673
10	11	0.02	0.980198673
10	12	0.02	0.980198673
11	13	0.40	0.670320046
11	14	0.39	0.677056874
12	13	0.40	0.670320046
12	23	0.52	0.594520548
13	23	0.49	0.612626394
14	16	0.38	0.683861409
15	16	0.33	0.718923733
15	21	0.2733	0.760864494
15	24	0.41	0.66365025
16	17	0.35	0.70468809
16	19	0.34	0.711770323
17	18	0.32	0.726149037
17	22	0.54	0.582748252
18	21	0.2333	0.791915963
19	20	0.2533	0.776234976
20	23	0.2267	0.797159894
21	22	0.45	0.637628152

Table 5: Failure rates and reliability of each connections in the network

B Matlab code

B.1 Objective functions

B.1.1 Reliability Efficiency

```
%Calculate the reliability efficiency of the network defined by the adjacency matrix A and
%the reliability matrix R.
function efficiency = network_efficiency(A, R)
    [costs paths] = dijkstra(A, 1./exp(-R));
    N = length(A);
    efficiency = 0;

    for n = 2:N
        for m = 1:n-1
            efficiency = efficiency + 1/costs(n, m);
        end
    end

    efficiency = 2*efficiency/(N*(N-1));
end
```

B.1.2 Cost

```
%Calculate the cost of the network defined by the adjacency matrix A and
%the reliability matrix R.
function cost = network_cost(A, R)
    cost = 0;
    N = length(A);
    for i = 2:N
        for j = 1:N-1
            if A(i, j) == 1
                cost = cost + 1/R(i, j);
            end
        end
    end
end
```

B.2 Initial population

```
%Generate an initial population of pop_size elements
%Need the adjacency matrix A and the voltage of each node (node_voltage(i)
%is the voltage of the ith node
function [population new_link_index] = initial_population(A, node_voltage, pop_size)
    N = length(A); % nbr of nodes
    n = 0; % nbr of possible new connections
    for i = 2:N
        for j = 1:i-1
            if (A(i, j) == 0 && node_voltage(i) == node_voltage(j))
                n = n+1;
            end
        end
```

```
13      end
14      new_link_index = zeros(n, 2);
15
16      n = 0;
17      for i = 2:N
18          for j = 1:i-1
19              if (A(i, j) == 0 && node_voltage(i) == node_voltage(j))
20                  n = n+1;
21                  new_link_index(n, :) = [i j];
22              end
23          end
24      end
25
26      population = round(rand(pop_size, 2*n));
27  end
```

B.2.1 Biased initial population

```
1   %Generate an initial population of pop_size elements. The
        probability of
2   %not adding a new line is 0.99.
3   %Need the adjacency matrix A and the voltage of each node (
        node_voltage(i)
4   %is the voltage of the ith node
5   function [population new_link_index] = bias_initial_population(A,
        node_voltage, pop_size)
6       N = length(A); % nbr of nodes
7       n = 0; % nbr of possible new connections
8       for i = 2:N
9           for j = 1:i-1
10              if (A(i, j) == 0 && node_voltage(i) == node_voltage(j))
11                  n = n+1;
12              end
13          end
14      end
15      new_link_index = zeros(n, 2);
16
17      n = 0;
18      for i = 2:N
19          for j = 1:i-1
20              if (A(i, j) == 0 && node_voltage(i) == node_voltage(j))
21                  n = n+1;
22                  new_link_index(n, :) = [i j];
23              end
24          end
25      end
26
27      population = zeros(pop_size, 2*n);
28      for i = 1:pop_size
29          for j = 1:n
30              if rand > 0.99
31                  r = ceil(rand*3);
32                  if r == 1
33                      population(i, 2*j-1:2*j) = [1 0];
34                  elseif r == 2
35                      population(i, 2*j-1:2*j) = [0 1];
36                  else
37                      population(i, 2*j-1:2*j) = [1 1];
```

```
38                  end
39              end
40          end
41      end
42  end
```

B.3 Reproduction

B.3.1 Crossover

```
1  %The crossover function recieve the population and two index x and
       y
2  %The function returns two chromosomes c1 and c2, obtained after
3  %performing a one-site crossover on the xth and yth elements of
       population
4  function [c1 c2] = crossover(population, x, y)
5      [pop_size chrom_nbr] = size(population);
6      c1 = population(x, :);
7      c2 = population(y, :);
8
9      cross_site = ceil(rand*chrom_nbr);
10
11     save = c1(cross_site:chrom_nbr);
12     c1(cross_site:chrom_nbr) = c2(cross_site:chrom_nbr);
13     c2(cross_site:chrom_nbr) = save;
14 end
```

B.3.2 Mutation

```
1  %mutation function recieve a binary vector "child"
2  % and performe a mutation from 1 to 0 and vice versa on each
       element
3  % of the vector with a probability of 0.001
4  function child = mutation(child)
5      mutation_prob = 0.001;
6      l = length(child);
7      for i = 1:l
8          if rand < mutation_prob
9              if child(i) == 1
10                 child(i) = 0;
11             else
12                 child(i) = 1;
13             end
14         end
15     end
16 end
```

