Università degli Studi di Trieste

XIX Ciclo del Dottorato di Ricerca in Ingegneria e Architettura

An Evolutionary Multi-Objective Optimization Framework for Bi-level Problems

Settore scientifico-disciplinare
MAT/09 - RICERCA OPERATIVA

COORDINATORE:
Diego Micheli

DOTTORANDO:
Stefano Costanzo

SUPERVISORE DI TESI:
Lorenzo Castelli

CO-SUPERVISORE:
Alessandro Turco

ANNO ACCADEMICO 2015 - 2016

ABSTRACT

Genetic algorithms (GA) are stochastic optimization methods inspired by the evolutionist theory on the origin of species and natural selection. They are able to achieve good exploration of the solution space and accurate convergence toward the global optimal solution. GAs are highly modular and easily adaptable to specific real-world problems which makes them one of the most efficient available numerical optimization methods.

This work presents an optimization framework based on the Multi-Objective Genetic Algorithm for Structured Inputs (MOGASI) which combines modules and operators with specialized routines aimed at achieving enhanced performance on specific types of problems. MOGASI has dedicated methods for handling various types of data structures present in an optimization problem as well as a pre-processing phase aimed at restricting the problem domain and reducing problem complexity. It has been extensively tested against a set of benchmarks well-known in literature and compared to a selection of state-of-the-art GAs. Furthermore, the algorithm framework was extended and adapted to be applied to Bi-level Programming Problems (BPP). These are hierarchical optimization problems where the optimal solution of the bottom-level constitutes part of the top-level constraints. One of the most promising methods for handling BPPs with metaheuristics is the so-called "nested" approach. A framework extension is performed to support this kind of approach. This strategy and its effectiveness are shown on two real-world BPPs, both falling in the category of pricing problems.

The first application is the Network Pricing Problem (NPP) that concerns the setting of road network tolls by an authority that tries to maximize its profit whereas users traveling on the network try to minimize their costs. A set of instances is generated to compare the optimization results of an exact solver with the MOGASI bi-level nested approach and identify the problem sizes where the latter performs best.

The second application is the Peak-load Pricing (PLP) Problem. The PLP problem is aimed at investigating the possibilities for mitigating European air traffic congestion. The PLP problem is reformulated as a multi-objective BPP and solved with the MOGASI nested approach. The target is to modulate charges imposed on airspace users so as to redistribute air traffic at the European level. A large scale instance based on real air traffic data on the entire European airspace is solved. Results show that significant improvements in traffic distribution in terms of both schedule displacement and air space sector load can be achieved through this simple, en-route charge modulation scheme.

TABLE OF CONTENTS

LIST OF TABLES

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

Bi-level Programming Problems (BPPs) are hierarchical optimization problems where the optimal solution of the bottom level constitutes part of the top level constraints. A BPP can be viewed as a static version of the non-cooperative two-player game introduced by Stackelberg in the unbalanced economic market context. The first player, or leader, starts first trying to achieve his/her objective and to anticipate the possible responses of the second player, or follower. The follower also tries to achieve his/her objective and reacts to the leader's actions but without considering the consequences of his/her actions on the leader's objective. The choices available to both players are independent, so the leader's decision affects both the follower's objective and actions, and vice versa.

BPPs are challenging and are well-known and studied in literature, but they are also becoming more and more relevant in the industrial sector. In fact, many real-world problems in areas such as management, economic planning or engineering involve a hierarchical relationship between two decision levels. This poses several challenges, including randomness, two-level decision making, conflicting objectives and difficulties in searching for optimal solutions. Given the difficulties associated with solving BPPs, this field still lacks efficient solution methods. The issues connected to bi-level programming arise primarily from the nested structure of the problem. In recent decades there have been numerous attempts to develop dedicated algorithms, but most of the available methods can either only be applied to highly restricted classes of problems or are too computationally expensive, rendering them practically unfeasible in large-scale bi-level problems. For instance, the application of exact methods is generally unaffordable for very complex and large scale applications. As a

result, the tendency is to use specifically developed approximation algorithms with the aim of obtaining high-quality solutions in a reasonable amount of time that are hopefully closer to the optimal solution. Of these approximation algorithms, most promising is the family of heuristic and meta-heuristic algorithms.

Evolutionary heuristics have been used rather successfully to handle mathematical programming problems and applications that do not adhere to regularities such as continuity, differentiability or convexities. Owing to these characteristics, attempts have been made to solve Bi-level Optimization Problems (BOPs) using evolutionary heuristics since even simple (linear or quadratic) BOPs are intrinsically non-convex, non-differentiable and, at times, disconnected. More specifically, evolutionary algorithms for bi-level optimization were first proposed back in the 1990s. One of the first evolutionary algorithms for handling BOPs used a nested strategy, where the lower-level is handled with a linear programming method and the upper-level is solved with a Genetic Algorithm (GA). Nested strategies are indeed a popular approach for handling bi-level problems because for every upper-level decision variable configuration, a lower-level optimization task can be executed.

GAs, a class of evolutionary algorithms, are considered to be amongst the most efficient numerical strategies available because they are highly modular and easily adaptable to specific problems. They are particularly well-suited for complex non-linear problems or problems with multiple objectives which generate a set of Pareto optimal alternatives. Most of the GA bi-level techniques use the nested approach in which an outer algorithm handles the upper-level optimization task and an inner algorithm handles the lower-level optimization task, thereby making the overall bi-level optimization computationally very intensive. This work presents the Multi-Objective Genetic Algorithm for Structured Inputs (MOGASI). This algorithm combines modules and operators of standard GAs with specialized routines aimed at achieving enhanced performance on specific types of problems. MOGASI classifies variables and constraints by applying specialized data handling strategies to sub-problems with different data structures. The algorithm has a classic pre-processing phase which restricts the feasible domain and reduces problem complexity by eliminating equality constraints. Unlike conventional GAs which work with a single population and evolve it towards optimal solutions, MOGASI works with two separate, concurrently existing but communicating populations. MOGASI also has a mechanism for replacing, whenever possible, unfeasible individuals. This algorithm has been extensively tested against a set of benchmarks well-known in literature and compared to other state-of-the-art GAs. Furthermore, it has been efficiently adapted to handle bi-level problems through a nested approach. In short, the GA is used at the upper-level and a custom solver at lower-level. This

application and its effectiveness are shown in two real-world scenarios, both of which fall into the category of pricing problems.

The first application scenario is the Network Pricing Problem (NPP). It consists in a company needing to determine the price of its products/services so as to maximize its revenue or profit. At the same time, it must consider the customers' reactions to these prices as they may refuse to buy a product or service if the price is too high. This class of problems was first studied in the 1990s and is NP-hard, although there are polynomial algorithms applicable to particular cases. The NPP application of the MOGASI algorithm concerns pricing problems in a road network, where typically an authority owns a subset of toll arcs and imposes tolls on them in an attempt to maximize revenues, whereas users traveling on the network seek to minimize costs. The NPP case described in the dedicated chapter involves connecting toll arcs to create a single path, similar to those used on highways. If users who leave the highway do not re-enter it, tolls can be simply imposed on paths uniquely determined by their entry and exit points. A custom set of instances was generated to analyze the possible differences between a solution with an exact solver and with an evolutionary bi-level approach. The main focus was on the behavior of the two approaches in relation to the increasing problem size.

The second application is the Peak-load pricing (PLP) problem, a two-tariff charging scheme commonly used in public transport and utilities. It is tested on the European Air Traffic Management (ATM) system as a means for redistributing air traffic and thus reducing airspace congestion in Europe. In particular, this work presents a centralized approach to PLP (cPLP) which presumes the existence of a Central Planner (CP) setting en-route charges for the entire network, as opposed in current system where charges are set in a decentralized way. CPLP consists in two phases. In the first phase, congested airspace sectors and their peak and off-peak hours are identified whereas in the second phase the CP assesses and defines en-route charges based on those hours in order to reduce the overall schedule displacement on the network. These charges should guarantee Air Navigation Service Providers (ANSPs) to recover their operational costs while inducing the Airspace Users (AUs) to route their aircrafts in a way that the network is able to sustain. The cPLP approach and the analysis presented in this work were developed in the framework of the SESAR WP-E project SATURN (Strategic Allocation of Traffic Using Redistribution in the Network), which investigated the possibility of mitigating the existing demand-capacity imbalances at the strategic level of flight planning, that is, months in advance of the day of operations, through the modulation of en-route charges. The solution of cPLP was tackled with a nested approach using MOGASI as the outer algorithm and a custom solver that was

specifically developed for the problem as the inner algorithm. The target was to show that significant improvements in traffic distribution in terms of both schedule displacement and air space sector load can be achieved through this simple en-route charges modulation scheme in a multi-objective implementation of cPLP. This approach solved much larger data instances than ever before, up to one day of traffic on the whole European network (ca. 30,000 flights over forty states).

To conclude, the innovative contributions stemming from the applications of the MO-GASI algorithm in a bi-level multi-objective optimization framework are summarized, jointly with the remaining open issues and the possibilities for future work.

CHAPTER 2

GENETIC ALGORITHMS

Genetic algorithms (GAs) are an important type of meta-heuristics that are used to address hard optimization problems (De Jong and Spears, 1989). They are inspired by the Darwin's evolutionist theory on the origin of species and natural selection. GAs belong to the larger class of Evolutionary Algorithms (EAs), the idea of which was introduced by Rechenberg (1965) and further developed in the following decades. The father of the GAs, due to his considerable contribution in the field, is deemed to be Holland (1975) of the University of Michigan.

The functioning of GAs can be explained by considering the analogy with nature. When the individuals reproduce, their genomes, i.e. their chromosomes, are combined, so their offspring inherits some of the characteristics of each of the parents. Only the specimens with the best combination of genes are well adapted to a given environment and have the possibility to survive and create offspring. Such individuals are deemed *dominating*, whereas all others with genes ill-adapted to the environment are called *dominated* and are expected to eventually die out. Considering that mostly "good" genes are passed on, the overall goodness of the entire group of individuals, called *population*, will gradually increase generation after generation and over time the population will steadily improve and eventually evolve.
This theory is based on the following concepts:

- Individuals in a population carry genes with a number of different traits;
- Genes are inherited from parents to offspring in different combinations;

- Individuals with the most advantageous gene combinations have the best chances for survival and reproduction in a given environment (survival of the fittest).

The chromosome of an individual is the result of the recombination and mutation of parents' genes. GAs are stochastic algorithms that use random processes in their search and are able to take large, potentially huge search spaces, looking for optimal solutions. Random draws are often used in the initial population generation, in the selection of parents' solutions, in the mating of selected solutions, and in the mutation of solutions. GAs are based on the idea that the recombination of solutions (individuals) in a population can potentially find new and better solutions called offspring. Translated into mathematics, each solution found by a GA is made of a combination or a mutation of elements of all variables that were used to generate it. These value combinations can be considered as the chromosome of the individual, whereas the values in each element constitute the individual genes.

Based on the problem characteristics (or environment in Natural Sciences), the algorithm computes a quantitative measure of goodness of a solution, referred to as *fitness* (adequacy to the environment). A selection of solutions with the highest fitness scores are preferred for the creation of subsequent solutions that have a higher probability to inherit good characteristics entailing higher fitness values. In the analogy with Natural Science the children generated by parents well adapted to the environment have a high probability to be well adapted too. Other, less fit individuals are discarded: the algorithm promotes the most promising portions of the problem space that should be searched intensively, keeping solutions residing there. In fact, it uses the *survival-of-the-fittest* principle to determine the individuals that will survive in order to become part of the new generation.

The steps of a GA can be summarized as shown in Figure 2.1. For a complete description of GAs' structure and properties see Chapter 3 of Eiben and Smith (2003).

In GAs a fundamental concept is the *diversity* of solutions, which allows the algorithm to explore vaster areas of the search space. There are a number of factors explored and studied in literature that can influence diversity, starting from the different operators to preservation and propagation mechanisms. The initial population generation, parent selection, recombination operator, mutation operator, survivor selection, population size and repair operator can all affect a search profile and therefore diversity, as can the parameters used to implement each of these strategies. However, understanding and predicting the exact impact of a GA strategy on the search space and diversity is difficult because it implies that the true nature of the problem is known. If this were the case the most appropriate methods could be chosen to search it, but this knowledge would also make the use of optimization

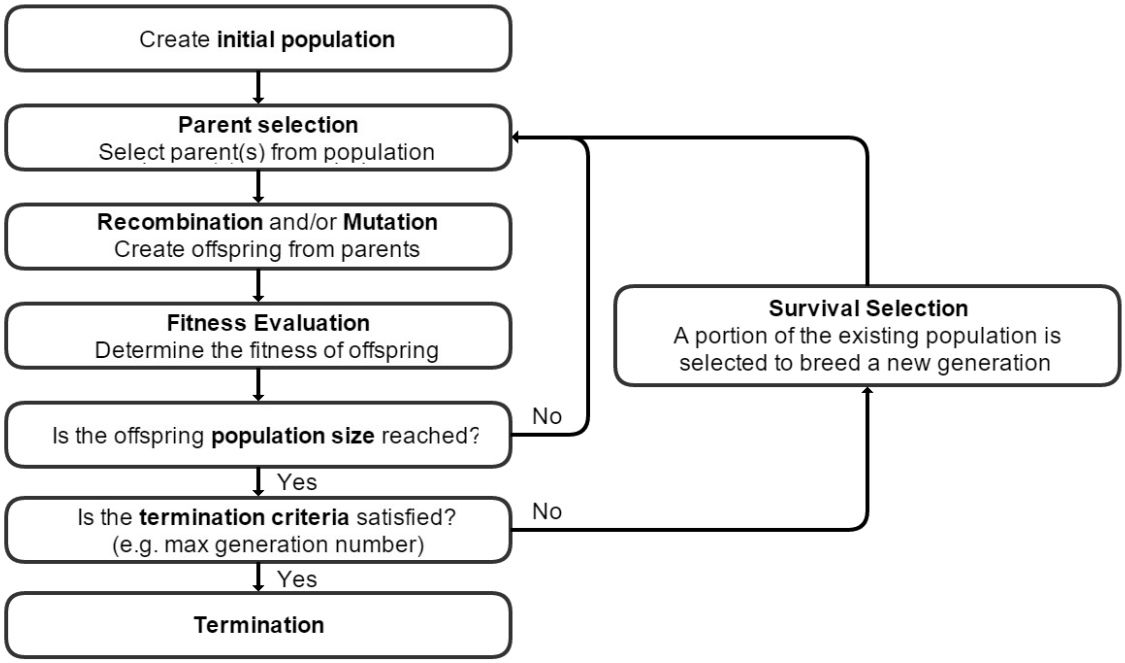


Figure 2.1: Generic Genetic Algorithm Workflow

superfluous. A clear example are engineering design optimization problems tackled in the industrial sector with the so-called *Black Box Optimization* (Muñoz et al., 2015), where the relationships between decision variables, objectives and constraints of the optimization problem are overly complex for a mathematical or analytic formulation. In this context, specialized engineering simulation solvers (e.g. thermodynamic simulations, finite element method simulations, and so forth) are used. Due to their intricate complexity it is almost impossible to make unique assumptions to fit a single optimization strategy. For this reason the GAs are particularly appreciated because of their stochastic nature (Michalewicz and Fogel, 2000; Turco and Kavka, 2011). In fact, they introduce a certain degree of randomness in the search process making the search less sensitive to modeling errors and escaping any local optima to converge to the global optimum without making additional assumptions about the underlying fitness landscape. It is therefore important to analyze how different GAs and methods implemented therein handle and maintain diversity and thus shape the search space.

2.1 Fundamental Genetic Algorithm Concepts

Mathematical and real-world problems have to be translated in a language that GAs are able to understand in order to solve them. Early GA research encoded individuals into a set of *binary strings* representing each decision variable (Eiben and Smith, 2003), i.e. using sets of 1s and 0s. To perform an optimization of two real valued decision variables, a binary string would be generated for each of the two decision variables. Since early works on GAs used binary encoding, problem details were frequently not embedded in the GA. Even though modern GAs use a wide variety of encodings, many of them are not binary strings and the chosen operators are typically specific to the encoding of the solution and the problem details (Eiben and Smith, 2003). For the sake of clarity and simplicity the explanation of the fundamental GA concepts in the following paragraphs is based on the binary representation. The encoded representation of an individual is its *chromosome*. The genes of an individual are the decision variables and their components, i.e. the individual 1s and 0s, are the *alleles*. For example, assuming a model with two decision variables, x and y, and a solution encoded as 10010010, the first four alleles (or bits) 1001 form the individual's gene representing a value of the variable x, whereas the last four alleles 0010 form the second gene of the individual's representing a values of the variable y. 10010010 is thus the chromosome of this individual.

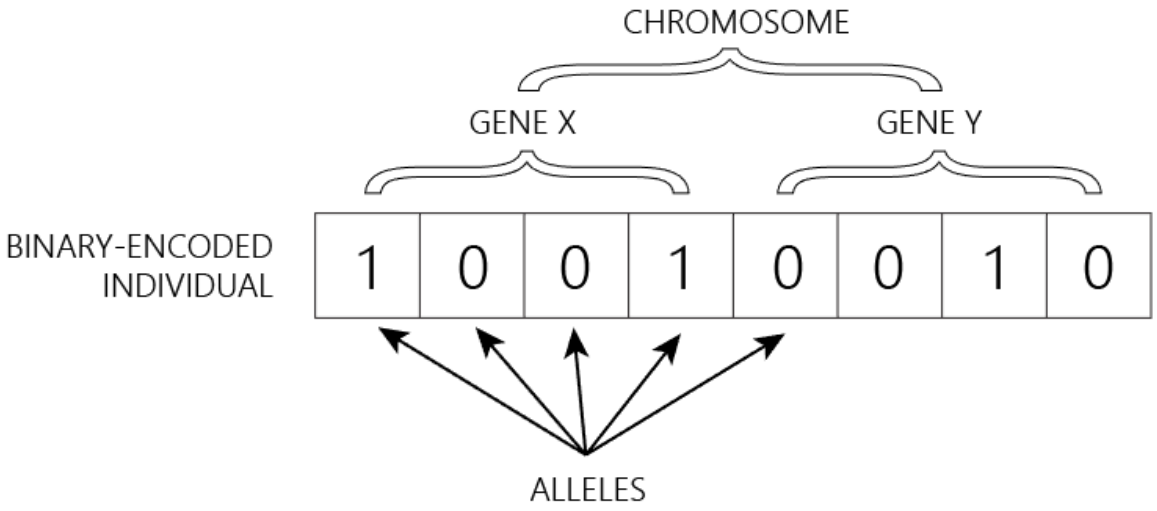


Figure 2.2: Representation of a Chromosome

The individual's encoding, i.e. how it is represented so it could be understood and manipulated by a computing system, is its *genotype*. On the other hand, the same individual in the real world solution space is represented differently, that is its *phenotype*. Depending on encoding, a given genotype may or may not represent all solutions that fully cover the

phenotype space, which is a well known issue discussed in literature (Michalewicz and Fogel, 2000). e.g. in the above example the phenotype space would contain all possible values of the variables x and y, whereas the genotype is the actual binary encoding of those values.

In a single-objective optimization framework each solution has a value for each objective, which determines its performance, i.e. *fitness*, with respect to that objective. The fitness values of all possible solutions make up the *fitness space*. An objective value of an individual is taken into account for the computation of its fitness, but it does not necessarily correspond to the *fitness value* which determines the goodness of a solution. In fact, objective values can be artificially modified for different purposes, such as constraint handling and promotion of diversity.

The *fitness space* should not be confused with the *problem landscape*, which is rather a mapping of the decision variables to the fitness space. The problem landscape term derives from its resemblance to a natural landscape with peaks and valleys, broad flat areas, areas of gradual increase or decrease and plateaus. Even though the majority of problems is multi-dimensional and cannot be so easily visually analyzed, the problem landscape concept is helpful for understanding other important GA concepts such as *neighborhoods*, *exploration* and *exploitation*. The definition of *neighborhoods* in particular heavily shapes the definition of diversity of GA, but it is problem-dependent. Neighborhoods can be defined in many different ways (Michalewicz and Fogel, 2000); for instance, as a set of distances in the search landscape (phenotypic or genotypic) or as a set of distances in terms of objective values. Friedrich et al. (2007) compared these two metrics and concluded that both have impacts on runtime performance. Most diversity literature focuses on diversity measured in the phenotypic search space, which is likely due to natural definitions for distance in that space and our intuitive perception of the search space as a landscape.

2.1.1 Exploration and Exploitation Balance

Exploration/Exploitation Balance (EEB) is the degree to which the GA searches in areas far from current solutions or close to current solutions (Maturana and Saubion, 2008). In particular, exploration can be viewed as a global algorithmic search able to cover most of the search space, no matter how roughly, even far from current solutions, for the purpose of discovery of new individuals. It can thus help maintain GA diversity by including very different solutions in the search. Exploitation can be viewed as a local search around a given solution or set of solutions in limited portions of the search space. Exploitation generally quickens convergence as it produces solutions that are rather alike existing solutions, pressuring the population towards homogeneity by quickly replacing and driving out worse solutions. In

most GAs exploration and exploitation are performed concurrently so it is important to strike a balance in the search profile to match the specific problem characteristics and handle a possibly limited amount of available time and computational resources. For instance, too much exploration can easily turn into a completely random search and the algorithm may (or may not) stumble upon the best individuals by pure chance. On the other hand, too much exploitation will focus the search in areas that may not be the actual best areas of the problem landscape and leave others unexplored being too initialization point dependent. The crossover genetic operator, explained in section 2.1.4.1, can create solutions that are either close to parents (exploitation) or solutions that are farther from parents (exploration). Varying the degree to which the GA produces similar offspring can help manage diversity in the search process.

The concepts of exploration and exploitation are also closely connected to the definition of neighborhood in a problem landscape. When a new search point is within an already identified neighborhood, the algorithm is exploiting (solutions). If the new search point is outside the known neighborhoods, the algorithm is exploring. Since most GAs do not keep a history of visited places (Michalewicz and Fogel, 2000), exploration is often conveyed as a direct function of the current population rather than a function of all the visited places.

Convergence and diversity are directly related. The faster a GA converges, the faster diversity leaves the population. EEB is one of diversity management strategies that can be used to slow down convergence and thus reduce the impact of performance-degrading phenomena discussed in the dedicated subsection 2.2.

Focusing entirely on aspects of exploration and exploitation can be misleading (Eiben and Schippers, 1998). This is due to two factors: the EEB focuses on only a crisp definition of neighborhoods. Many definitions of exploitation and exploration are possible due to the ambiguous definition of neighborhoods. Focusing on the optimization of a single EEB can be misleading since many definitions of a neighborhood imply that there are more than one possible EEB to a problem. In general a good practice should be to dynamically adjust the EEB ratio during the run so that the first optimization steps have a higher exploration degree and that the final optimization steps have a higher exploitation degree. This can help the algorithm to collect sufficient information on the search space before focusing only on the refinement of the most promising areas.

2.1.2 The Schema Theorem

The Schema Theorem, developed by John Holland and first published in his book Adaptation in Natural and Artificial Systems in 1975 (Holland, 1975), is a milestone in GA theory as it

explains why GAs work so well in practice.

According to the Schema Theorem, GAs are a method that samples large portions of the problem space called *hyperplanes*. Hyperplanes are nothing more than subsets of the entire set of solutions. Each *schema* describes one such hyperplane containing all possible individuals for which all the genes match that schema. In other words it is a kind of a template that enables the exploration of similarities among chromosomes. In fact, some portions of a schema are defined, while others are not, which ultimately determines the differences between the members of the set. For example, using the binary representation, a schema could be the following: *11*0***, where 1s and 0s are fixed allele values, meaning that all individuals in this set contain exactly those values in the given loci, i.e. "1" in the second and third allele and 0 in the fifth allele. "*" symbols, on the other hand, are *wild-cards*, meaning that at least one solution in the set has a different value than all the others in that allele, either 0 or 1. No information on whether more individuals have one value or the other is provided. The number of fixed-values alleles (non "*") determines the *order of the schema*. The above example is thus a 3-order schema. The distance between the first and the last fixed allele is called *defining length*. In the above example the distance is 3.

The Schema Theorem states that GAs perform well because short, low-order schemata with higher fitness increase exponentially in successive generations as a result of the application of the GA operators and the selection of the fittest individuals. It affirms that GAs sample hyperplanes (schema) in proportion to the representation of that schema in the population and the average fitness of the solutions sampled in that schema (Goldberg, 1989c).

More specifically, recombination and mutation genetic operators (discussed in Section 2.1.4) can be disruptive with regards to schemata. In general, the greater the defining length of a schema, the greater the probability that it is broken due to the crossover cut effect. Similarly, higher-order schemata have a greater probability of being broken by mutation than lower-order schemata. Finally, using a selection technique which chooses parents based on their fitness, fitter schemata will have a greater probability of finding their way unbroken from one generation to the next and also increasing their presence in the population.

Another fundamental concept closely related to the Schema theorem is the so called *implicit parallelism* introduced by Holland. It refers to the fact that the effective number of processed schemata is greater than the number of the processed structures, i.e. greater than the population size. More specifically, a population of size n with the chromosomes of length m can contain up to 2^m and $n \cdot 2^m$ schemata, but actually at least n^3 schemata are processed without any extra memory or processing requirements. This is accomplished by continuously exploiting the currently available incomplete information on those schemata

while trying to find more information on them and other, possibly fitter schemata.

The Schema Theorem, however, has several shortcomings. Firstly, it considers only the disruptive effect of the genetic operators - it does not take into account schemata being constructed in this way. This phenomenon is explained by the Exact Schema Theorem (Stephens and Waelbroeck, 1996). Secondly, it focuses on the number of surviving schemata and not on which schemata survive. Such considerations have been addressed with the use of Markov chains, as illustrated in Nix and Vose (1992). Finally, the statement of an exponential increase of fit schemata, meaning that they are always fitter than the population average, is misleading (Goldberg, 1989c) because the average population fitness increases with time and converges with the fitness of the best schemata.

The Schema Theorem is proven only for a single generational change and hold only under the assumption of infinite population sizes. The populations for practical GA implementations are always finite so any sampling error may lead to convergence of schemata in local optimum areas.

2.1.3 The Building Block Hypothesis

Matching the best partial solutions to construct better and better strings is the basis of the Building Block Hypothesis. It is closely related to the Schema Theorem because according to this hypothesis GA identifies and recombines short, low-order highly-fit schemata, called *building blocks*, into increasingly fit individuals (Goldberg, 1989a). The emphasis in the GA is placed on the recombination of individuals rather than on mutation, as is the case with Evolution Strategies (Bäck, 1996).

Building blocks should thus be processed with the minimum disruption caused by crossover and mutation. For this purpose, an advance knowledge of the configuration of potential building blocks is important for the appropriate operator design, in particular crossover. The Building Block Hypothesis implies that for the GAs to be effective, solutions should have some complementary components (synergies to be combined into better solutions). Most traditional GA approaches to single optimization problems use overall fitness in survival and parent selection, instead of fitness based on the solutions component parts.

GA schema fitness definition is of paramount importance in this context, but its meaning is often unclear. Schema fitness can be interpreted as the average fitness of all individuals with that schema with respect to the average fitness of all individuals in the search space Sivanandam and Deepa (2008) call this version the "static building block hypothesis". According to this interpretation, if all individuals but one in a schema have low fitness and the

one individual has very high fitness, the average fitness schema will be high, but nonetheless it is likely that this schema will disappear in a few generations. An opposite example is a schema containing many individuals with an above average fitness and few individuals with a very low fitness, resulting in a rather low average schema fitness. However, unlike the schema from the previous example, this schema will likely survive and produce good solutions. According to the abovementioned authors, a schema can be considered fit if the average fitness of its individuals in a number of population is higher than the average fitness of all individuals in all those populations. This interpretation is called the "relative building block hypothesis".

2.1.4 Standard Genetic Operators

Genetic operators are a core part of a GA because they guide the algorithm towards the solutions by promoting and preserving genetic diversity (*mutation*) and combining the existing solutions into new solutions (*recombination* or *crossover*). Mutation operates on a single chromosome so it is defined as the unary operator, whereas crossover operates on two chromosomes at a time and is thus defined as the binary operator.

2.1.4.1 Crossover

Crossover or recombination is the fundamental GA operator which takes more than one parent solution (chromosomes), combines their genes and produces one or more offspring solutions with the parents genetic material. Crossover can affect the algorithm convergence rate (i.e. the algorithm ability to find the optimal solution after as few evaluations as possible) because it can create individuals that are either very alike or very different from the parents. The dissimilitude of children to parents for different crossover operators is based partly on random choice, and partly on the crossover operator itself. The point of mating solutions is to transfer properties from differing parents in creating new offspring. According to the building block hypothesis, crossover attempts to create a child that is similar to the parents (probably located close to the parent genotype positions) but hopefully fitter than either of them. Selecting recombination operators that work in harmony with the overall GA strategy is important. Eiben and Smith (2003) provide more formal definitions for crossover and a good overview of a spectrum of crossover operators.

There are many recombination operators, most of which are problem specific. The following are the most frequently used crossover types in GAs:

The *single-point crossover* takes the first part of one parent and splices it with the later part of a second parent. The point in which the crossover operator cuts between the two parents is determined by a random draw. If the random draw happens to be near the beginning or end of a solution, the created children will highly resemble one parent or another. Figure 2.3 below demonstrates how single-point crossover can generate two children with a 3rd position cut on a 8-bit gene representation. The chance of creating offspring that are identical or nearly identical to the parents using one point crossover depends on the number of alleles that the parents differ by in total and how identical the parents are in the first and last portions of their genotypic representation.

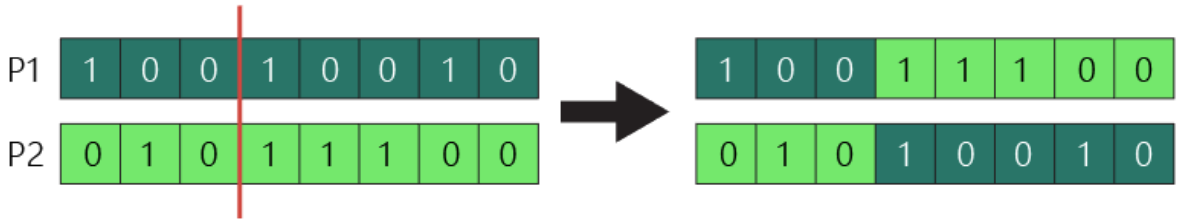


Figure 2.3: Example of an application of single-point Crossover

Goldberg (1989c) presents a generalized version of single point crossover called *multi-point crossover*. In this crossover the parent's chromosomes are cut in multiple random points and the alternating segments are then swapped between the parents to create offspring. The example on the figure below is a two-point crossover.

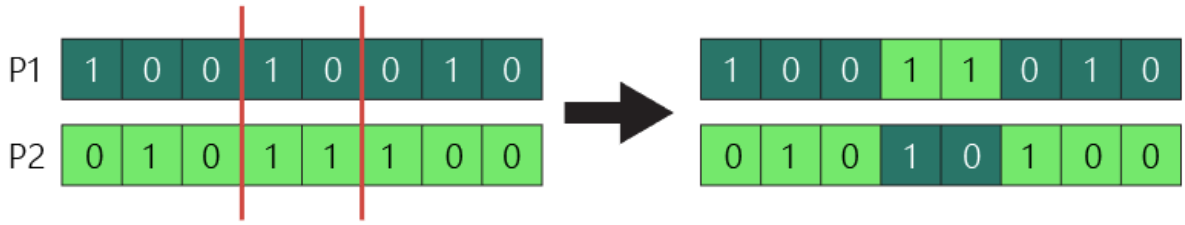


Figure 2.4: Application of multi-point Crossover with two cut points

In *uniform crossover* (UX) the chromosome are not divided into segments, but each gene is rather treated differently. UX picks which genes will be inherited from the different parents based on individual random draws for each non-unique allele. UX selects on average half of the genes from one parent and half from the other. The likelihood that parents will generate identical offspring (to the parents) is a function of the number of different alleles between the parents. For a genotype represented as a string of binary digits the likelihood of generating exact replicates of the parents can be approximated by the binomial distribution

where n is the number of bits differing between the parents, p is the probability of selecting a 0, and m is the exact number of successes to check. The binomial distribution is represented with the probability mass function as follows:

$$P(M = m) = \frac{n!}{m! \cdot (n-m)!} \cdot p^m \cdot (1-p)^{n-m} \tag{2.1}$$

For the typical UX operator, $p = 0.5$ which makes the distribution symmetric about $n/2$. For parents that produce exact copies of children, m must be 0 or n. For instance, to calculate the probability that parents that differ by five bits will generate exact replicates, n is 5, p is 0.5, and m is 0 and 5. For this example, the probability of creating an exact copy of either parent is 6.25 percent. As n increases, there is a lower chance of obtaining an exact copy of either parent via UX.

Another example is the *half uniform crossover* (HUX) operator proposed by Eshelman (1991), which is a variation on the UX operator. The HUX ensures that exactly half of the bits that are different between parents convey from one parent and the other half convey from the second parent. This operator ensures maximum diversity of the produced offspring. A child will have equal properties of both parents but can never be an exact copy of a parent, except when the parents differ by no more than one bit. Mauldin (1984) proposed a crossover operator that checks the offspring against the current population after normal crossover. If the offspring differs by less than k bits from any member in the population, bits are flipped at random until the offspring differs at least by k bits from each member. This method has the potential to introduce significant amounts of new genetic material into the gene pool. Over time k is gradually reduced to focus the efforts on solutions that are closer together as the algorithm progresses, this could remember an annealing procedure. This approach was the first attempt by GA researchers to explicitly maintain diversity in the current population.

2.1.4.2 Mutation

In simple terms, the mutation operation may be defined as a small random alteration in the chromosomes of an individual to get a new solution. It is used to maintain and introduce diversity in the GA population. For this reason, mutation is primarily referred to as an exploration operator. Most mutation operators are purely random and thus very high mutation rates can force the GA to become no better than random search. Pure random non-repeating search is often not the best search strategy on a constrained budget as random search does not use known problem information or structure. Normally, mutation rates are kept low so that the GA heuristic can work. Variable mutation rates often give better

results than static mutation rates. GA search often benefits from an adaptive or self-adaptive approach controlling the mutation rate (Thierens, 2002).

Different kinds of mutation operators exist. For instance, in the bit flip mutation one or more random bits are selected and flipped. This is used for binary-encoded GAs. Random resetting is the extension of the bit flip mutation for integer representation: a random value from a set of allowed values is assigned to a randomly chosen gene. In swap mutation, random positions in a chromosome are selected and interchanged. Scramble mutation chooses a subset of genes from the entire chromosome and shuffles their values randomly. In inversion mutation a chosen subset of genes is not shuffled but instead the entire string is merely inverted. These last three types are commonly used in GAs to solve problems with order-based encoded solutions.

Unlike standard mutation, cataclysmic mutation or mass extinction (Eshelman, 1991) is a widespread changing of many alleles in multiple solutions or the killing-off of many solutions in a single phase of an algorithm. This type of mutation can generally be thought of as being a reset or restart of the search process.

2.1.5 Selection Pressure

Selection pressure can be defined as the level of selectivity of the GA, that is its tendency to select only the best individuals in the current generation, either as parents or for survival. On one hand, an excessive selection pressure can have a negative impact on the genetic diversity and thus lead the algorithm towards a local rather than the global optimum. On the other hand, insufficient selection pressure will slow down convergence (Goldberg and Deb, 1991). For these reasons most GAs avoid taking only and exclusively the fittest of the population for reproduction and discarding all others. Such a strategy maintains a higher genetic diversity in a population because even the less fit individuals have a chance to reproduce. This enables the algorithm to better explore the search space and possibly find excellent solutions in unexpected regions, instead of concentrating on a narrow area. The actual amount of selection pressure depends on the implemented method.

Parent selection determines which solutions will be subject to genetic operators to pass on their genetic material to the offspring. It has therefore a direct impact on crossover diversity and convergence rate, for example mating parents that are very similar generally results in more exploitation while mating parents that are very different results in more exploration. The most common parent selection operators are tournament selection, roulette wheel selection, rank-based selection and random selection, each entailing a different selection pressure. They are explained below:

- The *tournament selection* is generally intended as a competition between two or more random individuals: the fittest among them wins and becomes a parent. It usually involves generating a random value for each individual taking part in the competition and comparing it to a pre-determined selection probability. If the random number is less than or equal to the selection probability, the fitter solution is chosen; otherwise, the weaker solution is chosen. This probability is usually a parameter and it can be therefore used to adjust the selection pressure and thus influences the population diversity.

- The *roulette wheel selection* (Holland, 1992) gives every individual a chance for being selected, even though this chance is greater for fitter candidates. Each individual is allocated a section of an imaginary roulette wheel of a different size proportional to their fitness, i.e. the fittest candidate has the largest slice, whether the weakest candidate has the smallest slice. After the wheel is spun the individual associated with the winning section is selected. The wheel spinning occurs as many times as is required to select the sufficient number of parents to produce the next generation. It may happen that particularly fit individuals are selected more than once as parents. For this kind of selection if fitness variance is low, it is more likely that parents will be selected from diverse parts of the population. Conversely, more variance in solution fitness indicates that there is a higher probability that the selected parents are fitter. Thus, in populations that have high fitness variance, roulette wheel selection is more likely to breed only the top individuals. This causes faster convergence than a roulette wheel selection on a population with low variance.

- In *rank-based selection* the individuals are sorted according to their fitness and assigned a selection probability proportional to their ranking regardless of the actual fitness score. This avoids premature convergence and stagnation in the same search region(s) because it is not important if a solution is 100% or 1% fitter than the next: what matters is their ranking against other individuals. As the fitness differences among solutions decrease in the course of the search, the selection pressure increases.

- In *random selection* individuals are selected randomly from a population. This operator is beneficial for diversity, but not for convergence, and if used alone it may easily result in the loss of good candidates if the offspring is weaker than the parents. For this reason it is usually coupled with the elitism approach. Elitism consists in copying a small portion of the fittest individuals unchanged to the next generation increasing the probabilities that their traits are passed on as they are eligible for selection as

parents just like all other individuals in a generation, but may easily appear in several generations.

Another main type of selection is the so-called *survivor selection*. It determines which individuals survive and become part of the next generation and which do not. For this reason it is sometimes referred to as population resizing method. It is crucial that it ensures that the fitter individuals are not lost, but at the same time without keeping those individuals only since this may lead to loss of diversity and premature convergence. Many GAs use elitism for this purpose. The simplest survivor selection strategy is randomly choosing the surviving individuals. This approach, however, can lead to convergence issues. In *age-based selection* (Eiben and Smith, 2015) the notion of fitness is not used. Instead, each individual exists in the population for the same number of iterations. In a generational model the number of offspring is the same as the number of parents so each individual exists for just one cycle and the parents are simply discarded. This does not preclude that genetic materials of some individual might persist over time in the population, but for this to happen the configuration must be replicated by the crossover or the mutation stages, as it was explained in subsection 2.1.4. An increase of the mean fitness over time relies on having sufficient parent selection pressure and using operators that are not too disruptive.

A wide number of fitness-based survivor selection strategies have been proposed in literature (Eiben and Smith, 2015). The idea is that the children replace a number of the least fit individuals in the population, usually based on a general ranking according to the Pareto domination criteria combined with some diversity management strategies (see Section 3.7). For example, in Whitley's GENITOR algorithm (Whitley, 1989) a number of worst parents in the population is selected for replacement. Although this can lead to very rapid improvements in the mean population fitness, it can also lead to premature convergence as the population tends to rapidly focus on the currently present fittest member. For this reason, it is commonly used in conjunction with large populations and/or "no duplicated solutions" strategy. The elitism approach to survivor selection is often coupled with the age-based and stochastic fitness-based strategies to prevent the loss of the fittest members of the current population. In this way, one or more currently fittest members in the population is kept instead of being replaced by less fit offspring individuals.

Based on the survivor selection approach and on the evaluation mechanism, a GA can be generational or steady-state. *Generational GAs* create individuals in batches: a new offspring set is created from the members of the current generation and the members of the next generation are selected among both parents and offspring. *Steady-state GAs* do not have generations: instead, the selection process occurs as the individuals are created and, if

selected, the new individual is inserted in the new population at any time. In general, steady-state approaches have a higher survivor selection pressure than generational approaches, but in many cases they are preferred owing to their ability to efficiently exploit computational resources.

2.2 Performance Analysis

According to the Schema Theorem, the highly fit schemata can be found over time in an exponentially increasing number of samples, resulting in narrower and narrower search areas (Mitchell et al., 1991). In other words, exploration predominates early on in a run of a GA, but over time the GA converges more and more rapidly to the fittest schema and exploitation becomes the dominant mechanism. The Schema Theorem is based on the assumption of infinite size populations, so in case of finite size populations, any, no matter how small, sampling error may be magnified and cause premature convergence (Goldberg, 1989c). It is a state of degeneration of the GA search where the population becomes dominated by one or more suboptimal solutions. According to literature (Mahfoud, 1995; Maturana and Saubion, 2008; Oppacher and Wineberg, 1999; Smith et al., 1993), premature convergence is one of the main issues that can degrade GA performance, beside other phenomena such as *epistasis*, *deception*, *genetic drift*, *duplication*, *hitchhiking* and *operator bias*. Even the metric used for the survival selection can affect the convergence behavior of the algorithm in relation with the shape of the problem landscape.

These negative forces are better explained in the following sections.

2.2.1 Epistasis and Deception

In genetics, a gene is said to be epistatic to another gene if it masks the phenotypic expression of the second one (Strickberger, 1968). Rawlins (1991) introduced the analogous notion in the GA theory by defining the minimal epistasis as the situation in which every gene is independent of every other gene and the maximal epistasis as the situation in which no subset of genes is independent from any other.

Deception is a special case of epistasis (Beasley et al., 1993), which occurs when the selection pressure leads the GA search away from the optimal solutions, i.e. low-order schemata do not lead to optimal points but instead lead away from them, or towards sub-optimal points (Goldberg, 1989a,b).

Epistasis and deception are not entirely distinct since deceptive functions by definition must have some component interactions. They are essentially independent but can mutually reinforce each other. Epistasis does not necessarily entail low order schemata to lead away from the optimal solution: a problem where all solutions but one have sub-optimal values can still be difficult for a GA even without deception. In this case it might be that the low order schemata simply do not indicate a direction of increasing fitness. Problems that have the highest levels of epistasis do not contain regularities in the search space and as a result heuristics are no better (and often worse) than non-repeating random search. Grefenstette (1992) concluded that deception is not the only condition that makes a problem hard for GAs. Grefenstette (1992) also demonstrates that deception can be viewed as a dynamic phenomenon using an example that showed how a GA can be deceived (for a while) and still reliably find the optimal solution.

Even though it is a not a perfect solution, diversity management represents an efficient way to manually fight deception because in spite of undesired alleles surviving for a long time, it slows down convergence giving the GA time (and space) to stumble upon the good solution. In other words, GA with high diversity are less prone to be fooled by the phenomenon of deception.

Several authors (Goldberg, 1989a,b; Liepins and Voset, 1990; Whitley, 1991) have studied the properties of a particular class of epistatic problems, known as deceptive problems. However, for solving practical problems it might be more important to be able to estimate the degree of epistasis to shape the most suitable strategy for tackling the epistasis itself. For instance, Davidor (1991) shows that problems with very little epistasis are generally simple to solve. On the other hand, highly epistatic problems are unlikely to be solved by any systematic method, including the GA. The encoding used for a GA (Radcliffe, 1992; Vose and Liepins, 1991), i.e. the solution representation, is also of critical importance. In fact, the appropriate choice of encoding may reduce epistasis in a problem, just like an inappropriate encoding may increase it as much as to make a problem unsolvable for the GA.

Several methods of problem sampling have been proposed to date to determine the degree of epistasis in a problem through problem landscape analysis. Davidor's measure of epistasis variance was the first attempt (Davidor, 1991) but the measure did not account for possible scaling in fitness and the number of negative and positive interactions (Reeves and Wright, 1995). Measures of normalized epistasis were proposed to address possible bias from scaling the magnitude of fitness values (Naudts et al., 1997; Vanhove and Verschoren, 1995). Reeves and Wright (1995) proposed a *Design of Experiments* (DOE) approach using contrasts. This approach breaks the interactions into different groups that simulate a number of effects

together. The DOE approach is attractive because both magnitude of change and direction of change can be accounted for in the analysis. However, like all the other methods to date, the DOE approach requires assumptions about the significance of different interactions in order to provide a meaningful insight since effects must be aliased together in an incomplete sampling of the problem space. The problem of measures of epistasis based on problem space sampling is that if it was known what makes the problem hard, it could be isolated to make the problem easy. However, most GAs work on extremely large problems where a large-scale sampling of the space is time-prohibitive. Determination of when a problem is well suited for a GA is likely better done according to past experience with similar problems and any knowledge that can be gleaned from input data rather than from objective space sampling.

2.2.2 Genetic Drift

Genetic drift is the process of accumulating stochastic errors in the population gene pool resulting in a premature convergence to a sub-optimal solution. It is caused by random increases in the number of alleles of the same type forming genes over time. Once it begins, the genetic drift will continue until the involved allele is either lost or remains the only allele present at a particular locus. Eiben and Smith (2003) describe a simple example to demonstrate drift: a GA with the population of individuals evenly split in half between two solutions with equal fitness. Without considering the effects of mutation and crossover, the example shows how the population must converge to either solution due to drift error through an argument based on probability.

Even when there is a significant difference in fitness between the best solution and the population, there may be a fair probability that a converged population causes the algorithm to melt down the hill to sub-optimal solutions. As the population size gets larger and fitness values more similar, the force of drift can be even more significant. This phenomenon can be partially remedied by using an elitist survival strategy. However, drift effects can also happen on sub-elite solutions such that drift is not prevented simply by using an elitist strategy.

The population can be thought of as having properties of inertia where a converged population tends to stay in a converged state. Populations having properties of inertia that implies early influence on the search can be important as it gets more difficult to affect bias later in the search.

Genetic drift can be countered with diversity management techniques. Diversity in a population slows genetic drift rates because drift is partially a function of population similarity. It takes more steps for a GA to drift to a single solution if there is greater diversity

since one solution must force out all of the other solutions from the population to cause complete convergence. When the population members are similar, the GA may need to replicate only a few solutions to completely converge. Genetic drift typically occurs in smaller populations where some alleles are present in fewer copies and are more likely to be lost, whereas it takes more steps to cause convergence in a larger population because of a large gene pool. In any case the selection pressure should be greater than the drift pressure to prevent convergence to an arbitrary solution. Gibbs et al. (2008) proposed a way to calculate the population size for real-based problems to ensure that the selection pressure is greater than genetic drift given a set of assumptions and operators.

However, this method is highly restricted to the problem representation (real values) and associated operators. Much discussion has been given to genetic drift although little research has been done to characterize drift rates when combined with selection pressure in a GA. It is hard to distinguish which convergence in a particular run or set of runs is due to drift and which convergence is due to the selection pressure, making online measurement of drift rates difficult.

2.2.3 Duplication

Most GAs do not check that solutions are unique, resulting in a certain level of duplication of solutions in a population. Population convergence is the extreme case of duplication where every individual in the current population is identical or at least they share most of their genetic material. Duplication occurs less prior to convergence. Duplication can be harmful because it increases drift pressure and repeated function evaluations occur for copies of the same individual consuming additional computational resources. However, ensuring solution uniqueness in a GA is also costly because it requires a pairwise comparison of all individuals. Since the problem landscape is normally much larger than the number of sampled points, most GAs do not check for complete diversity of all individuals within a run. The rate of duplication of individuals for a GA is a function of the diversity properties of the GA, the population size and the proportion of sampled points compared to the solution landscape size. Large solution landscapes do not necessarily mean that solutions will not be repeated because the negative effects of drift, inappropriate selection pressure or landscape shape should also be accounted for. Although using large population sizes should usually reduce drift and selection pressure, and thus slow down convergence, this may not be feasible in case of limited computation resources. In fact, in such cases a trade-off is achieved by reducing the number of generations, which in turn does not give the GA the possibility to properly converge to the optimum.

2.2.4 Hitchhiking

Hitchhiking is another phenomenon that may lead a GA astray during its search. It has been defined so by Forrest and Mitchell (1993) in their attempt to show how GAs use building blocks to generate better solutions, even though it has been also previously discussed (under the name "spurious correlation") by Schraudolph and Belew (1992) and Eshelman (1991), among others.

Forrest and Mitchell (1993) created a problem set called the Royal Road where GAs were hypothesized to outperform hill climbers on a simple structured problem. The results were counter-intuitive to their hypothesis and showed that the GA performed significantly worse than the hill climber on the Royal Road problem. The article provided evidence that high fitness solutions (not optimal) quickly dominated the results of the runs despite the fact that parent selection pressures were reduced. The reason behind this underperformance was the phenomenon they called hitchhiking. Once a higher-order schema with high fitness is discovered, this schema spreads quickly through the population with sub-optimal alleles hitchhiking along with the ones in the schema's defined positions. This slows down the discovery of schemata in other positions, especially those that are close to the highly fit schema's defined positions. In this way hitchhiking seriously limits the implicit parallelism of the GA by restricting the schemata sampled in certain allele positions.

Fitter solutions got a higher chance of becoming parents so a reduction in parent selection pressure led to less bias toward selecting parents with high fitness. This quick dominance of single solutions caused problems for the GA in constructing the overall optimal solution as diversity was driven out of the population. The single solution that dominated the population carried with it a series of suboptimal bits, but because it was able to take over the population, the suboptimal bits associated with that solution also became dominating even though they did not contribute positively to the fitness of the solution.

Formally, hitchhiking is the process of poor alleles being represented in many individuals in a population because they are associated with a highly fit solution. No amount of hitchhiking is useful, but some amount is unavoidable as, by definition, sub-optimal solutions must always contain a component part that is sub-optimal. Hitchhiking can be reduced with diversity management methods because when similar solutions are not allowed to reside in the population, hitchhiking is reduced.

2.2.5 Operator Bias

When selected parents are combined to create offspring, ideally GAs could determine the 'best' traits to be passed on. However, this is rarely the actual case as there are a vast number of ways to combine two solutions. This has led to a large body of literature studying various recombination operators for different representations and problems. In GAs, recombination is also commonly described as crossover, as already introduced in subsection 2.1.4.1. Crossover operators are known to be biased toward selecting certain bits from different parents instead of others (Eshelman, 1991). On one hand, positional bias refers to how bits that are close together in the genotype of a given solution are likely to stay close together (Eiben and Smith, 2003). In single-point crossover the chromosomes of each parent are cut in one random location and the genes, i.e. variables, at each side of the cut are exchanged to form two new individuals. The first part of a child comes from one parent, whereas the second part comes from the other, and vice versa. The single-point crossover has therefore high positional bias. Moreover, the positions of the alleles have an effect on what solutions are created. Parents with the respective genotypes 0000 and 1111 in this case cannot produce a child with a genotype where 0s and 1s are interlaced because their bits come from single consecutive strings of each of the parents. Distributional bias, on the other hand, refers to the number of bits parents are expected to pass on to their children (Eshelman, 1991) and for example it is a characteristic of one of the classic genetic operators, the Uniform Crossover (UX). UX uses a mixing ratio which defines the number of genes passed by one parent and the other to their offspring and each bit of the parents'strings is evaluated for exchange with a fixed probability, which is typically 0.5. A larger number of bits that differ between the parents entails a lower probability that only a few bits from a parents are selected. For this reason in case of very different individuals UX searches farther away from them, i.e. performs more exploration than exploitation, whereas in such cases the single-point crossover manages to achieve more balance between exploration and exploitation. Operator bias can be counterbalanced with some diversity mechanisms, for example by randomly introducing genetic material with mutation, even though mutation brings about the risk of making the search completely random.

2.3 Diversity Management

One of the most efficient ways to counter GA performance-degrading phenomena such as epistasis and deception, genetic drift, duplication, hitchhiking and operator bias, are *diversity management strategies*. They are most effective because they tend to slow down the

convergence and thus reduce the influence of the above phenomena. Furthermore, these mechanisms ensure that the solution space is adequately searched, especially in the early stages. In other words, a GA with such mechanisms should also be able to tackle multimodal optimization problems. A diverse population can in fact simultaneously explore several hills in the fitness landscape and thus help global exploration discerning the local from global optima.

Diversity is usually managed indirectly through choices made when setting up the GA that is the search profile defining, for example, the ratio between exploration and exploitation or understanding the problem landscape. These choices include the initial population generation, parent and survivor selection strategies, recombination and mutation operator strategies, population size and repair strategies, as well as the parameters used to implement these strategies. Finally, the stochastic aspect of the algorithm should also be accounted for. Most evolutionary algorithms have no direct method of guaranteeing population diversity (Michalewicz and Fogel, 2000). The concepts of using direct methods to maintain diversity were first explored quantitatively by Mauldin (1984). Direct methods use in fact a measure of diversity between past and/or present solutions but due to a potentially large computational effort, in general they focus on analyzing the current population.

Figure 2.5 shows where diversity can be influenced in GAs level-wise. This hierarchy mirrors the problem-solving process of the GA: the inner layers are affected by the outer levels in such way that they cannot preserve any more diversity than the levels containing them. For instance, since recombination is contained within parent selection, recombination generates individuals that are no more diverse than the parents. Similarly, parent selection cannot select parents that are more diverse than those that survived the previous survivor selection. This hierarchy implies that decisions at the higher levels of the GA hierarchy have more widespread influences than the lower levels on diversity preservation.

The diversity hierarchy diagram, presented in Figure 2.5, shows three major areas where diversity can be influenced. The landscape defining area (dark green) is concerned with the elements that can shape the search landscape, more specifically its size, number and nature of dimensions and surface. These elements include the problem formulation, representation of the individuals, fitness function and constraint handling strategy. The middle level, selection methods (light green), encompasses mechanisms which determine the selection pressure, that is survivor and parent selection, whereas the last level, solution generation covers how solutions are created, usually by recombination and mutation, after parent solutions are selected. Specific choices of operators in the GA included in the remaining two levels (initial population, parent and survivor selection, recombination and mutation

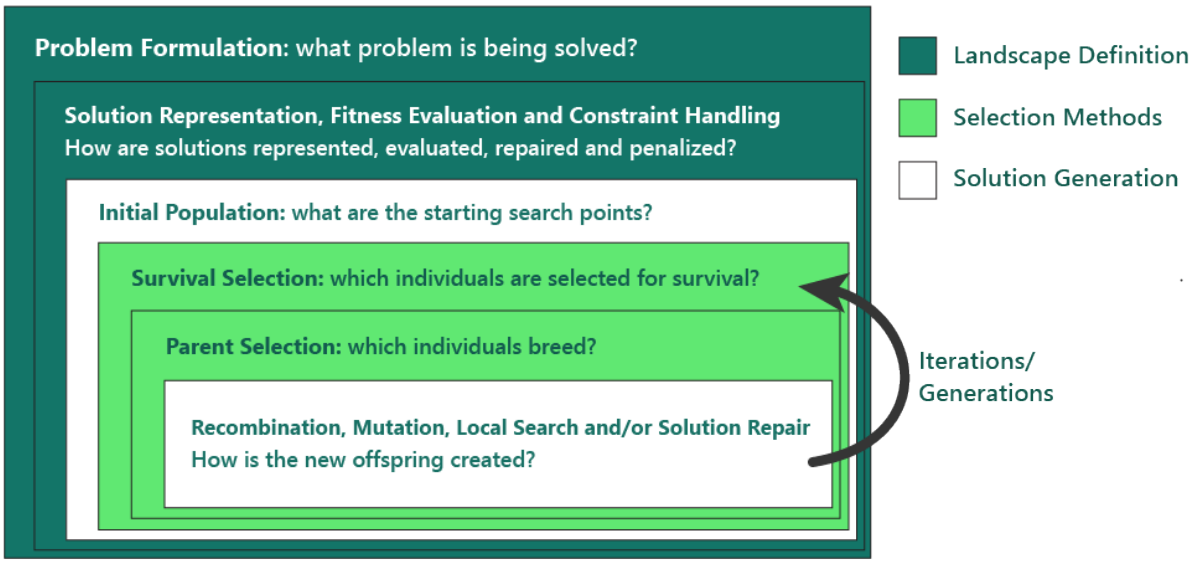


Figure 2.5: Possible problems in the adoption of a Genetic Algorithm

methods) can then be more or less effective depending on the landscape characteristics. Furthermore, any of these operators can contribute to a quick or slow convergence rate depending on how they are set up. For example, if survival selection directly eliminates duplicated individuals, as in Shimodaira (1997), it may slow down the convergence compared to selection by a method that does not remove duplicated individuals.