B.4 Replacement

B.4.1 Ranking

```
1  %function rank_population return the rank of each elements of the
2  %population pop_objective_vectors
3  function rank = rank_population(pop_objective_vectors)
4
5  l = length(pop_objective_vectors);
6  rank = zeros(l, 1);
```

```
 7  unranked = l;
 8  actual_rank = 1;
 9  while unranked > 0
10      for i = 1:l
11          if rank(i) == 0
12              non_dominated = 1;
13              for j = 1:l
14                  if j ~= i && (rank(j) == 0 || rank(j) ==
                        actual_rank) && first_dominate(
                        pop_objective_vectors(j, :),
                        pop_objective_vectors(i, :))
15                      non_dominated = 0;
16                      break
17                  end
18              end
19              if non_dominated
20                  rank(i) = actual_rank;
21                  unranked = unranked - 1;
22              end
23          end
24      end
25      actual_rank = actual_rank + 1;
26  end
```

B.4.2 Replacement

```
 1  % replacement function compares the element population(x),
        population(y),
 2  % child1 and child2 according to their rank and keep the two best
        in the
 3  % population, at the position x and y. A is the adjacency matrix
        and R the
 4  % reliability matrix
 5  function [population objective_vector] = replacement(A, R,
        new_link_index, population, objective_vector, x, y, child_1,
        child_2)
 6
 7      scores = [0 0 0 0];
 8      l = length(population(x, :));
 9      cp = zeros(4, l+2);
10      cp(1, 1:l) = population(x, :);
11      cp(1, l+1:l+2) = objective_vector(x, :);
12      cp(2, 1:l) = population(y, :);
13      cp(2, l+1:l+2) = objective_vector(y, :);
14
15      [e1 c1] = calculate_objective_vector(A, R, new_link_index,
            child_1);
16      [e2 c2] = calculate_objective_vector(A, R, new_link_index,
            child_2);
17
18      cp(3, 1:l) = child_1;
19      cp(3, l+1:l+2) = [e1 c1];
20      cp(4, 1:l) = child_2;
21      cp(4, l+1:l+2) = [e2 c2];
22
23      temp_objective_vector = objective_vector;
24      temp_objective_vector(l+1, :) = [e1 c1];
25      temp_objective_vector(l+2, :) = [e2 c2];
```

```
26
27      rank = rank_population(temp_objective_vector);
28      scores = [-rank(x) -rank(y) -rank(l+1) -rank(l+2)];
29
30      [sorted index] = sort(scores, 'descend');
31
32      population(x, :) = cp(index(1), 1:l);
33      objective_vector(x, :) = cp(index(1), l+1:l+2);
34      for i = 2:4
35          if cp(index(i), l+1:l+2) ~= objective_vector(x, :)
36              population(y, :) = cp(index(i), 1:l);
37              objective_vector(y, :) = cp(index(i), l+1:l+2);
38              break;
39          end
40      end
41  end
```

B.5 Pareto dominance

```
1   %Recieve two objective vectors V1 and V2. the first element of the
        vector is
2   %the reliability efficiency and the second element is the cost. The
3   %function returns 1 if V1 dominates V2 and 0 otherwise.
4   function true = first_dominate(V1, V2)
5       first_win = 0;
6       second_win = 0;
7
8       if V1(1) > V2(1)
9           first_win = 1;
10      elseif V2(1) > V1(1)
11          second_win = 1;
12      end
13
14      if V1(2) < V2(2)
15          first_win = 1;
16      elseif V2(2) < V1(2)
17          second_win = 1;
18      end
19
20      if first_win == 1 && second_win == 0
21          true = 1;
22      else
23          true = 0;
24      end
25  end
```

B.5.1 Pareto dominance with reasonable trades-off

```
1   %Reciev two objective vectors V1 and V2. the first element of the
        vector is
2   %the reliability efficiency and the second element is the cost. The
3   %function returns 1 if V1 dominates V2 (with trades-off a12 and a21
        ) and 0 otherwise.
4   function true = first_guided_dominate(V1, V2)
5       first_win = 0;
6       second_win = 0;
7
```

```
8       a12 = 331.3157;
9       a21 = 0;
10
11      GV1 = [V1(1) - V1(2)/a12 V1(2) - a21*V1(1)];
12      GV2 = [V2(1) - V2(2)/a12 V2(2) - a21*V2(1)];
13
14      if GV1(1) > GV2(1)
15          first_win = 1;
16      elseif GV2(1) > GV1(1)
17          second_win = 1;
18      end
19
20      if GV1(2) < GV2(2)
21          first_win = 1;
22      elseif GV2(2) < GV1(2)
23          second_win = 1;
24      end
25
26      if first_win == 1 && second_win == 0
27          true = 1;
28      else
29          true = 0;
30      end
31  end
```

B.6 Weighted sum

```
1   %Calculate the weigthed sum of the normalized objective with a
        weight of 0.1 assigned
2   %to the first objective and a weight of 0.9 assigned to the second.
3   function sum = weighted_sum(objective_vector)
4
5   %Maximum and minimum values of cost and reliability efficiency
6   Mc = 1.0744e+003;
7   mc = 602.3941;
8   Me = 0.5598;
9   me = 0.2992;
10
11  sum = 0.1*(objective_vector(1)-me)/(Me-me) + 0.9*(Mc -
        objective_vector(2))/(Mc-mc);
12
13  end
```