Among many existing diversity management methods, *crowding* and *niching* are most frequently used.

2.3.1 Niching

Niching was introduced by De Jong (1975) as a way for investigating many peaks in parallel and preventing the GA to get stuck in local optima (Sareni and Krähenbühl, 1998). In particular, it reduces the effects of the genetic drift. In nature, a niche can be defined as a subspace in the environment that can support different species, but its resources are finite so the population of the niche must share them. By analogy, in GA a niche is the location of each optimum of a multimodal function in the problem landscape, whereas the fitness represents the resources in that niche. Species can be defined as similar individuals in terms of similarity metrics.

The best known niching technique is *fitness* sharing, originally introduced by Holland (1975) and improved by Li et al. (2003). According to this technique each individual shares its fitness with its niche neighbors, so the search landscape is modified by reducing payoff in

densely populated regions. It lowers each population member's fitness by an amount nearly equal to the number of similar individuals in the population. In other words, solutions are penalized if there are too many individuals in the same region (Della Cioppa et al., 2004). The idea here is to discourage convergence to this single region of the fitness landscape by penalizing its "inhabitants". The more individuals seek to move in, the worse off they all are. In fitness sharing, individual fitness values become a function of the number of individuals in a niche. Thus the problem landscape can be thought of as transforming based upon the current population. The distance, i.e. similarity, between two individuals is based on either their phenotypic or genotypic similarity. Genotypic similarity is related to bit-string representation and is generally the Hamming distance. Phenotypic similarity is directly linked to the real parameters of the search space and it can be, for example, the Euclidean distance. Sharing based on phenotypic similarity may give slightly better results than sharing based on genotypic similarity. Goldberg and Deb (1991) use fitness sharing with the assumptions that niches were evenly spaced (by a σ distance) and that the number of the niches, q, is known a priori and q is much less than the population size n. The algorithm is $O(n \cdot 2)$ in complexity for all cases. Miller and Shaw (1996) use a greedy algorithm called *Dynamic Niche Sharing* to increase the efficiency of finding the different peaks during a GA run. The procedure has $O(n \cdot 2)$ complexity early and approaches $O(n \cdot q)$ as a run progresses. Della Cioppa et al. (2004) propose a method to calculate the optimal values for q and σ. The method does not require a priori knowledge about the problem landscape but since it requires a design of experiments process to determine values of n, q, and σ it can be restrictive since experiments may take some time. Shir et al. (2010) propose an algorithm called *Covariance Matrix Adaptation for Evolutionary Strategies* (CMA-ES). CMA-ES does not require a priori knowledge about the number of peaks q and the peak width σ. Instead, a parameter λ specifies the number of individuals per niche and the individual niche radii are calculated based on a learning algorithm and a learning rate parameter α. Unlike previous work, CMA-ES can find peaks with variable width. Peaks are punished for fitness if they contain more than λ individuals. Thus the implied problem landscape changes with λ and the maximum number of peaks that can be tracked.

Della Cioppa et al. (2007) propose a method called *Dynamic Fitness Sharing* that does not require the number of peaks q to be estimated beforehand and uses elitism on one individual from each niche. This method was able to solve difficult multimodal problem with smaller population sizes as the preservation of niches was explicit and thus large populations were not required to ensure the survival of each niche. However, ensuring survival of diverse solutions slows convergence as mating of diverse solutions commonly

causes more exploration than exploitation.

Smith et al. (1993) propose a fitness sharing method based on the biological analogy of antibody to antigen matching. This method uses a fitness bidding scheme to modify the fitness of solutions. Since partial matches can bid on solutions, generalist solutions can be preserved when the population is too small to maintain a set of all specialist solutions. Like other fitness sharing methods this one transforms the fitness landscape but there is a stochastic component to the landscape transformation. This stochastic component is a result of selecting antigens using a random process. Fitness is awarded based on the match of the individual population items to the selected antigen. The survival probability of an individual depends in general on its fitness and its difference from others in the neighborhood. The result is that as the GA converges to a single local optimum somewhere, the fitness of that optimum decreases because of the increased competition within the niche. Eventually, another region of the fitness landscape becomes more attractive and individuals migrate over there. In fact, fitness sharing tends to encourage the search in the unexplored areas. Sharing must be implemented with the less biased selection methods and use low recombination operators to promote stability of subpopulations. In fact, crossover between individuals of different niches often leads to poor individuals, called *lethals*. The formation of lethals can be successfully countered with mating restriction techniques, such as those discussed in Deb and Goldberg (1989), Yin and Germay (1993) and Miller and Shaw (1996).

Fitness sharing methods have two main disadvantages. They entail a significant computational cost as a result of the computation of niche counts of complexity for each generation. Moreover, they require advance knowledge on the distance of the optima to set a similarity threshold and can be somewhat unpredictable because they are based upon the contents of the current population and the problem landscape, both of which are unknown a priori. To contrast those disadvantages, Pétrowski (1995) proposed the *clearing method*, in which the best members of the sub-population take all resources in a niche instead of sharing them equally among all members. Coupled with elitism strategies, clearing can preserve the best elements of the niches through generations.

2.3.2 Crowding

Crowding is applied in the survivor selection step and consists in pairing each individual with a similar individual in the current population and deciding which of them survives. Based on how this replacement phase is carried out different types of crowding exist. The following are just a few examples. In standard crowding (De Jong, 1975), only a fraction of the global population specified by a percentage G (generation gap) reproduces and dies in

each generation. An offspring replaces the most similar individual (genotype-wise) from a randomly drawn subpopulation of size CF (crowding factor) (De Jong, 1975). This type of crowding has been shown to be limited in multimodal optimization (Deb and Goldberg, 1989; Mahfoud, 1995): even though it attempts to maintain diversity over time, this method can result in convergence to a single local optimum as the random choice of replacement candidates has an element of stochastic error that can accumulate over time and cause genetic drift.

To address the convergence issue, Mahfoud (1992) did several experiments with different crowding methods and noted that generally the offspring closely resembled one of the two parents. This observation led to an algorithm where only the parents were considered when eliminating the closest solution to the offspring (deterministic crowding). In fact, the offspring competes directly with their respective parents if they are in the same niche and replace one of them if they are fitter (Mahfoud, 1995). Similarity is computed using preferably phenotypic distance. The restricted tournament selection takes a random sample of CF individuals from the population after the recombination phase, as in the standard crowding. Each offspring competes with the closest sample element and the winners are inserted in the population. The keep-best crowding maintains the best parent and the best offspring to maintain good genetic material. The correlative family-based selection computes the distance between the individual with the best fitness in each family and other family members: the closest member is chosen to survive.

2.3.3 Diversity Management in Literature

The Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation algorithm (Eshelman, 1991), CHC in short, combines a highly disruptive crossover with elitism and incest control to promote high diversity. Incest control ensures that when parents are paired for mating the Hamming distance between them is computed and the parents create offspring only if half this distance exceeds a difference threshold. In fact, this method guarantees that the two offspring are always at the maximum Hamming distance from their two parents, which prevents premature convergence.

GAs can be classified as *panmictic* (i.e. use a single unified population) and *structured* (i.e. divide the population into separate component collections). As pointed out by Ursem (2002) classification, population structure can have a strong influence on diversity. There are a wide variety of names for structured populations including island, deme, multinational and colony models. These models share the common characteristic that they create separate subpopulations that pass information between each other in some fashion. Often these models

use a structure such as a hub-and-spoke, grid or circle to determine which individuals move between populations. Individuals move between sub-populations based on a migration operator.

There are classes of GAs that divide the population into subset populations sometimes called spatially structured evolutionary algorithms (EA) (Tomassini, 2005). By dividing the population these models affect both parent selection and survivor selection. Two common terms for spatially structured EAs are multiple-deme models and island models. These models maintain different demes (or islands) and occasionally the demes exchange individuals. Solutions are passed between the different islands after a number of generations have passed (an epoch's worth). Because these exchanges occur rather infrequently, spatially structured EAs are conducive to parallel computing efforts. Typically there is a structure that indicates which islands exchange individuals based upon a neighborhood definition for the islands (Cantú-Paz, 1998). Spatially structured models are sometimes also referred to as diffusion models because it takes a longer period of time for a good solution to propagate itself to all of the populations. Diffusion models can help combat premature convergence because local convergence rarely equates to global convergence in a distributed model. Since the exchange of individuals is relatively small compared to the number of individuals generated, a single population can converge and yet the overall GA can have significant diversity in the other populations. Different populations can use different operators so that if a given operator is causing quick convergence, it will not bias all islands in the model (Eiben and Smith, 2003).

Several classifications of diversity management methods and the effects they produce were proposed in literature. Mahfoud (1995) classified algorithms based on the negative phenomena they contrast, i.e. genetic drift, operator bias (Eshelman, 1991) and high selection pressure. However, Mahfoud (1995) does not distinguish between specific diversity methods and does not consider other negative forces such as deception (Goldberg, 1989b) and epistasis (Davidor, 1991). Another issue with this kind of classification is that a single algorithm may contain several diversity management methods and that a single diversity management method usually counters several of the above negative phenomena at the same time. Determining which among them is most prominently mitigated is subjective since there are no established and sufficiently reliable methods for measuring the extent of contribution to the GA performance degradation. Finally, Mahfoud (1995) describes diversity only in terms of the current population, even though diversity methods can use measures of inter-individual, inter-population and intra-population diversity.

Ursem (2002), on the other hand, recognized the possibility of having multiple diver-

sity management methods in an algorithm and attempted to classify them in three categories: structures that lower gene flow, operators controlling the selection procedure and re-introduction of genetic material. Structures that lower gene flow include the migration of individuals between populations and are usually implemented in spatially oriented evolutionary algorithms (EA), such as island or deme models, multinational EAs, and colony based methods (Tomassini, 2005). Operators that control the selection procedures include niching methods (e.g. deterministic crowding Mahfoud (1992) and fitness sharing (Deb and Goldberg, 1989). The re-introduction of genetic material implies the insertion of new genetic material in the gene pool, usually through mutation and random restarts. Ursem's classification unfortunately fails to consider some of the basic diversity management methods, such as crossover which cannot be classified using this method. It has been proved, however, that certain crossover types, such as half-uniform crossover (HUX) tend to create more diverse solutions than the uniform crossover (UX) and single-point crossover (Eshelman, 1991). Further the classification scheme does not identify the fact that solution representation can be an important part of diversity and as a result it cannot differentiate diversity achieved by changing the solution representation such as the highly redundant representation in messy GAs Goldberg (1989b). Lastly, the reintroduction of genetic material includes methods that can either reintroduce genetic material or attempt to preserve existing material.

Bhattacharya (2004) added two more categories to Ursem's classification scheme: Dynamic Parameter Encoding (DPE) (Schraudolph and Belew, 1992) and diversity controlled algorithms. DPE is a solution representation technique that adds precision to a solution as the algorithm converges. The DPE category attempts to address diversity in solution representation but the category is incomplete as there are many other ways to approach diversity in GAs using solution representations. The diversity controlled category is undefined because Bhattacharya does not specify what is meant by control. It can be assumed that control here is the measurement of diversity and adjustment of a population diversity based on that measure. The diversity controlled category is unfortunately not mutually exclusive of the original three categories. The attempt to classify some diversity methods as controlled highlights the fact that some diversity measures use an implicit level of control while others use a more direct level.

One of the most important distinctions is the one devised by Abbass and Deb (2003), that is the distinction between diversity promoting and diversity preserving methods. Diversity preservation is the act of maintaining diversity which already exists in the population, whereas diversity promotion is the act of adding new or rare variations to the population. Diversity promotion typically happens through the restart of an algorithm or through the

use of mutation. Diversity preserving methods, on the other hand, work only on the current genetic material in a population. Every operation in a GA that does not introduce random genetic material impacts diversity preservation.

Existing diversity classification schemes all share difficulties due to significant overlap within themselves which causes ambiguity and lack of general applicability. In fact, classifying GAs based on some diversity descriptors is difficult because diversity is affected by many elements such as operators, problem formulation, problem representation, fitness function and parameter settings. Furthermore, since there are many areas and several levels in which diversity is influenced, diversity control can be implemented in many different ways.

2.4 Optimization and Genetic Algorithms

Since 1940's when the term "mathematical programming" as a synonym for optimization was coined, optimization problems have greatly expanded in terms of size and complexity, especially with the enormous progress achieved in computer science and technology, and so have the optimization techniques for solving them in operations research, numerical analysis, game theory, mathematical economics, control theory and combinatorics. The existence of some sorts of optimization methods dates back to the days of Newton, Lagrange and Cauchy, but very little progress was made until the middle of the 20th century when high-speed digital computers made possible the implementation of optimization procedures and gave rise to intensive research of new methods. The major developments in numerical optimization methods have been made in 1960's. The development of the simplex method by Dantzig (1947) for linear programming problems and the introduction of the principle of optimality by Bellman (1957) for dynamic programming problems paved the way for development of the methods of constrained optimization. The work by Kuhn and Tucker (1951) on the necessary and sufficiency conditions for the optimal solution of programming problems laid the foundations for a great deal of later research in nonlinear programming.

Mathematically speaking, optimization can be intended as finding the best element considering of set of predefined criteria from a set of feasible solutions. It has three basic elements: a numerical quantity (or objective function) which has to minimized or maximized; a collection of variables, i.e. quantities whose values can be manipulated to optimize the objective; and a set of constraints, which are the restrictions on the values that variables can assume. The process of identifying objectives, constraints and variables of a problem is known as modeling or problem formulation, and that is an important preliminary step to optimization. Thus, the optimization starts from an existing model, often referred to as

baseline design, which should be improved as much as possible, taking into account the available time, resources and the designer's knowledge and ability to formulate the problem.

Once the problem is modeled, it can be solved using a numerical method, i.e. an algorithm. There is no universal optimization algorithm but rather a collection of algorithms, each of which is tailored to a particular class of optimization problems. Choosing the appropriate method is the responsibility of the designer and it may determine whether a problem is solved quickly or slowly, or whether it is solved at all. Mathematical expressions known as optimality conditions are used to measure if and to what extent has the algorithm been successful in finding the correct combination of variable values that represent the solution to the problem.

Optimization algorithms are iterative: they start from the baseline solution and generate a sequence of improved estimates until a termination criterion is satisfied. Important concepts related to algorithm functioning and performance are the following:

- *robustness*, which refers to the algorithm ability to find the global optima of the objective function even when starting far away from the final solution, without sticking prematurely to a sub-optimal solution.

- *accuracy*, which refers to the algorithm ability to get as close as possible to the objective function global optima without being overly sensitive to errors in data or arithmetic rounding errors that occur when the algorithm is implemented.

- *convergence rate*, which refers to the algorithm ability to find the optimal solution, i.e. reach convergence, after as little evaluations as possible and without requiring excessive computational resources.

Constraints are present in the large majority of real world applications. Based on the presence or absence of constraints, it is possible to distinguish between constrained and unconstrained problems. Unconstrained optimization finds the extreme of a function under the assumption that the parameters can take any possible value. Constrained optimization problems are those in which not all variable values, i.e. of independent or dependent variables, or both, are acceptable for the user. In some cases constraints can be safely disregarded if they do not affect the solution nor interfere with the algorithm functioning. Furthermore, some unconstrained problems arise as a reformulation of constrained problems in which constraints are replaced by penalties applied to the objective functions to discourage unfeasible solutions, i.e. those violating any of the constraints. Similarly, problems with too many objectives can be reformulated by replacing some of the objectives with constraints if

this makes their solution easier or even possible. Constraint handling and the high levels of complexity they bring to a problem are explained in Section 2.5.2.

In some problems the variable values are known only within a certain precision, they can be described by a probability distribution or are influenced by uncertainties or noise. Designer may obtain more useful solutions by incorporating additional knowledge about these uncertainties into the model rather than guessing or ignoring their existence. Stochastic optimization algorithms use these quantifications of uncertainties to produce solutions that optimize the expected performance of the model instead of the ideal performance.

2.4.1 Problem Formulation

As explained before, problem formulation, also known as problem modeling, is an important step of the optimization, as recognized by numerous research studies in the field. A good problem formulation captures important details while leaving out extraneous details. If the model is insufficiently accurate, it will not give useful insights into the practical problem; if it is too complex, it may be too difficult to solve. Good models have a structure that makes the solution process easier. Problem formulation involves a problem statement, data identification / collection, identification of design variables, identification of the criteria for measuring the success of the optimization and identification of constraints. The decisions made in the problem formulation phase usually imply the early selection of techniques appropriate for solving the problem. For instance, a problem formulated with multiple competing objectives should be solved with multi-objective methods.

The scope of the present work is GAs, but in general if the problem formulation is inadequate, any optimization algorithm may in fact fail. Radcliffe and Surry (1995) show how this can happen for the GAs in specific. The size of a problem instance, i.e. the number of decision variables and constraints, may also have an impact on GA performance. The general question is why a given search method performs well on some problems and not on others. Wolpert and Macready (1997) suggest that all search algorithms satisfying certain conditions regarding the way they work, perform on average in exactly the same way for all problems in the class. No algorithm offers a shortcut or, as the above authors stated, there is no free lunch.

Using the "no free lunch" metaphor, each restaurant (i.e. optimization strategy) has an identical menu in which each plate (i.e. problem) is associated with a price (i.e. algorithm performance on a problem) the only difference between the menus is that the prices are shuffled from one restaurant to the next. So the choice of the restaurant with the goal of paying as little as possible for one's lunch requires advance knowledge on what he/she wants

to eat and how much the chosen meal costs in all restaurants. Translated into mathematics, this means that the performance on a problem relies on using some prior knowledge to match the optimization strategy to the problem. In other words, if there are problems for which algorithm A is better than B, then there must be problems for which B is better than A (Wolpert and Macready, 1997). This can be one of the reasons why so many different GA customizations are present in literature. With this in mind the "no free lunch" theorem cannot be violated, but it can be circumvented under certain conditions. For example, one could design a GA maintaining general purpose performance characteristics and incorporate additional modules for handling specific problems. Naturally, the drawback is that those additional modules would lead to extremely low performance in solving problems far from its scope.

Problem formulation, in combination with the solution representation, constraint handling and fitness function definition, shapes the actual problem landscape that a GA searches, and thus determines the difficulty of the problem. Some landscapes are more difficult to search than others because, for example, they may have higher levels of deception which leads the GA astray more frequently.

2.4.1.1 Solution Representation

Solution representation is the way in which a solution is manifested in the algorithm, i.e. how it is encoded. Ultimately a solution representation maps to the decision variables of a problem and it affects the difficulty of the search landscape as well as GA diversity. Currently there is no proven theory on the influence of representations on the performance of genetic and evolutionary algorithms, but some researchers have made recommendations for the design of efficient representations. A detailed overview of such representations is given by Michalewicz and Fogel (2000).

One of the first such recommendations was made by Goldberg (1989c). He proposed the principle of minimal alphabets, which states that the alphabet of the encoding should be as small as possible while still allowing a natural representation of solutions, and of meaningful building blocks, which states that the schemata should be short, of low order and relatively unrelated to schemata over other fixed positions. The latter principle is directly motivated by the *Holland's Schema Theorem* (explained in section 2.1.2), whereas the former indicates to increase the potential number of schemata by reducing the cardinality of the alphabet. In fact, when using minimal alphabets the number of possible schemata is maximal. However, Goldberg also argues that in some cases a trade-off between the low cardinality of the alphabet and the natural expression of the problem is required because sometimes higher

alphabet cardinality could be helpful for the GA (Goldberg and Deb, 1991).

Palmer and Kershenbaum (1995) analyzed the properties of tree representations and the recommendations they gave can also be applied to other types of representations. They argue that an encoding should be able to represent all possible phenotypes; should be unbiased in the sense that all possible individuals are equally represented in the set of all possible genotypic individuals; should encode no unfeasible solutions; and should possess locality, that is small changes in the genotype should result in small changes in the phenotype. Lastly, the decoding of the phenotype from the genotype should be easy. Ronald (1997) presented a survey of encoding issues and recommendations on how to tackle them. He argues that encodings should be adjusted to genetic operators in such way to preserve the building blocks from parents to offspring, minimize epistasis and use an appropriate genotype-phenotype mapping if a simple phenotype mapping is not possible. He further states that the problem should be represented at the correct level of abstraction and that isomorphic forms should not be used if the phenotype of an individual is encoded with more than one genotype. An example of mapping of genotype space and phenotype space to the objective space can be observed in Figure 2.6, which was inspired from a similar figure presented by Weise et al. (2008).

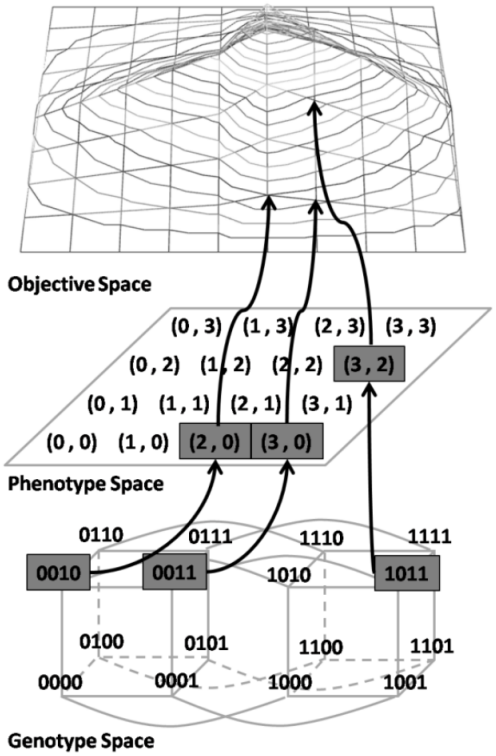


Figure 2.6: Example of Genotype, Phenotype and Objective Space Relationships

The theory of solution representation for GAs has often been ignored although the importance of choosing the proper representation has been stressed by many authors (Goldberg, 1989c; Liepins and Voset, 1990; Ronald, 1997; Thierens, 2002). These and other authors have underlined the existence of three basic elements of representation theory which affect the GA performance (Rothlauf, 2006), namely *redundancy*, *scaling of building blocks* and *modification of distances* between individuals when mapping the phenotypes on the corresponding genotypes.

A representation is defined as *redundant* if the number of genotypes is higher than the number of phenotypes, meaning that a phenotype is represented on average by more than one genotype. In general linear redundancy can be addressed as a matter of building block supply. The representation that gives more copies to high quality solutions in the initial population result in a higher GA performance, whereas encoding where high quality solutions are underrepresented make a problem more difficult to solve. Uniform redundancy has no influence in this sense.

The building block scaling refers to how different the contribution of the building blocks to the fitness of an individual is. If building blocks are uniformly scaled, the GA solves all of them implicitly in parallel. If they are not uniformly scaled, domino convergence occurs and the building blocks are solved sequentially starting with the most important building block (Thierens, 2002). As a result the convergence time increases and the search is more prone to the effects of the genetic drift. Less important alleles are less contrasted with sufficient selection pressure so some of them loose diversity and are randomly fixed.

Vose and Liepins (1991) showed that an inappropriate genotype-phenotype mapping can transform easy problems into very difficult problems for a GA by distorting the distances between individuals. High locality of representation, i.e. low phenotype-genotype distances, is a necessary condition for preventing this phenomenon (Rothlauf, 2006).

2.5 Encoding

Genotypic and phenotypic representations can have a large impact on the implied landscape. For instance, a problem encoded in a standard binary form can introduce Hamming cliffs into a landscape. Hamming cliffs occur where solutions that are close to one another in the phenotype space are far apart in the genotype space. Consider the phenotype numbers 7 and 8. Encoded in standard binary using four bits they are 0111 and 1000 respectively. While 7 and 8 are as close as possible in the phenotype space, they are as far apart as possible in the genotype space. Hamming cliff problems can be avoided by instead using gray coding which

ensures that numbers next to one another only differ by a single bit. Research studies on GAs frequently provide a mapping of gray code to standard binary, for instance in Eiben and Smith (2003). It has been argued that because of problems with encoding solutions, such as Hamming cliffs, it is easier to represent real-based problems with real-based phenotypic representations (Gibbs et al., 2008).

Binary encoding is the first and remains the most common form of encoding. Many of the first GA solutions were represented as strings of binary digits. In fact, binary coded GAs applied to problems with discrete variable are successful precisely because of the encoding and owing to the dedicated crossover operator which propagates building blocks from parent strings to offspring strings (Deb, 1999). The good building blocks should be coded tightly so that the crossover operator could combine them together. The problem of tight or loose coding of variable is known as the *linkage problem*, which is defined as the probability that two genes will be separated after recombination. In canonical GA adjacent genes have tighter linkages than non-adjacent genes, so the former are less likely to be disrupted by the crossover operator.

In the event that a problem is not naturally binary (i.e. has a continuous search space), the solution is usually encoded into a binary form and the search space is discretized. In other words, binary encoding of real variables divides a continuous real landscape into a discrete set of points. When binary-encoded GAs are used for real-valued problems a number of difficulties arise (Deb, 1999). One difficulty are the Hamming cliffs where a transition to a neighboring solution in the real space may require the alteration of many bits and thus hinder a gradual search. Another difficulty is the inability to achieve any arbitrary precision in the optimal solution because the string length is chosen a priori. The larger the string length, the greater the precision, but this also increases the population size requirement and thus the computational effort. Finally, in a continuous search space not all schemata are necessarily equally important. On the contrary, the meaningful schemata are those that represent the contiguous regions of the search space, so the crossover operator should be such to ensure their propagation. In fact, motivated by the success of binary-coded GAs in discrete problems, Deb et al. (2002) developed a real-coded crossover operator, called *simulated binary crossover* or SBX, similar to the single-point crossover in binary-coded GAs, but which works without using the coding of variables. Child strings are created from a probability distribution that depends on the location of the parent strings.

Schraudolph and Belew (1992) proposed that for encoded solutions requiring high levels of precision a solution could start with a low number of bits. When the solution with a given precision converges, bits are added to the solution representation with a zoom

operator. This action increases the precision of the search since the solution representation becomes larger. The algorithm for changing the solution representation through zooming was named *Dynamic Parameter Encoding* (DPE). It can be demonstrated that adding bits to the solution representation makes the problem landscape larger. However, since DPE waits for convergence before increasing the precision of the solution, it is, in effect, searching several smaller landscapes. While the name of DPE implies that it modifies parameters, it actually is working on the solution encoding, not the GA variables. DPE is theorized to reduce levels of genetic hitchhiking because a low number of bits is able to hitchhike with an early good solution.

In the past several decades solution representations in GAs have evolved to include more natural representations such as real encoded values or permutation strings. Three categories can be used to describe solution representations: *encoded versus natural*, interpreted versus direct, and static versus dynamic. The encoded versus natural and direct versus indirect classifications indicate how the variables are manifested into the solution representation. Some representations work on a set of variables that imply a solution through a heuristic rule which maps the genotype to specific solutions (interpreted). Other representations are more direct: the variables are represented in the solution itself. Two examples of interpreted representations include Aickelin (2002) and Carlson and Hougen (2010). For instance, in indirect set covering (Aickelin, 2002), the GA operates by evaluating row fitness rather than selecting individual rows. A heuristic is applied to determine which rows actually cover the columns. The heuristic that does the translation from the encoded to the decoded solution should be evaluated as an operator since such heuristics can have different approaches to diversity.

Both direct and interpreted representations can also be encoded or in a natural form. Variables that have their natural representation are in their corresponding natural type. For instance if a problem has a real variable x, a natural representation of x could be a floating point type or double floating point type. If x is represented in a non-natural form, it is encoded. A solution that has any encoded variables is defined as encoded solution. In representations that are both natural and direct, the phenotype space is identical to the genotype space.

Solution representations that can change over the course of the GA are dynamic representations. DPE (Schraudolph and Belew, 1992) and messy GAs (Goldberg, 1989c) are two examples that modify solution representation over the course of the run. In DPE additional bits of precision are added to variables as the algorithm converges. In messy GAs variables can be underspecified (some bits of a solution are absent) or redundantly represented

(appearing more than once) and thus the representation string may get larger or smaller, which in turn can be used to maintain diversity in the search process. While dynamic representations can be elegant, most GAs use a static representation in which the genotype components are fixed.

2.5.1 Fitness Function

The fitness function of a GA is a quantitative measure of goodness of a solution. In a single-objective optimization framework it can be considered as how close an individual is to the value of the optimization objective. In multi-objective optimization problems, however, the objective value of an individual is taken into account for the computation of its fitness, but it does not necessarily correspond to the *fitness value* which determines the goodness of a solution. In fact, objective values can be artificially modified for different purposes, such as constraint handling and promotion of diversity. Therefore, a number of factors should be additionally taken in account to explain its complexity and importance for the quality and the convergence speed of the algorithm.

Calculation of the fitness value is done repeatedly in a GA and therefore it should be sufficiently fast. A slow computation of the fitness value can adversely affect a GA and make it exceptionally slow. Fitness evaluation can be either natural or modified. A natural fitness function is one that has a one to one correspondence with the original problem formulation's objective function. Modified fitness functions, such as functions that incorporate additional penalties or rewards to certain solutions, can be classified as either *relative* or *absolute*. Relative fitness functions have fitness that may depend on other members in the population (such as fitness sharing in Deb and Goldberg (1989)) while absolute fitness functions only depend on the individual itself.

Fitness functions can take many forms and can be used directly or indirectly in the maintenance of diversity as they shape the problem landscape. Fitness sharing is an example of how the manipulation of fitness achieves this effect. In addition to fitness sharing, other fitness function strategies have been used to manage diversity. For instance, Lee et al. (2003) proposed two algorithms to maintain diversity in the population without using fitness sharing and by adding instead a bonus to the fitness of solutions that have rare alleles.

Fitness function strategies for managing diversity can modify the perceived or implied problem landscape based on the current population. For such strategies the implied fitness landscape can be viewed as dynamic. How a fitness function can be used to modify the implied fitness landscape is easily demonstrated by an example. Examine the proposed function (F_2) and sharing method described by Deb and Goldberg (1989). The F_2 function is

valid for the real variable x in the range of 0 to 1 and has five peaks. The F_2 function is as follows:

$$F_2 = e^{-2\cdot(\ln(2))\cdot\left(\frac{x-0.1}{0.8}\right)^2} \cdot sin(5\pi x)^6 \tag{2.2}$$

In the F_2 function each peak has a different height, see Figure 2.7.

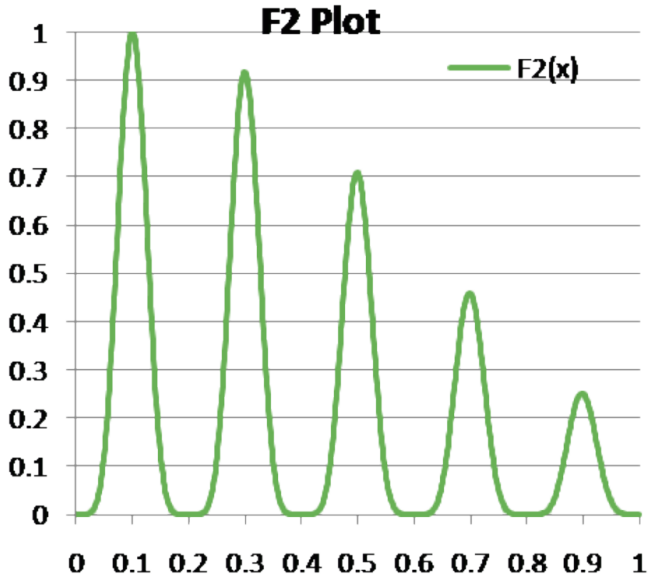


Figure 2.7: Plot of example function $F_2(x)$

For an example of how fitness sharing as proposed in Deb and Goldberg (1989) changes the implied fitness landscape, two different notional populations consisting of ten individuals were created. Their respective fitness values and implied fitness landscapes are shown in Figure 2.8 and Figure 2.9. These graphs assume a sharing distance (σ) value of 0.2. It is easily observable that the fitness landscape is dependent on the population and that it may vary significantly based on the current population. In a diversity management strategy such as fitness sharing, the fitness of each solution in a niche is equal to the best fitness value in the niche divided by the number of solutions therein. Generally having more solutions in a niche causes lower fitness for each member of the niche. For this reason individuals tend to move to other local areas of the landscape. Niching has been primarily used in attempting to solve multimodal problems where multiple optima can be found in a single instance of the GA.

Population P1	
x	F2(x)
0.110	0.928
0.240	0.040
0.260	0.265
0.280	0.690
0.310	0.844
0.380	0.001
0.420	0.001
0.500	0.707
0.720	0.322
0.920	0.172

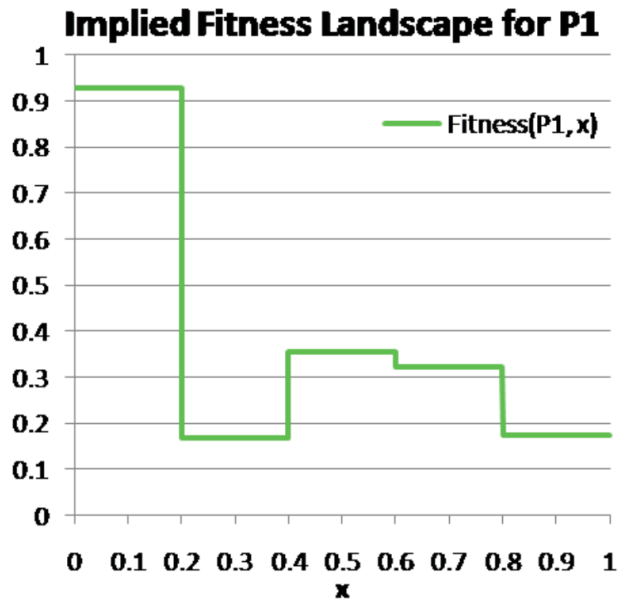


Figure 2.8: Implied Fitness Landscape for random population P_1

Population P2	
x	F2(x)
0.040	0.041
0.090	0.928
0.100	1.000
0.180	0.001
0.310	0.844
0.380	0.001
0.490	0.668
0.500	0.707
0.720	0.322
0.850	0.037

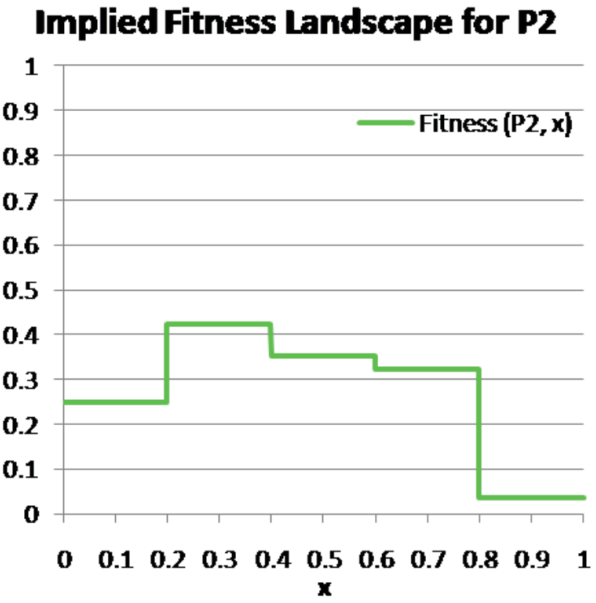


Figure 2.9: Implied Fitness Landscape for random population P_2

2.5.2 Constraint Handling

Problems can be formulated so that not all combinations of allele values are possible, i.e. not all places in the genotype space are valid. Such problems are known as constrained problems. Solutions which violate a constraint are called unfeasible solutions, whereas the

valid solutions are called feasible. Unconstrained optimization techniques are inadequate for solving these problems because they often find too many solutions that are unfeasible, and simply limiting iterative methods at the constraints may easily lead to sub-optimal solutions.

Constraint handling methods used in classical optimization algorithms can be classified into two groups:

- Generic methods that do not exploit the mathematical structure (whether linear or nonlinear) of the constraint.
- Specific methods that are only applicable to special types of constraints.

Generic methods, such as the penalty function method, the Lagrange multiplier method, and the complex search method (Deb, 1999; Ravindran et al., 2006) are popular because each one of them can be easily applied to any problem without much change in the algorithm. These methods are generic, so the performance of these methods in most cases is not satisfactory. However, specific methods, such as the cutting plane method, the reduced gradient method and the gradient projection method (Ravindran et al., 2006), are applicable either to problems having convex feasible regions only or to problems having a few variables because of increased computational burden with large number of variables.

Since GAs are generic search methods, most applications of GAs to constrained optimization problems have used the penalty function approach of handling constraints. The penalty function approach involves a number of penalty parameters which must be set right in any problem to obtain feasible solutions. The idea of penalty was first introduced by Courant (1943). Application of a penalty component to the fitness function for a constrained optimization problem also changes the problem landscape in that unfeasible regions become searchable but generally have poor fitness. In other words, the idea is to transform a constrained problem in an unconstrained problem by adding or subtracting a certain value, i.e. penalty, to or from the fitness function of unfeasible solutions. This searchability opens the possibility for the algorithm to find feasible areas in unexpected places of the landscape, for example in the midst of a large unfeasible region, or pass on genetic traits from unfeasible solutions to offspring. Typically, penalties are applied based on some measure of distance such that the further a solution is away from a feasible region, the more it is penalized. Constraint handling through penalty functions increases the size of the landscape, which in turn may increase the required computation effort, but also has the benefit of not introducing bias from a repair operator and can be useful when it is difficult to generate many feasible solutions. Penalty-based constraint handling techniques for multi-objective is similar to

single objective except that the penalty factor is added to all the objectives instead of only one objective.

The dependency of GA's performance on penalty parameters has led researchers to devise sophisticated penalty function approaches such as multi-level penalty functions (Homaifar et al., 1994), dynamic penalty functions (Joines and Houck, 1994), and penalty functions involving temperature-based evolution of penalty parameters with repair operators (Michalewicz and Attia, 1994). All these approaches require extensive experimentation for setting up appropriate parameters needed to define the penalty function. Michalewicz (1995) describes the difficulties in each method and compares the performance of these algorithms on a number of test problems. In a similar study, Michalewicz and Schoenauer (1996) concluded that the static penalty function method (without any sophistication) is a more robust approach than the sophisticated methods. Static penalty depends only on the degree of violation, whereas if the penalty function depends also on the current generation count, it is called dynamic penalty (Joines and Houck, 1994). In adaptive penalty (Farmani and Wright, 2003) information gathered during the search process is used to control the amount of penalty added to unfeasible individuals. An self-adaptive penalty function keeps track of the number of feasible individuals in the population to determine the amount of penalty added to unfeasible individuals. If there are a few feasible individuals in the whole population, a larger penalty factor is added to unfeasible solutions.

Another typical constraint handling strategy is the *validity assured*. Only feasible solutions are evaluated for fitness by the GA, so it requires an operator for repairing unfeasible solutions. The repair operator, however, is not always able to complete this task if constraints are very stringent. Crossover operations are often specialized to repair solutions violating constraints. Repair operators may also be required after mutation if mutations produce unfeasible solutions. Depending on how the repair operator works, the individual presented to the operator and the closeness of the feasible space, repair operators may fix solutions to be very alike or dissimilar to the original solution. Often repair operations are implicit to the crossover and mutation operators and as such they may be indistinguishable.

Repair mechanisms have the possibility of changing the problem landscape because they turn unfeasible solutions into feasible solutions. The new solution may generally replace the original unfeasible individual with a certain probability. The question of replacing repaired solutions is related with the so-called Lamarckian evolution, which assumes that an individual improves during its lifetime and that the resulting improvements are coded back into its chromosome (Whitley et al., 1994). Good repair operators maintain the essential properties of a solution, but sometimes even a well-designed repair operator may make

it more difficult to find certain solutions. Consider for instance when the GA generates an unfeasible solution close to an optimal solution. The repaired solution could be fixed to be the optimal solution, but it could also be repaired in such way to be placed far from the optimal solution. Thus, the repair function impacts how easy it is for a GA to search certain areas of the landscape. The impact of various repair operators on problem hardness is most often studied by comparison of crossover or mutation methods that integrate these operators. Since repair operators are problem-formulation-specific, research into the effect of those operators must be explored on an operator-by-operator and problem-by-problem basis.

Deb et al. (2002) proposed a constraint-handling method using the binary tournament selection, where two solutions are picked from the population and the better one is chosen. The following rules apply to the selection:

- If one solution is feasible and the other one is not, the feasible solution is selected;
- If both solutions are feasible, the one which dominates the other is selected;
- If both solutions are unfeasible, the one with a smaller overall constraint violation is selected.

This method is often referred to as the *superiority* of feasible solutions because any feasible solution is always better ranked than any unfeasible solutions. Moreover, unlike in a similar approach proposed by Fonseca and Fleming (1998), unfeasible solutions are also ranked instead of simply being placed as members of the same non-dominated front. This, in fact, prevents the algorithm from wandering in the unfeasible region for more generations without reaching the feasible space through constraint boundaries. This approach has been shown to be particularly efficient with real-coded GAs (Deb et al., 2002).

The ϵ-constraint handling method was proposed by Takahama and Sakai (2005) and it is similar to Deb's superiority of feasible solutions methods. The main difference is the relaxation of the constraint violation during the early stages of the search process using the ϵ parameter. As some useful information may be contained in the infeasible solutions with relatively small overall constraint violation, the relaxation of the constraint violations may include more unfeasible solutions in the population during the early stages of evolution. The ϵ value is gradually reduced until the generation counter reaches a certain number. After this number of generations, the ϵ value is set to zero and this method becomes the same as the superiority of feasible solutions methods.

2.6 Initialization

The initial population is the first step of a GA, i.e. the first generation of individuals which should mate and create offspring. Each member of the initial population encodes a possible solution to the problem. In most cases the initial population is created randomly (or pseudo-randomly) with no particular restrictions and without compromising the possible algorithm convergence, i.e. its ability to find the best trade-off solutions. Naturally, it can be argued that starting from a population in a search region where the probability of finding the optimal solution(s) is high may lead to a quick evolution and convergence. The drawbacks are that this requires some advance knowledge on the problem landscape and it may result in a detrimental accumulation of repeated solutions without uncovering any unexpected good results in other areas. In fact, the initial population and any restarted populations have the potential to bias a GA managed search. For instance, if certain alleles are not represented in the initial population, they may only be evaluated by the GA by having mutation stumble upon them.

For these reasons, for most GAs it is preferred that the initial population is random and well distributed over the entire search space, which can be obtained with quasi-random low discrepancy point generation sequences such as Sobol (1979). Since even a random population can be biased due to stochastic error on a given trial, many GAs employ a restart function to prevent a single initial population from skewing the entire search. Space filling methods or minimum distance between the initial individuals as suggested in Michalewicz and Fogel (2000) can help prevent random errors from causing the initial population to be skewed to a subset of the search space. Similar considerations should be applied when determining the size of the initial population. Researchers usually argue that a small population size may hinder a sufficient exploration of the search space (Pelikan et al., 2002; Piszcz and Soule, 2006), whereas a large population may result in an excessive computational effort for the creation of each generation and thus reduce the algorithm convergence to the optimal solutions (Lobo, 2011; Lobo and Goldberg, 2004). The decision on the appropriate population size is thus a trade-off of feeding the algorithm with enough chromosomes to find good solutions (Goldberg et al., 1991).

Initial studies seemed to find a correlation between the problem dimension (i.e. number of decision variables) and the consequent exponential growth of the number of individuals in a population. However, later studies have refuted such correlation by demonstrating that populations with 30, 50 or 100 individuals, defined as the most common dimensions, are suitable for most problems (Goldberg and Rudnick, 1991). Other works have in turn shown

the adequacy of very small populations, especially for binary-coded GA. Michalewicz and Fogel (2000) and others have suggested that the initial population can include a heuristic for finding good initial starting solutions by using heuristics. However, Eiben and Smith (2003) and others have argued that heuristic initialization does not significantly improve GA performance as good solutions are generally found quickly by the GA. The argument points out that the focus of the GA is frequently on finding the smaller improvements in the already good solutions since little resources are wasted on finding the good initial solutions.

2.7 Solution Generation: Classification of Genetic Operators

GAs create new individuals by recombining the characteristics of the existing individuals and then randomly changing some of their genes. As explained in Section 2.1.4 these are the basic GA operators: crossover and mutation. Initial population generation, parent and survivor selection, repair operators in constrained GAs and migration operators in spatially oriented GAs (Tomassini, 2005) have also an important contribution in GA optimization. It is also common to see repair operators independent of mutation and recombination and to see implementation of local search operators that are applied after solution creation. The study of memetic algorithms which combine local search procedures with a GA structure has received much interest in light of the No-Free-Lunch theorem (Wolpert and Macready, 1997). Local search procedures and repair operators are typically invoked after either or both mutation and crossover. Local search and repair operators are generally classified separately from the mutation and crossover operators when a distinctly independent set of code is used for them. The use of local search procedures often causes quick convergence and loss of diversity (Eiben and Smith, 2003) and as such these operators should be classified separately when possible.

There are at least five major diversity categories based on which GA operators can be classified: preserving vs. promoting, directness, occurrence pattern, randomness and complexity. Of these, complexity, randomness, and occurrence pattern are general categories that qualify operators. In contrast, preserving vs. promoting and directness are inspired by the operator's approach to diversity. Complexity and randomness are generic operator qualifiers.

The main question is what makes some problems computationally easy and other problems hard. The former are efficiently solvable, on the contrary the latter cannot be solved efficiently and sometimes at all. The best, the worst and the average case complexity

refer to three different ways of measuring problem complexity, and the genetic operators can be classified according to their performance in those cases. Software complexity theory has been discussed in detail by Sipser (1997). Randomness refers to that fact that some elements of the GA search process are random and others are not. For instance, parent and survival selection are often deterministic (unless the random selection is used), whereas crossover and mutation are stochastic. Diversity methods such as the *Cross-generational Probabilistic Survival Strategy* (CPSS) in *Diversity Control oriented Genetic Algorithm* (DCGA) can use a stochastic process in different areas so we qualify operators accordingly, as in Shimodaira (1997).

Occurrence pattern refers to the fact that operators are executed at different times in the GA and can thus be classified according to their patters of occurrence. For example, a section of code can occur only once during the entire search process, single occurrence at multiple points or repeatedly at a given point, or in every repeated step that the algorithm performs. These correspond to the four major categories of occurrence: one-time execution, cyclic execution, phased execution and recurring execution. One-time executed code is simply code that is only executed once. Cyclic events are singular events that occur once every time a triggered condition is met. Restarts are typically cyclic in that they happen based upon one or more termination conditions. Phase-based code is executed when the algorithm is in a certain phase of the overall search strategy. For example, the *Diversity-Guided Evolutionary Algorithm* (DGEA) (Ursem, 2002) uses distinct diversification and intensification phases to control mutation and recombination respectively. In most GAs recurring operations include survivor selection, parent selection, mutation and crossover.

Operators can be classified according to their general approach to diversity, i.e. whether they promote it or just preserve it. Abbass and Deb (2003) defined the promotion of diversity as the creation of genetic material that may be different from all of the possible material created by the recursive recombination of the current population members. Thus algorithms that are diversity-promoting must be capable of adding new genetic material to the population. Diversity preserving methods, on the other hand, are those that try to maintain diversity which already exists in the population. Diversity promotion typically happens through the restart of an algorithm or through the use of mutation. Diversity preserving methods work only on the current genetic material in a population. Every operation in a GA that does not introduce random genetic material impacts diversity preservation.

Directness refers to how diversity is measured. Operators can be thus *direct* or *indirect*. Direct diversity methods are those methods that use a metric of diversity to control or guide their process. An example of a direct method of diversity control is the incest control

mechanism in CHC (Eshelman, 1991) where individuals do not mate unless they are at least a certain threshold apart. A second example of a direct method is the survival selection in DCGA (Shimodaira, 1997) that eliminates duplicate solutions and uses the distance from the best solution for selecting survival candidates. Indirect methods are more common. The majority of operators in GA literature are indirect because direct methods have an overhead associated with explicitly measuring diversity.

CHAPTER 3

MULTI-OBJECTIVE GENETIC ALGORITHM FOR STRUCTURED INPUTS

In general GAs are particularly appreciated because of their stochastic nature (Deb et al., 2002; Michalewicz and Fogel, 2000; Turco and Kavka, 2011). As already explained in Chapter 2, they introduce a certain degree of randomness in the search process, make the search less sensitive to modeling errors and try to escape any local optima to converge to the global optimum without making additional assumptions about the underlying fitness landscape. GAs are generally applied to solve global optimization problems and used in many fields, including engineering, life sciences and economics. A specific GA implementation, however, highly depends on the problem to be solved, as it transpires from the many different GA customizations available in the literature. Usually this customization is translated into specific optimization procedures or genetic operators that use information contained in the problem in order to generate better results. This is fully consistent with Wolpert and Macready (1997), who suggest that under very broad conditions all search algorithms perform on average in exactly the same way. In other words, if there are problems for which algorithm A is better than B, then there must be problems for which B is better than A. In formal terms, there is "no free lunch" when the probability distribution on problem instances is such that all problem solvers have identically distributed results. However, an inadequate problem formulation may be the cause of a failure too, as shown for the GAs in Radcliffe and Surry (1995). The size of a problem instance, i.e. the number of decision variables and constraints, may also have an impact on the GA performance.

In this context, the key contribution of this work is to introduce an optimization framework able to automatically detect the GA implementation that is best tailored for the problem under examination. This framework is referred to as Multi-Objective Genetic Algorithm for Structured Inputs, or MOGASI. The aim is to maintain all performance characteristics belonging to general purpose state-of-the-art GAs, while incorporating additional custom modules designed for handling specific problem characteristics and for increasing its performance, such as the convergence rate, when these characteristics are present.

MOGASI's main feature is the ability to activate custom modules as a reaction to the problem peculiarity arising from the different decision variable characteristics and from the problem formulation. This enables the algorithm to combine the most suitable GA module alternatives and consequently increase its performance. More specifically, the objective is to effectively handle optimization problems with the following main characteristics:

- *continuous and discrete variables*: the algorithm is able to handle also other variable types but this work only considers the continuous and discrete domains.
- *variables' bounds*: all decision variables have an interval of values where they are considered acceptable and which is defined by a lower and an upper bound.
- *multi-objective*: the algorithm is able to solve problems with several objective functions and correctly handle incomparable Pareto optimal solutions.
- *constrained problems and unconstrained problems*: they are both taken into consideration. In the former case, the algorithm handles both linear and non-linear inequalities and equalities.

As other GAs, MOGASI is able to address the so-called *Black Box Optimization* (Muñoz et al., 2015), which is a popular approach in industrial engineering where the relationships between decision variables, objectives and constraints of the optimization problem are overly complex for a mathematical or analytic formulation. In this context, specialized engineering simulation solvers (e.g. thermodynamic simulations, finite element method simulations, and so forth) are used. As an additional feature, MOGASI improves the standard behavior of GAs by better exploiting all information not hidden inside the black box, i.e., which are explicit and available to the analyst. This point is examined in Section 3.1.

3.1 Black Box Optimization

The Black Box (BB) processes the decision variables in input and computes the optimization objectives and constraints in output. In this optimization scenario all details on the problem formulation which are not expressed in a format readable by the algorithm cannot be used during the optimization. With this in mind, the more the information that can be defined outside the Black Box, the higher the probability to increase the convergence rate of the algorithm. So in a BB optimization framework, a distinction is made between the information that are available to the optimization algorithm and those that are hidden within the BB. In particular, it is assumed that the optimization algorithm knows:

- the values of the decision variables $x \in R^m$ (BB's input vector);
- the lower and the upper bounds l_i and u_i, respectively, of each variables $x_i \quad i = 1,\ldots,m$;
- the analytic form of some equality ($v_a(x) = 0 \quad a = 1,\ldots,A$) and inequality ($w_b(x) \leq 0 \quad b = 1,\ldots,B$) constraints, which are referred to as *input constraints*.

Inside the BB, variables x can be subject to some transformations $z(x) = z_1(x),\ldots,z_K(x)$, where $z_k(x) \in \mathbb{R}, k = 1,\ldots,K$. The analytic form and the final value of functions $z_k(x), k = 1,\ldots,K$ are not known outside the BB, but are used as inputs of the problem's objective functions, which are defined as $f_1(x,z(x)),\ldots,f_n(x,z(x))$, where $f_j(x,z(x)) \in \mathbb{R}, j = 1,\ldots,n$. Similarly, $z_k(x), k = 1,\ldots,K$ values are used as inputs of additional problem's inequality and equality constraints, defined as $g_c(x,z(x)) \leq 0$, where $g_c(x,z(x)) \in \mathbb{R}, c = 1,\ldots,C$ and $h_d(x,z(x)) = 0$, where $h_d(x,z(x)) \in \mathbb{R}, d = 1,\ldots,D$, respectively. Again, the analytic form of all functions f, g and h is not made explicit outside the BB. However, for each input value x the optimization algorithm knows the final value of each function $f_j(x,z(x)), j = 1,\ldots,n$, $g_c(x,z(x)), c = 1,\ldots,C$ and $h_d(x,z(x)), d = 1,\ldots,D$, and hence knows whether the constraints (3.1e)-(3.1f) are satisfied or not (see below).
To summarize, the mathematical formulation of the BB optimization problem under examination is:

$$\min_{x} \quad \{f_1(x, z(x)), \ldots, f_n(x, z(x))\} \tag{3.1a}$$

$$l_i \leq x_i \leq u_i \qquad i = 1, \ldots, m \tag{3.1b}$$

$$v_a(x) = 0 \qquad a = 1, \ldots, A \tag{3.1c}$$

$$w_b(x) \leq 0 \qquad b = 1, \ldots, B \tag{3.1d}$$

$$g_c(x, z(x)) = 0 \qquad c = 1, \ldots, C \tag{3.1e}$$

$$h_d(x, z(x)) \leq 0 \qquad d = 1, \ldots, D \tag{3.1f}$$

where all details of the constraints (3.1b)-(3.1d) are available outside the BB, whereas the analytic form of all functions f, g, h, and z are hidden in the BB. Only the values of $f_j(x, z(x)), j = 1, \ldots, n$, $g_c(x, z(x)), c = 1, \ldots, C$, and $h_d(x, z(x)), d = 1, \ldots, D$ are known outside the BB, for any feasible input value x.

The information related to decision variables (3.1b) are made explicit because they are essential for the correct definition of the BB inputs. In industrial applications these information do not include only variable bounds and variable domain, but also decision variable relations that can be mathematically expressed as constraints. Most of the information contained in the above mathematical formulation remain hidden inside the BB. However, information produced after an evaluation contribute much less to a possible algorithm strategy redefinition than information that is freely produced beforehand. The latter, being defined on decision variables only, do not require any computational effort nor solver execution. For this reason they can be analyzed and made explicit (i.e. exposed out of the Black Box) with little effort on behalf of the user.

Input constraints (3.1c)-(3.1d) can be either linear or non-linear. Assuming that there are q linear equalities and r linear inequalities, these constraints can be written as $V \cdot x = \beta$, with $V \in \mathbb{R}^{q \times m}, \beta \in \mathbb{R}^q$ and $W \cdot x \leq \delta$, with $W \in \mathbb{R}^{r \times m}, \delta \in \mathbb{R}^r$, respectively. MOGASI exploits this structure in the initialization phase, as outlined in Section 3.3.3.

3.2 Modular Algorithm Architecture

In evolutionary computation, modularity is intended as the ability to freely combine and reuse the separable components of an algorithm and it has been intensively studied as a way of improving the innovativeness and the scalability of the evolutionary search (see, e.g., Wagner (1995), Garibay and Wu (2003), Goldberg (1989c)). Evolutionary algorithms,

especially GAs, are particularly suitable for such decomposition and re-composition owing to their module-based structure.

General purpose GAs have standard modules and as such are able to find reasonably good trade-off solutions in most generic optimization problems. However, many authors have demonstrated (see, e.g., Aickelin (2002), Turco and Kavka (2011), Boussaïd et al. (2013)) that custom-composed GA implementations can be used at the same time for multiple purposes or adapted to specific problems, and as such easily outperform standard GAs not explicitly designed for a given target. Indeed, these implementations take advantage of the modularity concept and enrich the generic GA phases with specialized modules that are developed *ad hoc*.

The main phases of a GA with a generic structure are: application of the genetic operators, evaluation, verification of the termination criteria and selection. These phases compose the GA Main Loop, which starts after the algorithm is initialized, as shown in Figure 3.1.

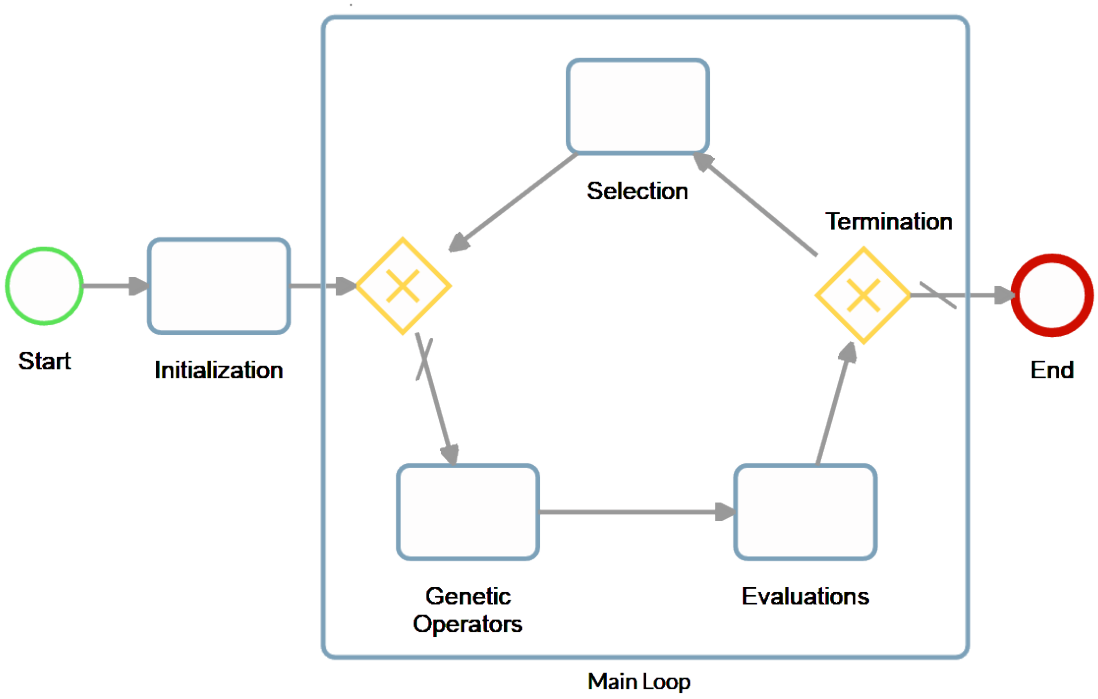


Figure 3.1: Graphical representation of a GA Main Loop with its generic phases

Each phase of the Main Loop can be considered as an autonomous module able to communicate with other modules and each uses the data generated and shared thereby, but otherwise entirely independent from the implementation of other modules. In a custom GA a generic module, or even all of them, can be replaced by specialized modules responding

to precise problem characteristics, or specialized modules can be added thereto for such purposes. This is precisely the idea the MOGASI structure has been based on.

MOGASI has been designed to combine the generic GA phases with specific modules chosen according to the target problem characteristics. The algorithm first analyzes the data structures of the underlying optimization problem and then it applies the most suitable specialized module based on this analysis; in particular, it focuses on the characteristics of the imposed constraints.

The workflow in Figure 3.2 shows how the classic GA structure has been modified and expanded to obtain the MOGASI structure. For this representation the Business Process Model

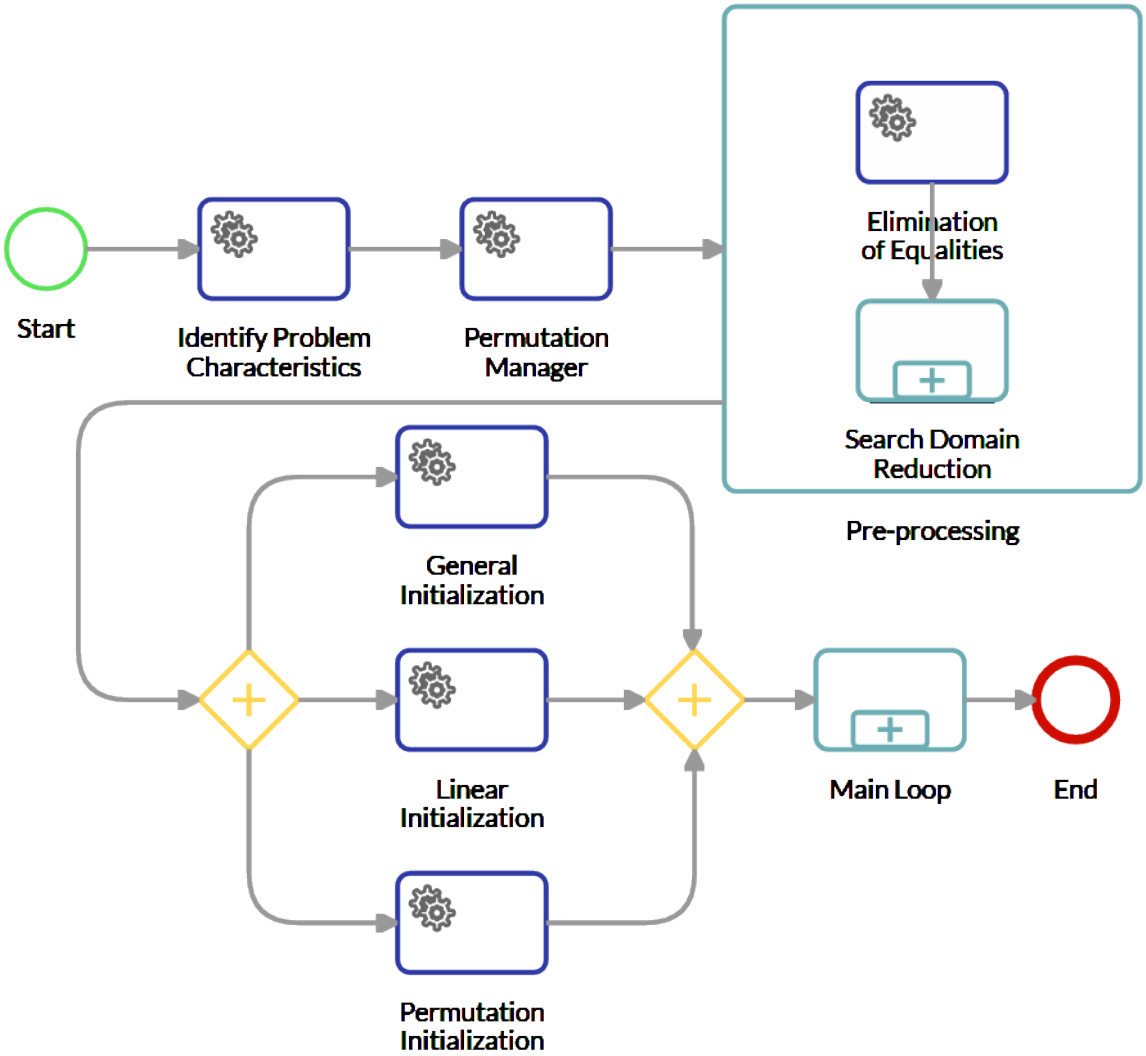


Figure 3.2: Detailed initialization phase before the MOGASI optimization main loop

and Notation 2.0 (BPMN 2.0) (Object Management Group, 2011) standard was deemed most appropriate due its expressiveness and ease of use. It is a standard for drawing diagrams which visually depict a detailed sequence of activities and information flows needed to complete a process, and it is becoming increasingly used in numerous fields, from economics to engineering, from optimization to human resources management. The dark blue rectangles in the BPMN workflows represent low-level modules designed to complete a single specific task, whereas the light blue rectangles with a "+" symbol represent high-level modules incorporating a number of low-level modules and other components designed to complete an entire set of tasks. The yellow diamonds with the "+" symbol indicate the beginning or the end of sub-phases in which different modules are executed in parallel. The yellow diamonds with the "X" symbol represent "if" conditions. The workflow portions included in an "if" condition are iteratively looped until the enclosed task is successfully completed.

Figure 3.2 shows a customized initialization phase starting with the identification of the problem characteristics which shall influence the subsequent choice of strategy. The goal is to reduce as much as possible the problem complexity, in terms of number of variables and constraints (e.g. elimination and appropriate substitution of equality constraints) and decision variable ranges, against a reasonable computational cost to pave the way for the Main Loop. Moreover, a repair module called FIXER (for details see section 3.6) has been inserted before the Main Loop to further improve the quality of the initial points. It is important to underline in the context of this work any reference to permutation (on Figures 3.2–3.3) should be disregarded as it has been designed to respond to different target problems as explained in Costanzo et al. (2014) and therefore not applied herein. The components of the workflow in Figure 3.2 are explained in detail in the following Sections.

Figure 3.3 shows an expanded Main Loop phase, which starts with the evaluation of the fitness of all individuals in a population, followed by the strategies aimed at their ranking, sorting and elimination of redundancies. The double population mechanism, described in detail in Section 3.5, is another MOGASI customization, which exploits the results of the data structure recognition and recombination process implemented in the initialization phase to guarantee data feasibility. The linearity/non-linearity of constraints is also considered with the application of specific genetic operators, applied concurrently with the classic GA and permutation operators in a parallel sub-phase. Finally, a further FIXER module is implemented before the update of the population and, consequently, the end of the Main Loop phase. Each of these modules is explained in the next sections.

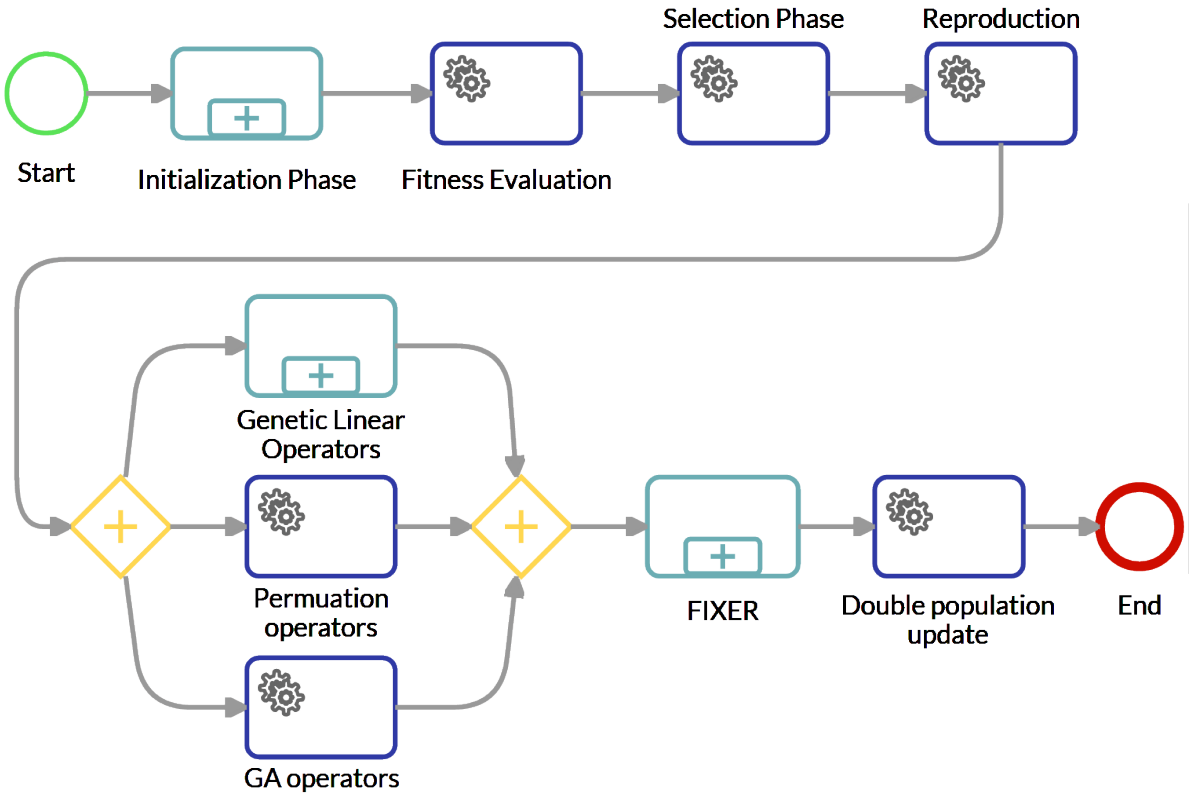


Figure 3.3: Simplified diagram of the MOGASI main loop

3.3 Initialization phase

The processes belonging to the initialization phase are generally aimed at reducing the problem dimensions. The number of variables and constraints considerably influences the complexity of a problem and the convergence speed of the algorithm. In both cases a meaningful dimension reduction can lead to numerous benefits in terms of computational cost in several algorithm phases.

3.3.1 Identification of problem characteristics

As first step during its execution, MOGASI analyzes the details of the optimization problem it has to solve. Even though the problems are coded so that they could be efficiently handled by the MOGASI modular structure, the algorithm uses an object-oriented paradigm to selectively load the problem. Specific problem components, such as the variable characteristics and the presence of constraints, determine the individual's elements for a given execution. Furthermore, they serve as indicators which activate specific functional modules of MOGASI,

enabling thus a flexibility otherwise precluded by the sheer number of possible use cases.

3.3.2 Permutation Manager

The Permutation Manager is a common natural encoding useful in many optimization problems, such as those requiring an optimal ordering of the elements in a set (e.g. the traveling salesman problems, vehicle routing, scheduling problems, etc). The modeling of a variable set which satisfies the requisites to be defined as permutation requires a large number of constraints. For this reason MOGASI contains native modules intended for the management of such encoding and a significant simplification thereof. This topic, however, is not discussed in this work (for further details refer to Costanzo et al., 2014).

3.3.3 Pre-processing

Pre-processing procedures are sequences of simple operations that simplify constraints and reduce the number of decision variables and narrow their bounds with the aim of speeding up the optimization algorithm convergence (Mészáros and Suhl, 2003). MOGASI implements a pre-processing strategy composed of two steps described in detail in the corresponding Sections:

1. *Elimination of the linear equality constraints* in (3.1c): linear equalities, if present, can be used to replace some variables by a suitable linear combination of the remaining variables, Section 3.3.3.1.
2. *Reduction of the search space*: the convex set defined by the set of linear constraints can be made smaller by eliminating redundant constraints and tightening variables' bounds, Section 3.3.3.2.

3.3.3.1 Elimination of linear equalities

A set of $q(\leq A)$ linear equalities in (3.1c) can be represented in matrix form as $V \cdot x = \beta$, where $V \in \mathbb{R}^{q \times m}$, $x \in \mathbb{R}^m$, and $\beta \in \mathbb{R}^q$. Assuming $p(\leq q)$ independent equality equations, there are p variables $x_{i_1}, x_{i_2}, \ldots, x_{i_p} (i_1, \ldots, i_p \subseteq 1, \ldots, m)$ that can be determined in terms of the other variables. The matrix V can be vertically split into two matrices V_1 and V_2, such that the j-th column of the matrix V belongs to V_1 iff $j \in \{i_1, \ldots, i_p\}$, V_1^{-1} exists. Hence, $V \cdot x = \beta$ can be split as:

$$V_1 \bar{x}_1 + V_2 \bar{x}_2 = \beta, \tag{3.2}$$

where $\bar{x}_1 = [x_{i_1}, x_{i_2}, \ldots, x_{i_p}]$ and $\bar{x}_2$ contains the remaining $m - p$ variables. It follows that $\bar{x}_1$ can be calculated as:

$$\bar{x}_1 = V_1^{-1}\beta - V_1^{-1}V_2\bar{x}_2, \tag{3.3}$$

and p constraints (3.1b) can be redefined in terms of $\bar{x}_2$ only as:

$$l_1 \leq V_1^{-1}\beta - V_1^{-1}V_2\bar{x}_2 \leq u_1, \tag{3.4}$$

where $l_1 = [l_{i_1}, l_{i_2}, \ldots, l_{i_p}]$ and $u_1 = [u_{i_1}, u_{i_2}, \ldots, u_{i_p}]$.

Similarly, the linear inequalities in constraints (3.1d) can be redefined in terms of $\bar{x}_2$ only. In fact, $W \cdot x \leq \delta$ (where $W \in \mathbb{R}^{r \times m}, \delta \in \mathbb{R}^r$) can be split as $W_1\bar{x}_1 + W_2\bar{x}_2 \leq \delta$, and hence:

$$(W_2 - W_1V_1^{-1}V_2)\bar{x}_2 \leq \delta - W_1V_1^{-1}\beta. \tag{3.5}$$

The set of all linear constraints is therefore:

$$l_2 \leq \bar{x}_2 \leq u_2 \tag{3.6}$$

$$V_1 l_1 \leq \beta - V_2\bar{x}_2 \leq V_1 u_1 \tag{3.7}$$

$$(W_2 - W_1V_1^{-1}V_2)\bar{x}_2 \leq \delta - W_1V_1^{-1}\beta \tag{3.8}$$

where $l_2 = [l_{i_{p+1}}, \ldots, l_{i_m}]$ and $u_2 = [u_{i_{p+1}}, \ldots, u_{i_m}]$

There is no way of knowing *a priori* which valid split $x = (\bar{x}_1, \bar{x}_2)$ will lead to the best algorithm performance during the optimization. For this reason, in the current MOGASI implementation an elaborate method on how to split the variables and which of them should be eliminated (see Eq. 3.3) is not implemented. The available options for this choice are just two: through an automatic uniform random probability criteria or delegated to a direct interaction with the user who has to pick which variables to eliminate.

3.3.3.2 Search Domain Reduction

Once all linear equalities have been eliminated, the convex space defined by constraints (3.6)-(3.8) can be reduced through a collection of techniques that aim at detecting and

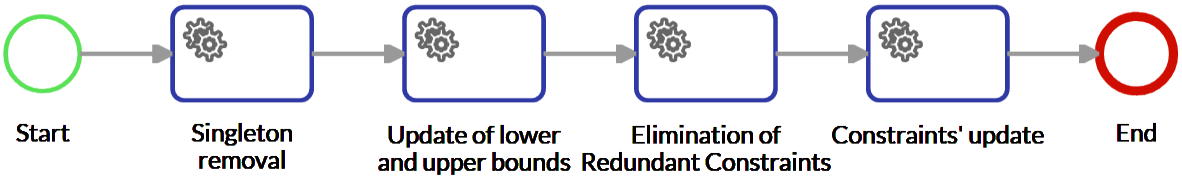


Figure 3.4: Space reduction phases

eliminating redundant constraints and variables, and tightening bounds where possible (Wolsey, 1998). If the resulting search space is smaller, GAs will typically converge much more quickly.

Many techniques are available for this purpose. In this context it is important to point out that a clear and complete formulation of the objective functions is not available to the algorithm, which can only evaluate such objectives in a black box scheme, as explained in Section 3.1. For this reason, this work excludes all those techniques that base their domain reduction process on the analysis of the objective function(s).

Following these considerations, MOGASI implements a space reduction process composed of four steps, as illustrated in Fig. 3.4:

1. *Singleton removal.* It consists in finding constraints involving only one variable and absorbing the information contained within by the remaining constraints;

2. *Update of lower and upper bounds.* It consists in checking the minimum and maximum variable values (upper and lower bounds of their ranges) to re-calculate the reachable constraints' bounds;

3. *Elimination of redundant constraints.* After the application of steps 1 and 2, it may happen that some constraints turn out to be redundant and are therefore discarded;

4. *Constraints' update.* It consists in modifying the constraints' coefficients and restricting the constant terms based on considerations similar to those used in step 2.

The identification of the best balance between the pre-processing benefits and the time spent for pre-processing is often a difficult task. In fact, the iteration of the entire workflow as in Figure 3.4 may be beneficial since the reduction of the search space in step j at iteration t may lead to new simplifications in step $i < j$ at iteration $t+1$. MOGASI adopts a double termination criterion to iteratively apply the pre-processing workflow: fifty complete

iterations or lack of significant changes to the constraints (with the significance threshold set to 10^{-6} for each coefficient refinement).

It is shown that pre-processing can effectively reduce the search space. Tests have been performed on the constrained problems with linear input constraints introduced in Section 3.10. Table 3.1 compares the pre (original) and post (reduced) pre-processing volume of the convex search space.

Table 3.1: Volume reduction by pre-processing on constrained problem

	Original Volume	Reduced Volume	Gain
DEB	4.500E0	4.4445E0	01.23%
SRN	1.600E3	9.3200E2	41.75%
OSY	1.366E2	1.4310E1	89.52%

The volume has been calculated with Matlab software (Language and Computing, 2004) by first defining the constraints in matrix form, then converting the polytope defined by it in a list of vertices K, and finally calculating the convex hull on K and the corresponding area volume bounded by K. Even though these are standard literature benchmark problems, the pre-processing phase has yielded promising results. In particular, the rotated OSY problem (see Sec. 3.10.3.6) has been identified as the most suitable, due to its high number of linear constraints.

The entire MOGASI pre-processing does not burden the algorithm search as much as the real design evaluations performed by industrial simulation software, which is the real computational bottleneck as explained in Section 3.1.

3.3.4 General, linear and permutation initialization

The last step of the initialization phase is the creation of the first generation of individuals. It can be argued that starting from a population in a promising search region may lead to quick convergence. However, this requires some advance knowledge on the problem landscape and it may result in an accumulation of repeated solutions without discovering any unexpected good results in other areas. Three types of initialization methods are dedicated each to specific classes of variables with a determined characteristics. More specifically the General initialization methods are applied to all variables which are not in the other two categories. Linear initialization apply to variables that are involved in the subset of linear constraints and handled as explained in Section 3.3.3. The last category of initialization methods, Permutation, is tasked with creating valid permutations, i.e. each permutation

variable value can appear just once in it and all the values have to belong to the permutation domain.

In general, MOGASI can be initialized with four different methods. It can accept a user-defined initialization configuration set or create the initial population randomly with no particular restrictions but without compromising the possible algorithm convergence. Moreover, each of these configurations can be generated either as full or as incomplete population. In the full population mode the initialization set contains as many individuals as required, whereas in the incomplete population mode the dimension of the initialization set is not relevant, i.e. there can be less individuals than required. The algorithm operation in the latter case does not change, but it only takes more time to saturate the population size. The search for solutions that satisfy the constraints is non-intensive in this phase, but the validity of the initial permutations is mandatory.

Similar considerations should be made when determining the size of the initial population. A population too small may hinder a sufficient exploration of the search space, whereas a population too large may result in an excessive computational effort for the creation of each generation and thus reduce the algorithm convergence to the optimal solutions. Initial studies seemed to find a correlation between the problem dimension (i.e. number of decision variables) and the exponential growth of the number of individuals in a population (Goldberg et al., 1991; Gotshall and Rylander, 2000). However, later studies refuted such correlation by demonstrating that population with 30, 50 or 100 individuals, defined as the most common dimensions, are suitable for most problems (Goldberg and Rudnick, 1991). Other works showed in turn the adequacy of very small populations, especially for binary-coded GAs, but there is no a unique method to determine the best population size.

3.4 Solution Generation: Genetic Operators

GAs create new individuals by recombining the characteristics of the existing individuals and then randomly changing some of their genes. These two processes are translated as the standard GA operators: crossover and mutation. It is a common practice to create custom GA operators for specific problems, but the design of a custom GA operator that guarantees that all individual genes are kept within the constrained solution space is not always possible. In case of linear constraints (Michalewicz and Janikow, 1996), however, this can be achieved very efficiently. For instance, constraints as formulated in (3.6)-(3.8) enable to easily check whether a decision variables' vector x is feasible for all linear constraints, without the need to call the (time-consuming) black box solver. Similarly, when an operator is applied to a

feasible solution x, constraints (3.6)-(3.8) enable to check whether feasibility in the linear domain is preserved or not. Hence, MOGASI enriches the classical operators with additional controls in order to promote the creation of offspring solution vectors that do not violate the constraints (3.6)-(3.8). The simplest, yet most efficient method is the repetition of the operator until a feasible offspring solution is generated. It is also possible to build operators with a structure that guarantees the solution feasibility, such as the arithmetic crossover $(x \cdot a + y \cdot (1 - a))$ of two feasible solutions, x and y, which yields a feasible solution in a convex search space, as long as $0 \leq a \leq 1$. A systematic use of operators that comply with constraints (3.6)-(3.8) decreases the computational load and enables MOGASI to direct its search through the most complex regions (i.e., non-convex). In particular, in problems with non-linear constraints MOGASI searches the non-convex feasible region using the double-population structure, as developed in GENOCOP III (Michalewicz and Nazhiyath, 1995). This topic is discussed in detail in Section 3.5. In this work no formal mathematical representation of all GA operators is provided as they have already been extensively discussed in literature. The formulation of the less known operators is given in Michalewicz and Janikow (1996) and Poloni and Pediroda (1997). The operators inherited from the former paper are rather different from classic operators described in literature. The main reasons can be summarized as follows:

- the solutions are coded in Double-precision floating-point format in a space with real values R^q.
- the value of a vector variable component depends on the values of the remaining components of the vector according to constraints (3.6)-(3.8), so the GA operators are dynamic.
- some GA operators are non-uniform as regards the evolution time, i.e. the result depends on the current progress measured through the generations.

To summarize, MOGASI uses eight different operators selected according to the above-mentioned generic problem formulation (see section 3.1). Four mutation operators have been chosen - simple uniform mutation, forced boundary mutation, single non-uniform mutation and whole non-uniform mutation - as well as four crossover operators: simple N-points crossover, arithmetic crossover, heuristic crossover and directional crossover.

3.5 Double Population Mechanism

MOGASI extends the GENOCOP III (Michalewicz and Nazhiyath, 1995) applicability from single- to multi-objective problems and inherits its double population mechanism, so it works with two separate but communicating populations.

The idea is to optimize the set of decision variables $x_1, ..., x_m \in R^m$. The entire space R^m can be decomposed as $S \subset F \subseteq R^m$, with S defined as *search population* and F as *reference population*. The former is maintained feasible with respect to the convex set defined by constraints (3.6)-(3.8) by means of specialized operators, as explained in the previous section. The latter aims to be feasible with respect to all problem constraints (3.1b)-(3.1f).

3.5.1 Search Population management

MOGASI uses a dedicated system for complex data and keeps problem characteristics separated, as explained in Section 3.3.1. This distinction enables to work normally on S using both the inherited GENOCOP logic for generating search points and the MOGASI internal optimization strategy. With the exception of custom modules, MOGASI can be potentially seen as a standard GA for this sub-problem, as in Figure 3.1. Hence, the search space is a polytope defined by decision variable bounds and linear equalities and inequalities. This permits the adoption of specialized GA operators yielding significant benefits, as exemplified in Section 3.4.

3.5.2 Update Reference Population

The reference population F can be considered as an elite set, which means it is a protected space in which the best solutions are maintained from one generation to the next. Furthermore, at each iteration MOGASI tries to populate the reference space and combine S solutions that do not belong to F with the "population update" mechanism, as shown in Figure 3.3. The generated solutions are not entirely in line with the logic of GAs (i.e. genetic operators), but this phase works instead like a standalone point generator with its own logic.

The mechanism works as outlined in Figure 3.5: each individual in S is picked for the update with a certain probability. This probability can be uniform random or fitness-related defined by a boolean MOGASI parameter. A picked set is defined as $\overline{S}$ and is matched randomly with an individual of F defined as $\overline{F}$. A new set of decision variables $\overline{Z}$ is computed with the following formula until they are either feasible for the input constraints (Eq. 3.1c–3.1d) or until a maximum number of computation attempts is reached:

$$\overline{Z} = a\overline{S} + (1-a)\overline{F} \tag{3.9}$$

The obtained $\overline{Z}$ is evaluated. If it is feasible it is added to F and it can replace the $\overline{S}$ in S with a uniform random probability p_r. Otherwise, $\overline{S}$ can be replaced by $\overline{F}$ with the same probability.

Due to the probability of replacement p_r an individual in the current population can be modified directly in the reproduction phase. This behavior can lead to an evolutionary leap in the algorithm generational progress.

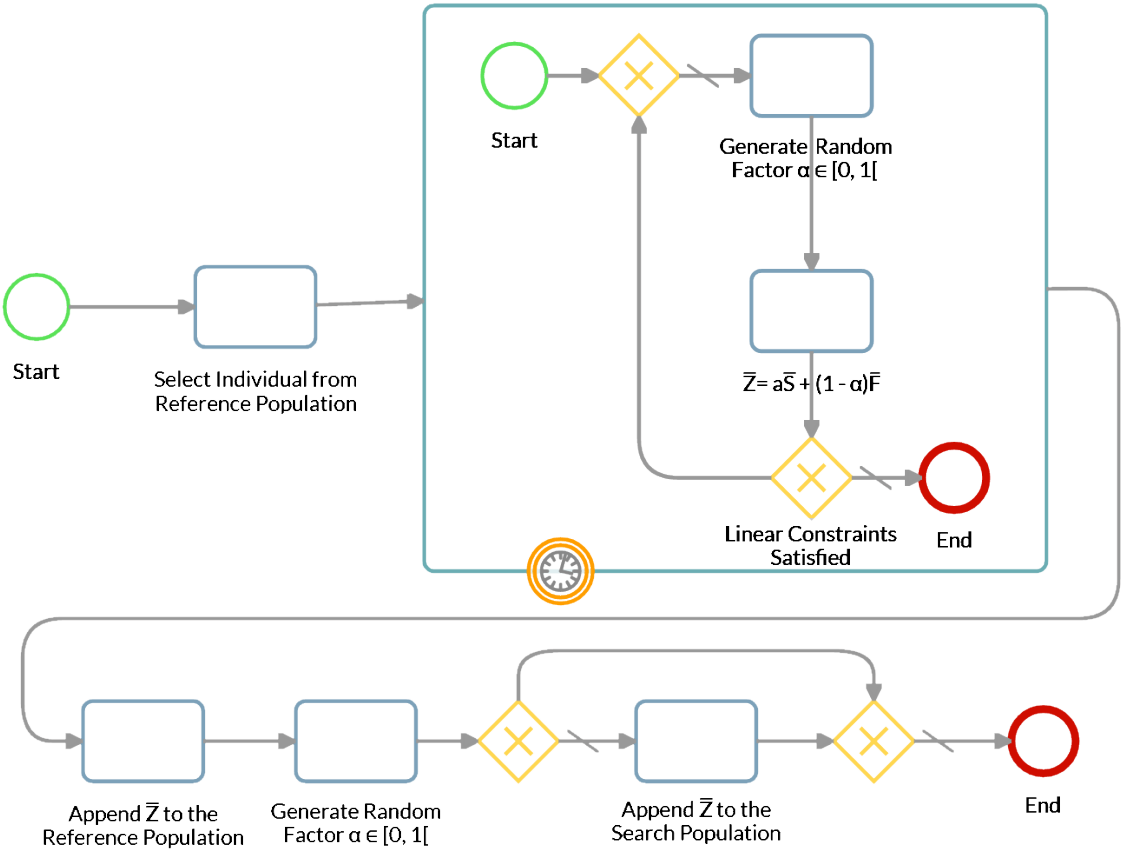


Figure 3.5: Representation of a single iteration of the standalone point generator

Since the adopted structure greatly benefits from variable domains (it preserves the values within the imposed limits) and takes advantage of the linear constraint set, it is rather important that the problems are well formulated. This characteristic benefits from the operation carried out in the pre-processing phase, which is aimed at simplifying the problem.

3.6 Repair Phase

Solution generation operations may break the structure imposed by the double population mechanism. In this context, a generic repair mechanism can play an important role in speeding up convergence as formulated by Zinflou et al. (2010), especially during the standalone point generator phase for the reference population (see Section 3.5.2).

In MOGASI the repair phase is performed in a module called FIXER (see Figure 3.2–3.3), where many strategies cooperate to produce valid and feasible solutions starting from an unfeasible solution. FIXER corrects invalid data structure configurations and/or repairs unfeasible individuals. All the different functions that compose this module are grouped together and applied in the correct order to prevent that the result of a phase is broken during another phase.

Another problem that arises is the possible insertion of an unfeasible solution in the current search population. A common strategy used by GAs for constrained problems is to recognize an unfeasible solution and repair it before this insertion. This approach has a justification inspired by nature: any unfeasible solution should be treated as an immature individual which, in spite of being initially unfit for its purpose, can develop over time into a perfectly acceptable individual. In particular, the replacement of unfeasible solutions with their repaired versions is related to the Lamarckian evolution (Whitley et al., 1994), which assumes that an individual improves during its lifetime and that the resulting improvements are coded back into the genetics of that individual. The issue in this case is whether the next generation should use the original unfeasible individuals or their repaired versions. If only repaired versions were kept, the search could artificially deflect towards local minima and lead to a general robustness loss of the algorithm. Orvosh and Davis (1994) present experimental evidence that suggests that this decision should be stochastic. They report a so-called 5% rule, which states that, when combining a repair procedure with an evolutionary algorithm, the best results are obtained when a small percentage of repaired individuals replaces their original unfeasible versions. The repair function used in the MOGASI repair phase is empirically set to be applied to 15% of the individuals.

3.7 Multi-Objective Approach

In a product design there is usually no single aspect summing up the design essence. Instead, the designer must balance a multitude of factors (such as performance, durability, cost, etc.) and parameters for measuring off those factors against each other to arrive at what he/she

believes is the best combination of properties of the final product. In other words, it is often difficult to fix a single metric in design optimization, but in order to solve the problem properly it is required to simultaneously consider a number of aspects to be improved, i.e. multiple design objectives.

An important concept in multi-objective optimization is the Pareto optimality and the related idea of Pareto dominance. A perfect multi-objective solution is often unreachable, on the contrary a reasonable one is to investigate a set of equally good compromises. A Pareto optimal set (or Pareto front) is a set of solutions that are non-dominated with respect to each other, i.e. no design objective can be further improved without prejudicing another. In other words, while moving from one Pareto solution to another, there is always a certain amount of sacrifice in one objective to achieve a certain amount of gain in the others. Therefore, mathematically speaking, each Pareto optimal point is equally acceptable for a multi-objective optimization problem.

More formally a Multi-Objective Optimization Problem (MOP) consists in finding a vector $x \in R^m$ that optimizes the vector functions $f(x)$, given a vector of objective functions to be minimized $f(x) = f_1(x), ..., f_n(x)$ and a feasible solution space R^m. Given a MOP, a solution vector x dominates x' (depicted as $x \prec x'$) if $f_i(x) \leq f_i(x')$ for all i functions in f, and there is at least one i such that $f_i(x) < f_i(x')$. Hence a Pareto optimal solution can be defined as x^* if there is no vector $x \in R^m$ such that $x' \prec x^*$. Finally, a Pareto Front is defined as the set of $PF^* = \{x^* \in R^m\}$. An example of the above explanation is shown in Figure 3.6.

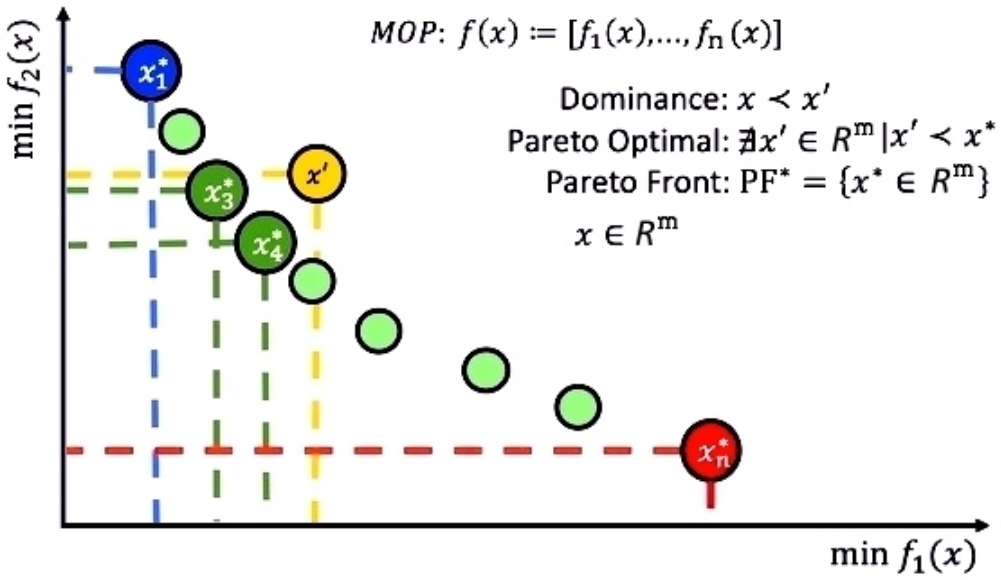


Figure 3.6: Graphical representation of the Pareto Dominance concept (Santiago et al., 2014)

In practical terms, Pareto optimal solution sets are often preferred to single solutions when considering real-life problems since the final solution of the decision-maker is always a trade-off based on a personal or objective preference. For many multi-objective problems the size of the Pareto optimal set is enormous, if not infinite, so identifying it in its entirety is computationally impossible. The designers must therefore focus on a subset of the Pareto solutions, which should be as uniformly distributed and diverse as possible and capture the entire Pareto spectrum, including the extreme ends of the objective functions, to provide the decision-maker with a comprehensive picture of trade-offs.

A classic approach to multi-objective optimization is the so called preference-based method. This method scales a set of objectives into a single objective by multiplying each objective with its relative importance. The single objective becomes a composite function as the weighted sum of the objectives. The weights are assigned by the user so a relative preference factor for each objective should be known in advance, but they are often difficult to define correctly. Moreover, this method is able to find only solutions located on the convex part of the Pareto front. If the Pareto is not convex, the points on the concave parts of the trade-off surface will be missing. Finally, evenly distributed weights do not guarantee evenly distributed Pareto points.

Since this approach has rather many drawbacks in spite of its simplicity, most state-of-the-art GAs implement the Pareto domination principle with a wide trade-off among objectives. This method does not make any assumptions on the importance of the objectives, but attempts to find as many non-dominated solutions which make up the Pareto front as possible. Once the Pareto has been reached, appropriate diversity preservation and promotion methods should be applied to ensure that the front is well covered in all its parts, including the function extremes, and that the solutions are well distributed.

From the user's point of view the final goal of either single- or multi-objective optimization is basically the same: finding one solution to their problem in a reasonable amount of time. Solving problems with multiple objective functions is an intrinsically complex task which can be solved with many different approaches discussed in literature (Miettinen, 1998). Naturally, the issue is quite complicated in this case since it will be faced with possibly hundreds or even thousands or equally good solutions. The final choice involves higher-level information, which is often non-technical, qualitative and experience-driven (Deb, 1999). It is up to the user to make considerations based on his/her knowledge and compare the available solutions before making an informed choice.

Multi-objective methods applied in GAs have proved to be one of the most successful options. In accordance with the methodology proposed by Deb et al. (2002), MOGASI adopts

three procedures to handle multiple objectives:

- individual ranking function;
- crowding distance calculation;
- constrained crowded-comparison operator (CCCO).

The individual ranking function is explained in detail in Section 2.1.5, whereas the crowding distance calculation is illustrated in Section 2.3.2. The CCCO, introduced by Deb et al. (2002), enables the ranking of individuals using the non-dominated sorting and crowding distance strategies. This operator compares pairs of solutions by checking the values of three attributes for each of them: a constraint violation coefficient, the ranking of the front the individual belongs to and the crowding distance ranking of the individual. Based on that attributes it gives preference to one or another to ultimately rank the entire population. This comparison procedure is as follows:

- Between a feasible and an unfeasible solution, preference is given to the feasible solution.
- If both solutions are feasible, preference is given to the solution belonging to the front with a better ranking; if both belong to the same front, preference is given to the solution with the better crowding distance ranking.
- Between two unfeasible solutions, preference is given to the solution with the lower constraint violation coefficient.
- If the above steps give preference to no solution, a random solution is chosen.

The ranking of unfeasible solutions, i.e. those that violate at least one constraint, is done separately as they are the last ones to be taken into consideration after all feasible solutions have been accounted for. A different coding strategy than the standard Deb version has been developed to increase the execution efficiency. For example, the ranking function that divides the designs between fronts has been implemented somewhat differently. This implementation includes a single comparison among all individual pairs of the population. This information is used to build two main data structures for each design: one placeholder strategy focuses on which individuals are dominated, the other focuses on the number of dominating individuals. This enables an iterative dominance placeholder-consuming cycle in order to build all dominance fronts starting only from the information received from the placeholders.

3.7.1 Application to Single-Objective

One of the goals of this work was to enable MOGASI also to work with single objective problems. Considering the properties of multi-objective GAs, several authors have claimed that the use of multi-objective solvers might be helpful for single-objective optimization as well (Abbass and Deb, 2003). The applications of native multi-objective GAs to single-objective optimization problems can be divided in two main categories:

- schemes known as "multi-objectivization", the goal of which is to translate a single-objective problem in a multi-objective problem. This can be done in several ways: by transforming the fitness landscape (Knowles et al., 2001), by translating a constrained single-objective problem in an unconstrained multi-objective optimization problem (Mezura-Montes and Coello, 2008) and by considering population diversity as an objective (Bui et al., 2005);
- algorithms handling both kind of problems using the same procedures.

Multi-objectivization can be applied outside the GA framework, i.e. in the problem definition phase which precedes the algorithm run. For this reason MOGASI belongs only to the second category since the already presented CCCO is feasible-compliant (Deb et al., 2002): after the ranking routine, feasible designs are always preferred over the unfeasible ones. These criteria produce seemingly trivial yet effective results and speed up the convergence rate of the algorithm: the ranking function creates a list of individuals ordered based on the single objective value, whereas the crowding distance calculation is disregarded because each front consists of a single individual. This makes the method suitable also for single-objective problems and its performance is tested in Section 3.10.2.

3.8 Termination Criteria

The definition of the conditions to stop the run of a GA can have different alternatives. The final choice of the method depends largely on the context. The most common and widespread termination criteria are the following:

- maximum number of generations;
- definition of a maximum time limit for the run;
- maximum number of evaluations of the fitness value;

- comparison between the average fitness value of the entire generation and that of the best solution. The algorithm stops when the fitness values of the last computed generations are largely at the same levels, which indicates that no significant improvement has been registered over several generations.

The framework presented in this work implements all the four described options but as default setting only the first and the third termination criteria are enabled. The other two criteria are specifically implemented for the benchmark tests presented in Chapter 5.

3.9 Parameter Summary

This section summarizes the exposed parameters of the proposed algorithm. All parameters related to the optimization problem setup and the definition of data structures which are not addressed in this PhD thesis (see for example Section 3.3.2) have not been included in the following list. The parameters used in each benchmark or application section are presented in tabular form (e.g. Tables 3.4, 3.7 and 6.7). The following list includes brief explanations that can be useful for reading the parameter tables:

- *Population Size*: this parameter consists in the maximum reachable number of individuals of the Search Population for each generation (Section 3.5). This value sets the size of the Reference Population too, always equal to half of the Search Population size.

- *Number of Generations*: one of the termination criteria (Section 3.8) and extensively discussed in Chapter 2.

- *Maximum number of Evaluations*: it consists in the maximum number of fitness value calculations of the Black Box solver (Section 3.8).

- *Operator probability*: this parameter can have two values, uniform or user-defined probability. In the former case the application probability in the offspring generation of any genetic operator (Section 3.4) is uniform. As an alternative, the user can define all 8 different probabilities with an absolute value.

- *Auto scaled operator probability*: the auto scaled mode is effective when the user-defined option of the previous parameter is selected. If it is set to true, all the probabilities are normalized and their sum is equal to 1.

- *Individual probability distribution*: this parameter influences the parent selection and can assume two values: uniform or cumulative. If uniform is set, all the current individuals have the same probability to be elected as parents. Otherwise, the probability to be elected is proportional to their goodness in the population ranking and defined by the next parameter.
- *Q individual distribution*: this value is used only if the individual probability distribution is set to cumulative. If it is so, the marginal probability to be elected as parent associated to each individual $\{i_0, i_1, i_2, ..., i_n\}$ is calculated as $\{Q, Q(1-Q)^1, Q(1-Q)^2, ..., Q(1-Q)^n\}$.
- *Initialization*: it can be performed in four variants described as all combinations of user-defined or random points that can satisfy the required population size (Section 3.3.4).
- *Replacement probability*: it is a specific probability involved in the Double Population Mechanism (Section 3.5).

3.10 Mathematical Benchmarks

MOGASI has been designed to yield high performance on a wide variety of problem types. Two categories in particular can be distinguished: global optimization problems and problems with specific identifiable characteristics, e.g. presence of linear equalities constraints. As explained in previous Sections of this Chapter, the modularity of MOGASI enables the creation of a high number of different execution configurations. This, however, requires many benchmark problems to properly test the algorithm.

This section presents a set of universally accepted benchmark problems that include an increasing number of structures the algorithm has to deal with. The objective is to show that MOGASI can perform well on all instances compared with MOGA-II and NSGA-II, the current state-of-the-art GAs, and that its performance improves on target test cases (see Section 3.1).

MOGA-II: Multi-Objective Genetic Algorithm II (Poloni and Pediroda, 1997) is an improved version of the multi-objective genetic algorithm (MOGA) that uses a smart multi-search elitism. Elitism is applied in the survival selection phase: the best non-dominated individuals are stored in the elite set, which is used jointly with the new parents to create the next generation. MOGA-II is designed to work with discrete variables, whereas continuous

variables are discretized internally. Constraints are dealt with a penalty policy: a penalty function proportional to the constraint violation is added to (or subtracted from) the objective values, which places solutions violating constraints always with a lower rank in the selection phase. A tolerance added to the penalty function ensures that not all unfeasible solutions are discarded. Beside the classic one-point crossover, MOGA-II also uses directional crossover, which assumes that a direction of improvement can be detected based on the fitness values of individuals in the same generation.

NSGA-II: Non-dominated Sorting Genetic Algorithm II (Deb et al., 2002) is a fast and elitist multi-objective genetic algorithm developed at Kanpur Genetic Algorithms Laboratory (KanGAL). Elitism implemented in this algorithms ensures that all non-dominated solutions discovered up to the current point are preserved. Diversity and spread of solutions are guaranteed without the use of sharing parameters: instead, a suitable parameter-less niching approach is adopted. More precisely, NSGA-II uses crowding distance, which estimates the density of solutions in the objective space, and the crowded comparison operator, which guides the selection process towards a uniformly spread Pareto frontier. The constraint handling method does not make use of penalty parameters. Instead, a modified definition of dominance for an efficient solving of constrained multi-objective problems is used. NSGA-II is able to handle both discrete and continuous variables with the appropriately modified mutation and crossover operators.

Among the high number of existing GAs the choice has fallen on these two algorithms because they are used as general-purpose GA worldwide for industrial and academic applications. Furthermore, the NSGA-II algorithm is quoted in literature more than any other GA and the majority of benchmarks used in this chapter became well known precisely in relation with NSGA-II. The performance of MOGA-II, NSGA-II and MOGASI has been tested and compared on well-known benchmark functions for optimization taken from the Deb benchmark library (Deb et al., 2002), from the work of Osyczka and Kundu (1995) and from the Michalewicz benchmark library (Michalewicz and Fogel, 2000). The results are presented below. The test problems have the following characteristics:

- Unconstrained problems
- Constrained problems with
  - linear constraints (inequalities and equalities)
  - non-linear constraints

  - mixed constraints

- Single-objective problems
- Multi-objective problems
- Many objectives problems (>3)

Since the test problems are rather self-explanatory and do not require a detailed description, the focus has been moved to a correct assessment and comparison of the results they yielded using the three abovementioned algorithms.

3.10.1 Performance Metrics

In industrial engineering optimization problems finding one (single-objective) or more (with multiple objectives) exact solutions of an unknown new design is practically impossible. Indeed, the existing evolutionary algorithms are able to find solutions in the promising regions which are only approximations of the optimal solution/Pareto set, i.e. which are as close as possible to the optimal solution/front. The goal is to use the technique able to reach the best possible approximation of a problem solution/s, but this raises the question of what criteria should be used to validate the performance of the different alternatives and choose the most appropriate one. Algorithm performance is based both on the quality of these approximations and on the computational effort, measured in time or number of evaluations of the fitness function, required to reach these quality levels. No universal performance indicator exists, although many equally valid alternatives have been developed and discussed in literature.

3.10.1.1 Single-Objective Metrics

In case of single-objective optimization, the issue of the quality of solutions is rather simple as it can be easily defined in terms of the objective function which has to be either minimized or maximized. In case of multi-objective optimization the quality of solutions is influenced by multiple factors discussed below. In any case the main difficulty is not so much related to the solution quality, but rather in the stochastic nature of GAs which makes the results of a run unpredictable. For this reason a reliable and meaningful comparison usually requires multiple runs.

To assess the goodness of the results a graphical performance representation is selected for the Single Objective Benchmark problems, specifically the Empirical Attainment

Function (EAF) introduced by López-Ibáñez et al. (2009). Generally, the EAFs describe the probabilistic distribution of the outcomes obtained in the objective space by a stochastic algorithm. The results are plots of summary attainment surfaces and differences between the first-order EAFs. These plots can be used for exploring the performance of stochastic local search algorithms and can show up to two axes. Even though they are most often applied to the analysis of the results of a stochastic algorithm and bi-objective problems, they are equally efficient for representing single-objective problems and offering detailed performance information. As shown in Figure 3.7, the abscissa shows the number of evaluations that the GA performed instead of a value analyzed by the EAF (usually an objective to be minimized), whereas the actual objective is plotted on the ordinate. In this way the chart simultaneously shows the best, the median and the worst performance of the algorithm on the entire run. Furthermore, if the chart is read as a history chart, it is also possible to assess the algorithm convergence speed.

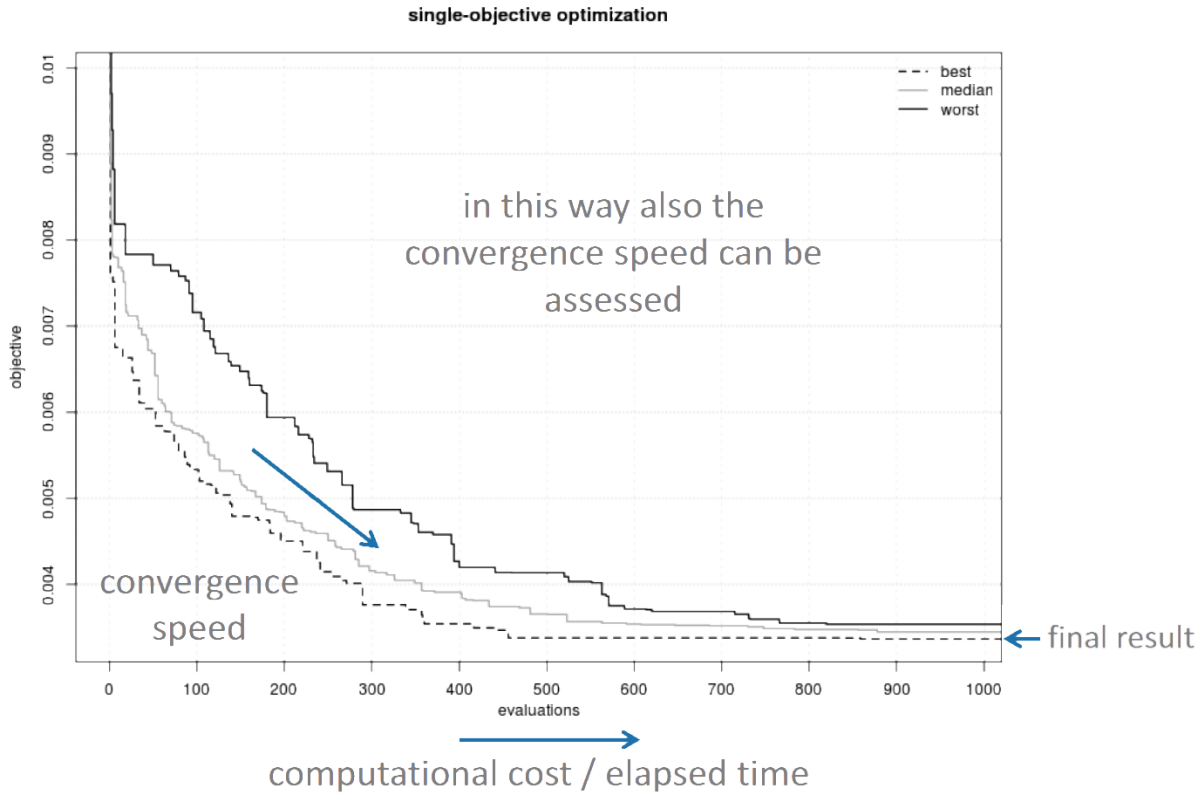


Figure 3.7: Empirical Attainment Function for Single-Objective (EAF-SO)

Another way of reading this chart is to draw a vertical threshold line to determine the objective values that can be obtained with a target number of evaluations or, in other words, guaranteed performance for a certain computational cost. This also enables to identify the

best, the median and the worst performance achieved by the analyzed stochastic algorithm. For example, the chart in Figure 3.8 has been cut on the 300th evaluation on a total of 1,000 evaluations and the objective values at that point can be observed. Similarly, if a horizontal threshold line is drawn, the chart shows the number of evaluations required for obtaining a target objective value. For example, the chart in Figure 3.9 has been cut at the objective value of 0.005: to reach it, the algorithm needs 130 evaluations in the best scenario, 167 in the median scenario and 273 in the worst scenario. Any lower objective value threshold would require more evaluations and vice versa, any higher allowed threshold would require less evaluations.

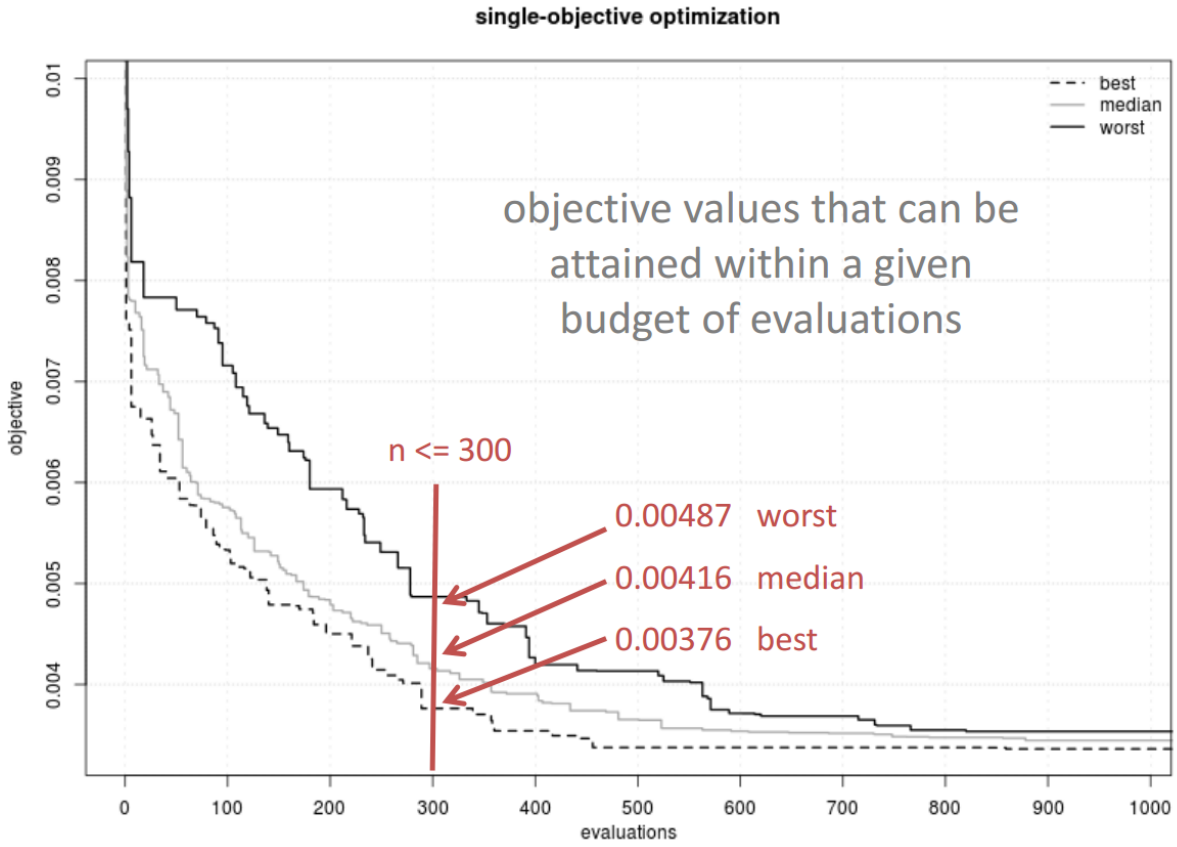


Figure 3.8: Vertical Cut explanation on EAF-SO

The EAF chart can also be used to represent the results of repeated experiments with a stochastic algorithm on a constrained single-objective problem. If the chart is cut horizontally, an additional information can be obtained as shown in Figure 3.10. In this case the chart shows the number of evaluations of the algorithm best, median and worst performance that are required to find the first feasible solution. On the contrary, in the previous two EAF

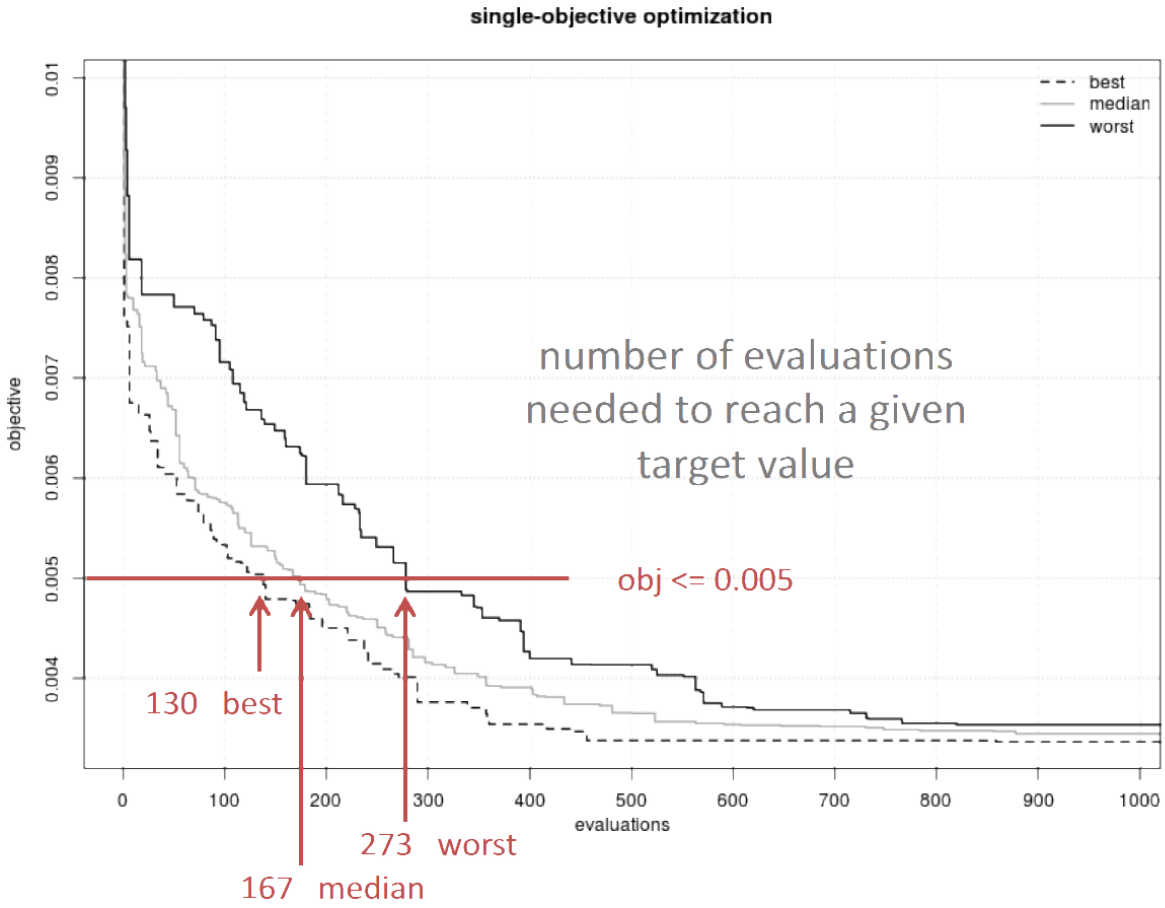


Figure 3.9: Horizontal Cut explanation on EAF-SO

figures where the three lines are concentrated in the upper left part, it can be seen that in all the runs some feasible solutions have been generated in the initial evaluations.

The final feature used to analyze the results of single-objective problem instances is a simple expansion of the abovementioned. The information used for creating single EAF charts is combined to produce a very effective two-algorithm comparison EAF, as shown in Figure 3.11. The elements plotted on each of the axes on both charts are the same: objective 1 on the x axis and objective 2 on the y axis. In SO cases the former is replaced with the number of evaluations. The chart on the left represents the behavior of Algorithm 1 with respect to Algorithm 2, the chart on the right represents exactly the opposite. The three EAF lines (i.e. best, median and worst) are replaced by the equivalent information relative to the performance achieved by both algorithms. In each plot the black areas are those in which the plotted algorithm completely dominates the other one (100%), whereas the areas with different shades of gray indicate the percentage (the darker the area, the larger the

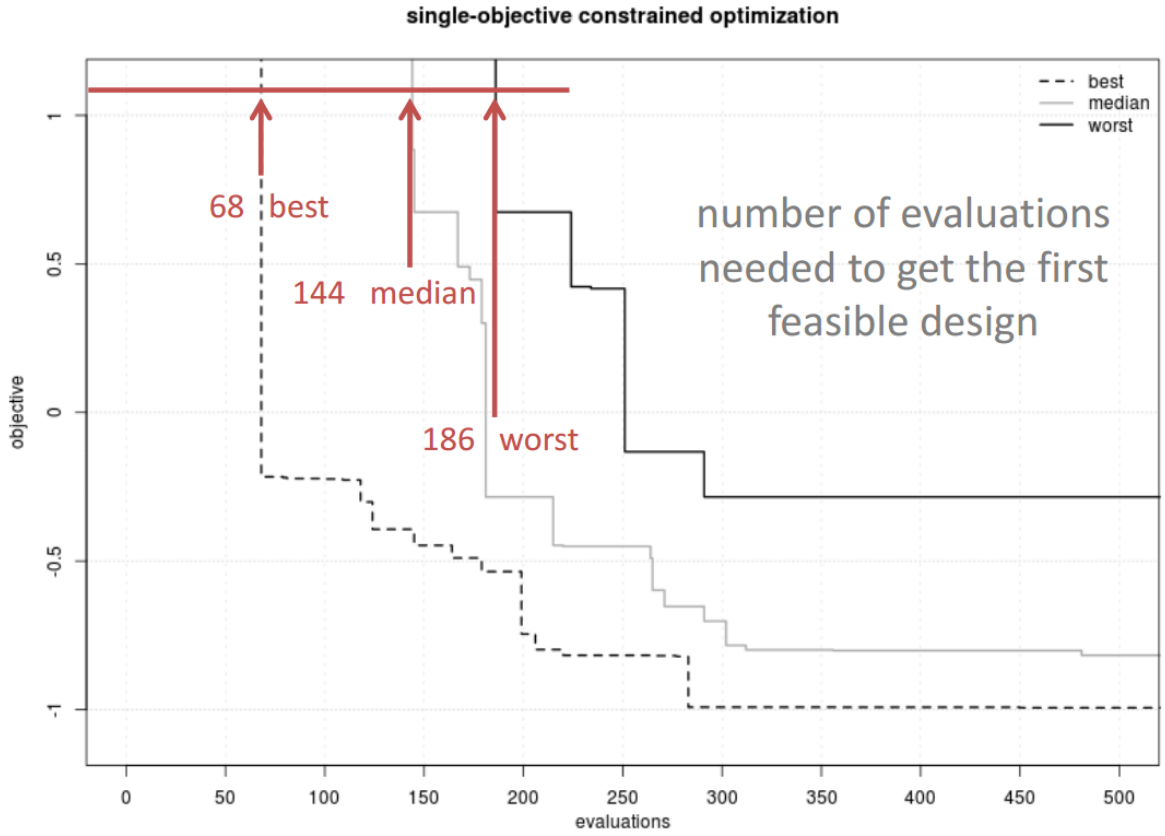


Figure 3.10: Additional considerations on the constrained problem on EAF-SO

percentage) by which the results of the plotted algorithm dominate the results of the other algorithm and its legend. For example by analyzing the chart on the right, a black area can be observed in the bottom right corner. This means that Algorithm 2 produces better solutions then Algorithm 1 in that area of the objective landscape.

3.10.1.2 Multi-Objective Metrics

In case of multi-objective optimization one of the main difficulties is the fact that algorithm identifies a set of solutions instead of a single solution and that the number of equally good solutions that could exist in theory is virtually infinite. This considerably complicates the definition of quality which can thus involve, but not necessarily be limited to the following:

- *accuracy*, i.e. how close the solutions are to the true Pareto front
- *coverage*, i.e. if the generated solutions cover the entire Pareto front considering the minimum and maximum ranges of all objectives

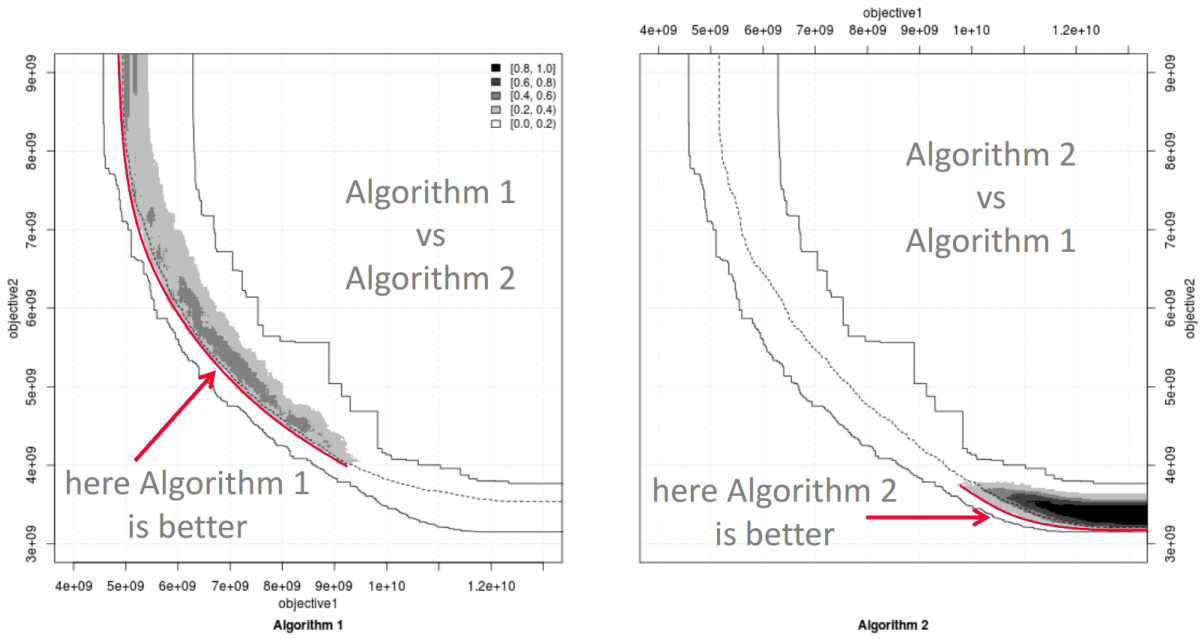


Figure 3.11: EAF comparison of two algorithms: darkest areas indicates a better performance of the plotted algorithm

- *distribution*, i.e. if the generated solutions are well spread over the entire Pareto front or they are concentrated in certain regions, whereas other Pareto front areas remain unexplored and unexploited
- *number of non-dominated solutions* generated by the algorithm
- *convergence rate*, i.e. how many fitness function evaluations are required to reach the Pareto front, which is closely related to the algorithm computational effort

It is not possible to devise a single indicator able to fully capture all aspects of the algorithm performance. The researcher should in fact treat also this issue as a multi-objective problem and find trade-offs by giving more importance to one aspect in place of another. Finally, the same indicator (or multiple indicators focusing on different aspects) should also be applied to other algorithms and their values compared with the indicator values of the algorithm under examination. In this way it is not only possible to determine which is better, but also by how much and, if none is better than any other, what are the aspects in which an algorithm outperforms another. In this context, a distinction should be made between generic and special-purpose GAs, which is independent from the choice of performance indicator. Whereas the former are able to find some acceptable solutions for all or at least

most problems, the latter are designed for niche problems with specific characteristics. Therefore, for a more universal validation of the algorithm the benchmark problems used for the comparison should include both target and non-target problems. Most of the existing performance indicators assume that the true Pareto front of the multi-objective optimization problem is known. In this way the front generated by the algorithm under validation and the true front can be compared to obtain a distance or error value indicating the algorithm efficiency. The following is a short overview of some performance indicators proposed in literature:

- *Error Rate* (Van Veldhuizen, 1999), which indicates the percentage of solutions in the generated dataset which do not belong to the true Pareto front.
- *Generational Distance* (Van Veldhuizen and Lamont, 1998), which computes the Euclidean distance of each solution in the generated dataset from the closest solution of the true Pareto front, and then calculates the average distance.
- *Spread* (Deb and Sachin, 2002), which focuses on the distribution of solutions of the generated dataset in the non-dominated region.
- *Hyper-volume* (Zitzler et al., 2003), which computes the volume of the objective function space covered by the generated non-dominated solutions.
- *Coverage Function* (Zitzler et al., 2003), which computes for a pair of generated datasets the fraction of solutions from one set weakly dominated by one or more solutions from the other set.

To assess the goodness of the identified Pareto fronts a performance metric has been carefully selected to enable a good coverage of the aforementioned qualities. The Inverted Generational Distance (IGD) (Zitzler et al., 2003) is an extension of the Generational Distance indicator (Van Veldhuizen, 1999). This metric is able to measure the distance of the found non-dominated front from a reference sample of the Pareto front and to collect information on the accuracy and the uniformity of the computed front. Low values of the IGD indicator (lower than or close to one) indicate a good approximation of the reference front under both aspects. The metric is defined as follows. Let P^* be a set of uniformly distributed points on the true Pareto front in the objective space (reference set). If A is a computed set of points approximating the Pareto front, then the IGD from P^* to A is:

$$IGD(A, P^*) = \frac{\sum_{p \in P} d(p, A)}{|P|} \tag{3.10}$$

where $d(p, A)$ is the minimum Euclidean distance between p and all points in A. Both diversity and convergence of the approximated set A can be measured using $IGD(A, P^*)$. This metric is more accurate if P^* has many well-distributed points.

3.10.2 Single-Objective Optimization: Michalewicz Benchmarks

The optimization problems presented in this section have been taken from the Michalewicz benchmark library (Michalewicz and Fogel, 2000). The choice has fallen on this set because they are the official benchmarks of the GENOCOP algorithm family, so they were deemed the most appropriate to test MOGASI as general-purpose GA incorporating some characteristics of that family. Seven problems have been chosen: t01, t02, t06, t12, t13, t17, t26. For each problem each algorithm was run fifty times in order to obtain robust results without changing any input parameters other than the seed of the random sequences which were used to generate the initial populations of each run. The Empirical Attainment Functions have been chosen as the most appropriate performance metric for this category of benchmarks and used in this section for the creation of all charts. The parameters of all three algorithms are set to their default values for all problem instances (see Tables 3.2–3.4) to avoid possible performance gaps that might occur due to parameter tuning. The only difference is the *test problem 02* (t02): due to its complexity it was necessary to double the number of generations for all algorithms raising the total evaluation number to 70,000.

Table 3.2: Single-Objective Problem: NSGA-II parameters

Description	Value
Population Size	70
Num.Generations	500
Max Evaluations	35,000
Crossover prob.	0.9
Real Mutation prob.	1.0/nvar
Distribution Index for operators	20.0

The focus is on the comparison of the solutions quality found by the algorithms. Moreover a time-based comparison of algorithm performance is usually interesting in case of different algorithm structures. However, MOGASI, MOGA-II and NSGA-II share the same nature and require no computationally intensive operations, so the predicted computational cost is negligible and is almost the same. It is important to stress that a pure time-based comparison would not have been entirely fair due to the fact that MOGASI has been implemented in a batch standalone module, whereas the two competitors are part of a commercial

Table 3.3: Single-Objective Problem: MOGA-II parameters

Description	Value
Population Size	70
Num.Generations	500
Max Evaluations	35,000
Directional Crossover prob.	0.5
Mutation prob.	0.1
DNA Mutation Ratio	0.05
Selection prob.	0.05

Table 3.4: Single-Objective Problem: MOGASI parameters

Description	Value
Population Size	70
Num.Generations	500
Max Evaluations	35,000
Operators prob.	uniform
Auto scaled op.prob.	true
Initialization	random
Replacement prob.	0.15
Ind. prob. distribution	cumulative
Q ind. distribution	0.1

optimization platform with many options and graphical support that slow down their execution. Hence, the presented algorithm would easily win in a time-comparison, which would be misleading, and for this reason no results related to execution times are presented.

3.10.2.1 Test problem: t01

The first problem taken from the Michalewicz's optimization suite has four variables, two linear and three non-linear constraints, and consists in minimizing a non-linear objective function. It is in fact a good example of a mixed problem in spite of its simplicity. In particular, the objective function is extremely smooth and the constraints define a rather enclosed, simple and easy-to-explore area. The exact mathematical formulation of this problem is shown in Eq. 3.11.

$$
\begin{aligned}
& f(x) = min \;\; x_0^2 + x_1^2 + 2 \cdot x_2^2 + x_3^2 - 5 \cdot x_0 - 5 \cdot x_1 - 21 \cdot x_2 + 7 \cdot x_3 \\
& g_1(x) = x_0 - 2 \cdot x_2 \leq 1 \\
& g_2(x) = x_0 + x_3 \leq 5.1 \\
& g_3(x) = 8 - x_0^2 + x_1^2 - x_2^2 - x_4^3 \geq 0 \\
& g_4(x) = \sin(\frac{x_1 \cdot x_3}{2}) + 0,1 \geq 0 \\
& g_5(x) = 2 \cdot x_0 \cdot x_1 + x_3 \geq 0 \\
& x_0 \in [-1, 7] \\
& x_1 \in [0, 4] \\
& x_2 \in [-7.32, 1] \\
& x_3 \in [-1, 1]
\end{aligned}
\tag{3.11}
$$

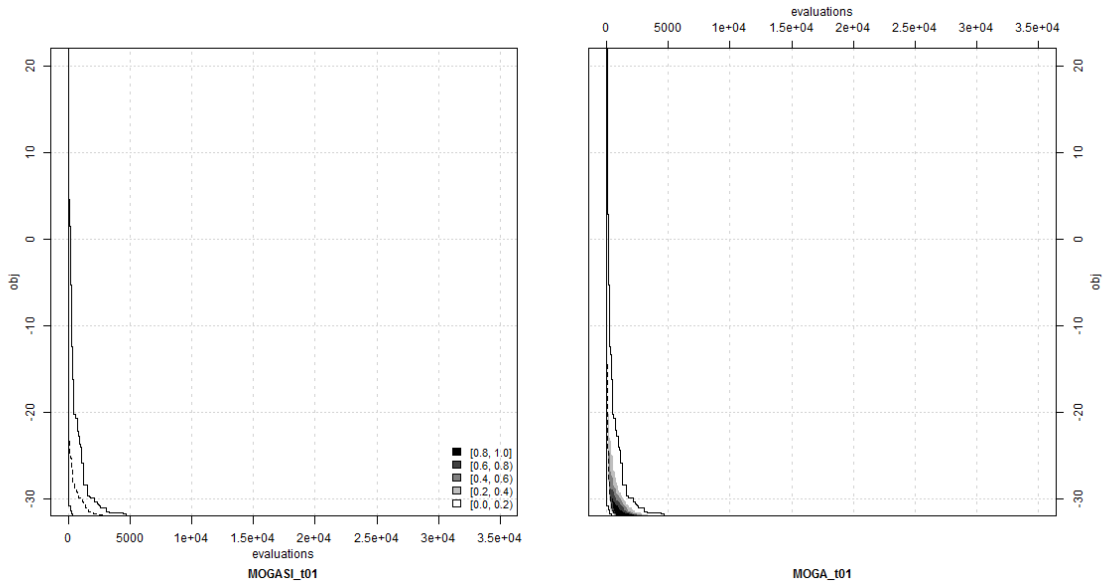


Figure 3.12: T01 problem: MOGASI vs MOGA-II

All algorithms perfectly converge to the optimal value within the given evaluation budget, as shown in Figure 3.12 and 3.13, since the lowest plotted value for the objective function axis is the global optimal value of the T01 problem. Indeed, the three compared algorithms converge perfectly to the optimal value within 5,000 evaluations. In this case, in spite of the presence of some barely visible gray and black shading on the right charts on both figures for some of the initial evaluations, the convergence behavior of all three algorithms is totally

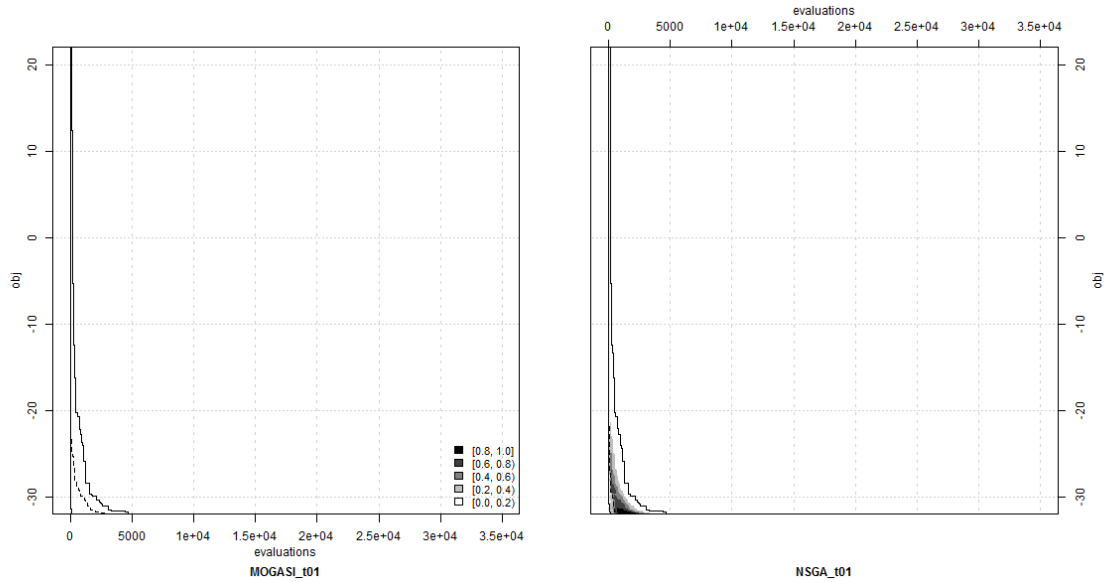


Figure 3.13: T01 problem: MOGASI vs NSGA-II

comparable. The differences resulting from this run, no matter how small, are mainly due to the simple search space of the problem. Nevertheless, this experiment was useful for demonstrating the comparability of the results and the ability of all three algorithms to achieve perfect convergence well before reaching the fixed evaluation number limit.

3.10.2.2 Test problem: t02

The second test problem is often used for single-objective benchmarking due to the complexity of the objective landscape and its multi-dimensionality. It has twenty variables and a non-linear maximization objective, as well as two constraints, one of which is linear and the other non-linear. The exact mathematical formulation of this problem is shown in Eq. 3.12.

$$
\begin{aligned}
f(x) &= max \quad -\frac{\left[\sum_{i=0}^{19}\cos(x_i)^4\right] - 2\cdot\left[\prod_{i=0}^{19}\cos(x_i)^2\right]}{\sqrt{\sum_{i=0}^{19}(1+i)\cdot x_i^2}} \\
g_1(x) &= \prod_{i=0}^{19} x_i \geq 0.75 \\
g_2(x) &= \sum_{i=0}^{19} x_i \leq 150 \\
x_i &\in [0,10] \quad \text{with} \quad i = 0,\ldots,19
\end{aligned}
\tag{3.12}
$$

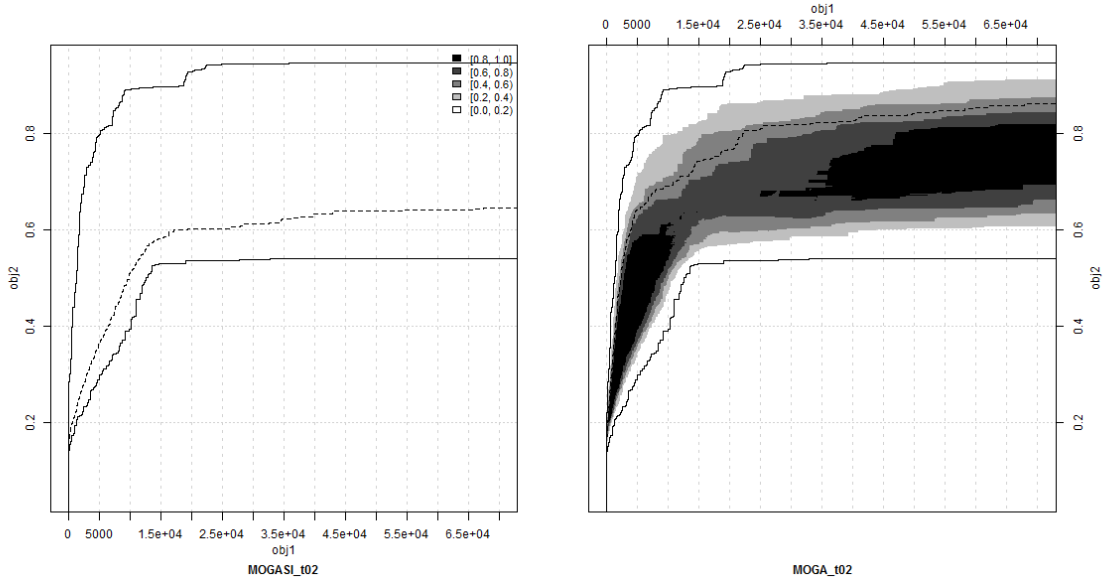


Figure 3.14: T02 problem: MOGASI vs MOGA-II

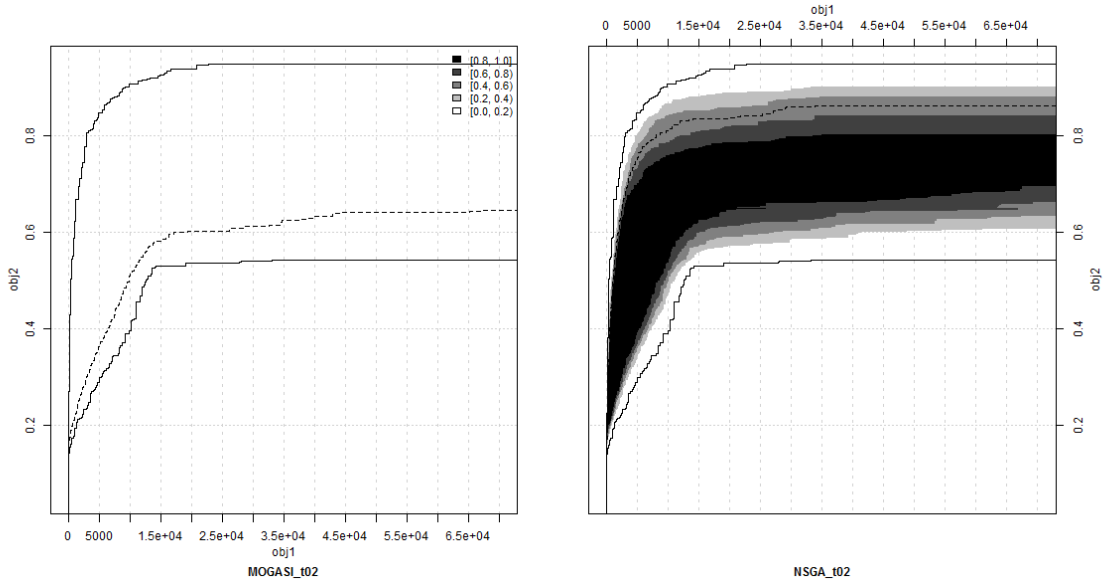


Figure 3.15: T02 problem: MOGASI vs NSGA-II

As shown on Figures 3.14–3.15, the non-linear nature of this problem results in an heavy disadvantage of MOGASI in the search space exploration. The light gray coloring in the upper part of the competitor charts shows low result robustness, whereas the dark coloring in favor of the competitors indicates the fact the MOGASI is not able to efficiently escape the multiplicity of local optima and reach the global optimal region. To this end a detailed

study of this kind of issues will be conducted in order to increase the exploration effort of the algorithm in the presence of non-linear elements.

3.10.2.3 Test problem: t06

The t06 test problem is much simpler than the others which can be seen from its short mathematical formulation (3.13) and low cardinality of decision variables (only three). It is a minimization problem with a linear inequality constraint and a linear equality constraint. The exact mathematical formulation of this problem is shown in Eq. 3.13.

$$
\begin{aligned}
& f(x) = min \;\; 0.2 \cdot x_0^2 + 0.08 \cdot x_1^2 + 0.18 \cdot x_2^2 + 0.1 \cdot x_0 \cdot x_1 + 0.04 \cdot x_0 \cdot x_2 + 0.06 \cdot x_2 \cdot x_1 \\
& g_1(x) = x_0 + x_1 + x_2 = 1000 \\
& g_2(x) = -0.14 \cdot x_0 - 0.11 \cdot x_1 - 0.1 \cdot x_2 \leq -120 \\
& x_i \in [0, 500] \quad \text{with} \quad i = 0, 1, 2
\end{aligned}
\tag{3.13}
$$

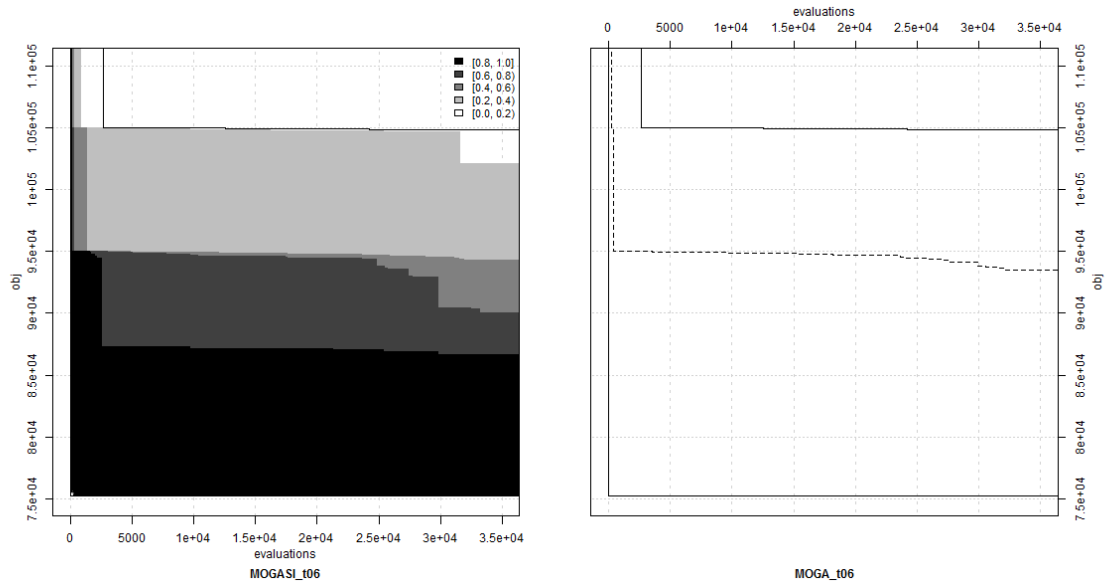


Figure 3.16: T06 problem: MOGASI vs MOGA-II

Figure 3.16 clearly points out the many difficulties of a general purpose GA in correctly handling and extracting some fundamental information from the linear equality constraints. The sharp borders between the different shades of gray represent different domination percentages of the solutions found by MOGASI with respect to MOGA-II on the total number of evaluations. This indicates significant difficulties of the latter algorithm in uncovering

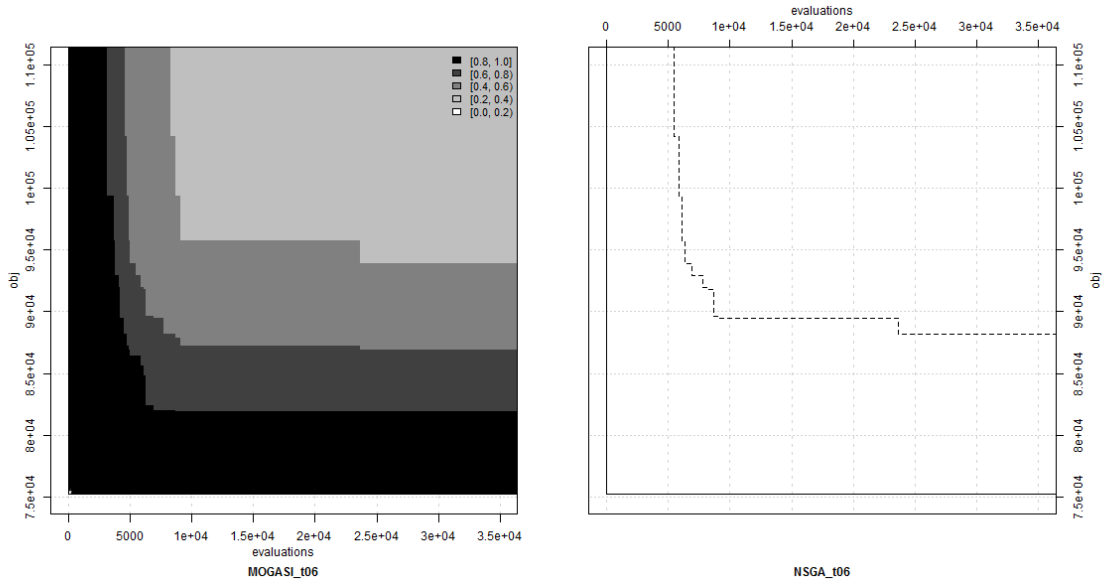


Figure 3.17: T06 problem: MOGASI vs NSGA-II

promising feasible search space regions in a high number of cases. Although this phenomenon is also present in the MOGASI–NSGA-II comparison charts (see Figure 3.17), it is less pronounced. The convergence curve has in fact much fewer steps. The dark coloring in the lower part of the MOGASI charts on both figures indicates that after about a mere hundred evaluations in the majority of runs MOGASI was able to find solutions close to the optimum. This is most probably the combined outcome of the problem reduction achieved as a result of the Elimination of Equalities, see Section 3.3.3.1, and of the fact that being both constraints linear, MOGASI's search population mechanism is able to explore very quickly only the feasible region, unlike its two competitors.

3.10.2.4 Test problem: t12

The t12 test problem can be intended as a more complex version of the former problem, at least regarding its size. In fact, even though t12 consists in maximizing a single objective, it has eight variables and five constraints, three of which are linear inequalities and two of which are linear equalities. The exact mathematical formulation of this problem is shown in Eq. 3.14.

$$
\begin{aligned}
& f(x) = max \;\; x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \\
& g_1(x) = x_0 + x_3 + x_5 + x_6 + 2 \cdot x_7 \leq 20 \\
& g_2(x) = x_2 + x_4 + x_5 + 2 \cdot x_6 \leq 25 \\
& g_3(x) = x_1 + x_3 + x_4 + 2 \cdot x_5 \leq 11 \\
& g_4(x) = x_0 + 2 \cdot x_1 + 2 \cdot x_7 = 12 \\
& g_5(x) = x_2 + x_3 = 5 \\
& x_i \in [0,8] \quad \text{with} \quad i = 0, \ldots, 7
\end{aligned}
\tag{3.14}
$$

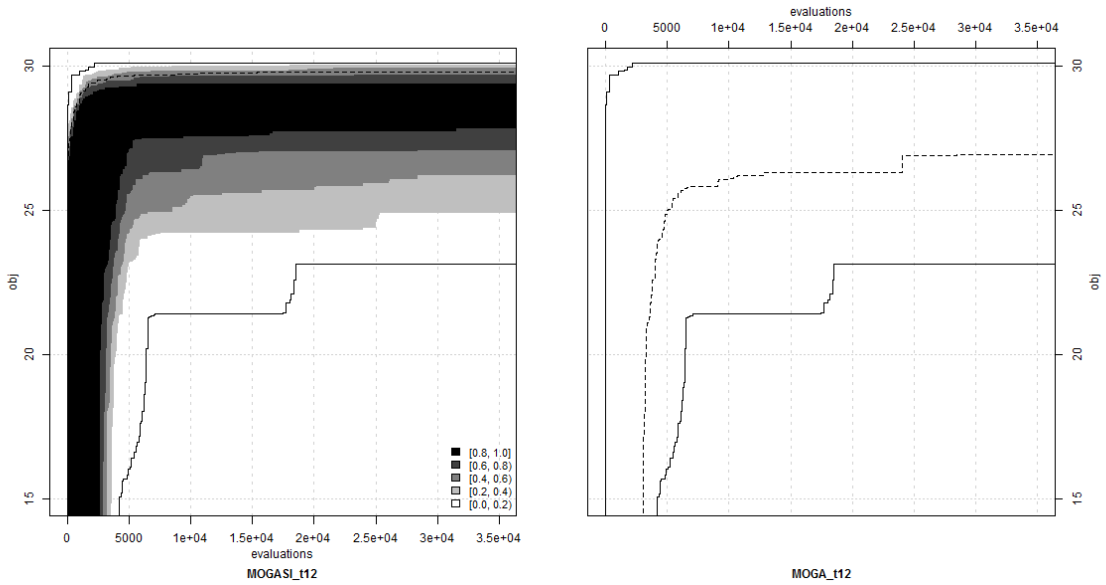


Figure 3.18: T12 problem: MOGASI vs MOGA-II

Figure 3.18 shows the effectiveness of the algorithms under examination. The coloring differences in the upper part of the first chart indicates some difficulties of MOGASI to rapidly converge to the global optimum, even though the differences with the results obtained by MOGA-II are still quite remarkable in favor of MOGASI. The situation resulting from the charts shown in Figure 3.19 is much more straightforward. The lines representing the median and the worst algorithm behavior are all together absent and the chart is almost entirely black. This shows that in most cases the NSGA-II was not able to find any feasible solutions and it was thus not possible to compute and plot these lines. This is mainly due the presence of equality constraints which MOGA-II and NSGA-II have extreme difficulties handling despite their simplicity.

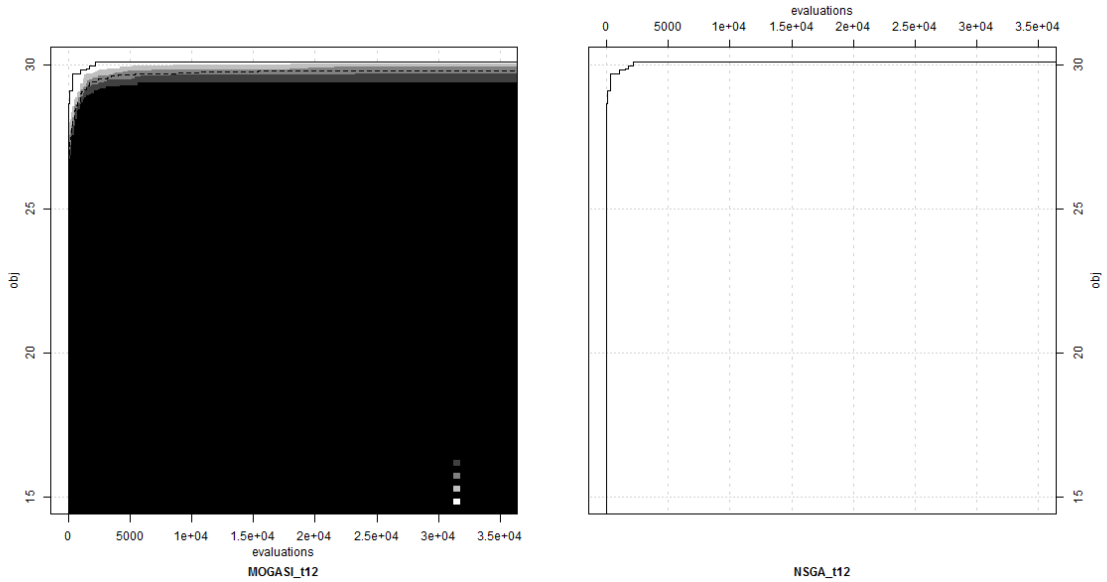


Figure 3.19: T12 problem: MOGASI vs NSGA-II

3.10.2.5 Test problem: t13

After demonstrating the difficulties caused by equality constraints on the performance of general purpose GAs such as MOGA-II and NSGA-II, the next test problem, t13, presents a maximization objective, four variables and only one linear equality constraint. The exact mathematical formulation of this problem is shown in Eq. 3.15.

$$
\begin{aligned}
& f(x) = max \;\; x_0 \cdot x_1 + x_2 + x_3 \cdot x_4 \\
& g_1(x) = x_0 + x_1 + x_2 + x_3 + x_4 = 10 \\
& x_i \in [0, 10] \quad \text{with} \quad i = 0, \ldots, 4
\end{aligned}
\tag{3.15}
$$

Figures 3.20–3.21 show that despite what appeared to be a simple problem formulation, neither MOGA-II nor NSGA-II were able to approximate the optimal solution like MOGASI. The two competitors have a similar behavior to the previous case in dealing with constrained search spaces such as this one. More specifically, MOGA-II is strongly dominated by MOGASI and in the majority of cases NSGA-II was again not able to find any feasible solutions, with the exception of a very small batch of evaluations in the total of 50 runs. As regards the performance of MOGASI, a small dominance difference can be observed with respect to the previous test case which exhibited a light gray shading in the convergence region not present here.

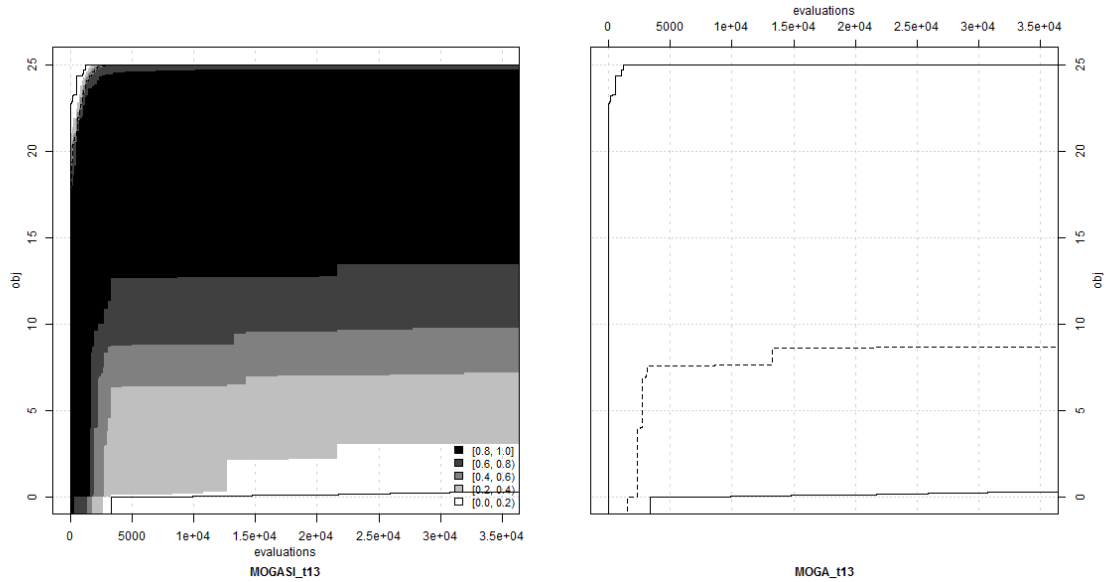


Figure 3.20: T13 problem: MOGASI vs MOGA-II

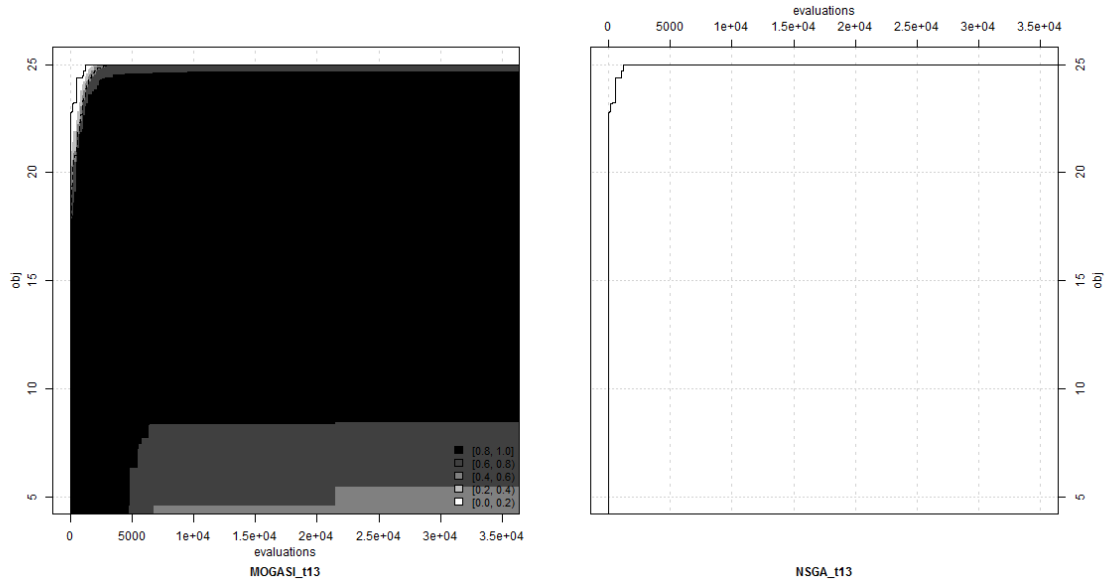


Figure 3.21: T13 problem: MOGASI vs NSGA-II

3.10.2.6 Test problem: t17

The t17 test case is a further attempt to solve non linear problems with equality constraints. This problem has a minimization objective, ten variables and three constraints, as shown in mathematical formulation in Eq. 3.16. According to the description contained in the test library, the search space is more complex but the constraints should be easier to handle

by general purpose GAs. Given this perspective, the main trial for MOGASI is showing an efficient simplification of no less than three constraints.

$$
\begin{aligned}
& f(x) = min \sum_{i=0}^{9} [x_i(c_i + \log(\frac{x_i}{\sum_{i=0}^{9} x_i}))] \\
& \text{with} \quad C = [-6.089, -17.164, -34.054, -5.914, -24.721, \\
& \qquad -14.986, -24.100, -10.708, -26.662, -22.179] \\
& g_1(x) = x_0 + 2 \cdot x_1 + 2 \cdot x_2 + x_5 + x_9 = 2 \\
& g_2(x) = x_3 + 2 \cdot x_4 + x_5 + x_6 = 1 \\
& g_3(x) = x_2 + x_6 + x_7 + 2 \cdot x_8 + x_9 = 1 \\
& x_i \in [0.005, 1] \quad \text{with} \quad i = 0, \ldots, 9
\end{aligned}
\tag{3.16}
$$

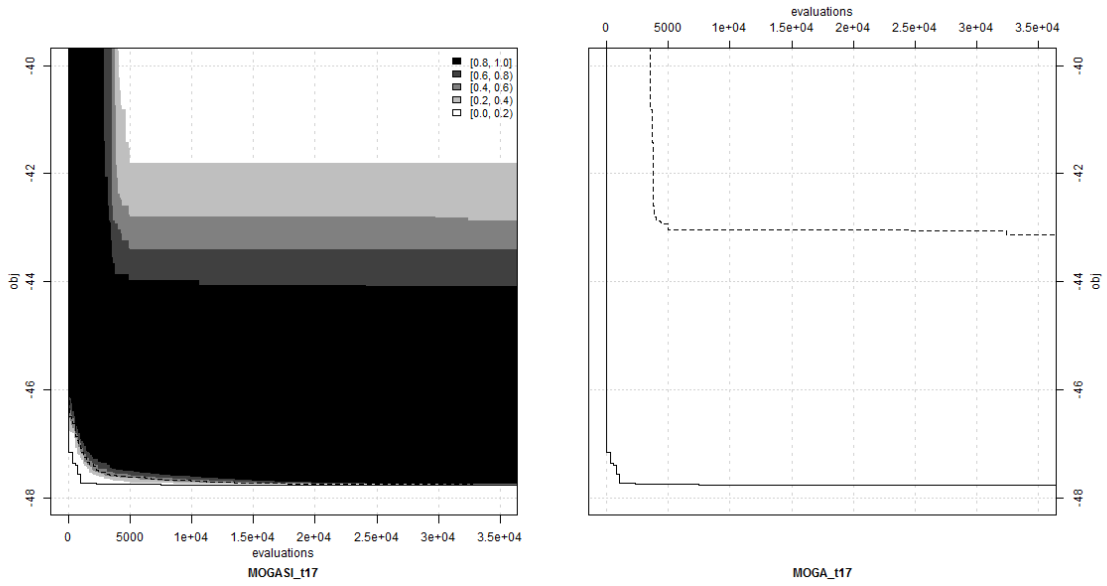


Figure 3.22: T17 problem: MOGASI vs MOGA-II

Unlike in the previous two cases, Figures 3.22–3.23 do not have entirely black areas, which indicates that the search space is indeed more suitable for general purpose GAs. It transpires from Figure 3.22 in particular that the line representing the worst behavior is slightly outside the shown area. A zoom-in action, however, resulted in an impossibility to view in detail the shadings of the EAF comparison plot in the convergence areas. It also concealed the fact that despite the MOGASI's speed in reaching the optimal zone, this algorithm had difficulties in refining it and increasing its accuracy in all 50 runs. This behavior is not entirely unexpected

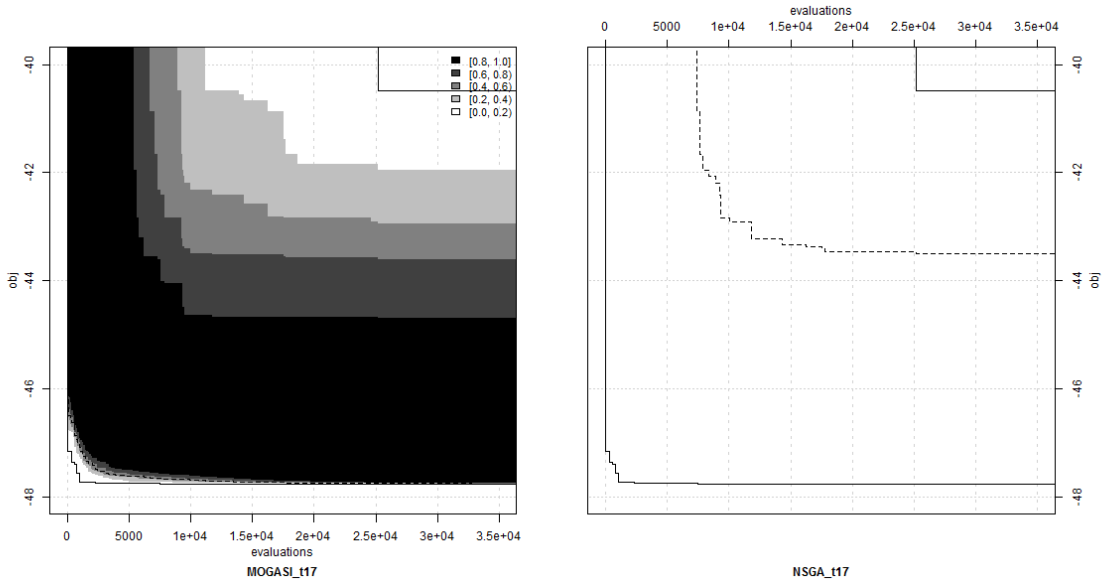


Figure 3.23: T17 problem: MOGASI vs NSGA-II

from this class of heuristics. In fact, it could be a indication of its limited ability to reach high precision levels in problems with many equality constraints. As regards the comparison with the competitors, the situation is identical to the previous two cases.

3.10.2.7 Test problem: t26

A maximization single-objective problem with four variables is used as the final test, with one equality and two inequality constraints. The exact mathematical formulation is shown in Eq. 3.17.

$$
\begin{aligned}
& f(x) = max \;\; x_0 \cdot \sin(x_1) + x_2^2 - 12.7 \cdot x_3 \\
& g_1(x) = 2 \cdot x_0 + x_1 - 3.5 \cdot x_3 = 1 \\
& g_2(x) = x_1 + x_2 \leq 3 \\
& g_3(x) = x_0 - 2 \cdot x_1 \leq 0 \\
& x_0 \in [1,8] \\
& x_1 \in [0,8] \\
& x_2 \in [-2.1,3.1] \\
& x_3 \in [0,4.4]
\end{aligned}
\tag{3.17}
$$

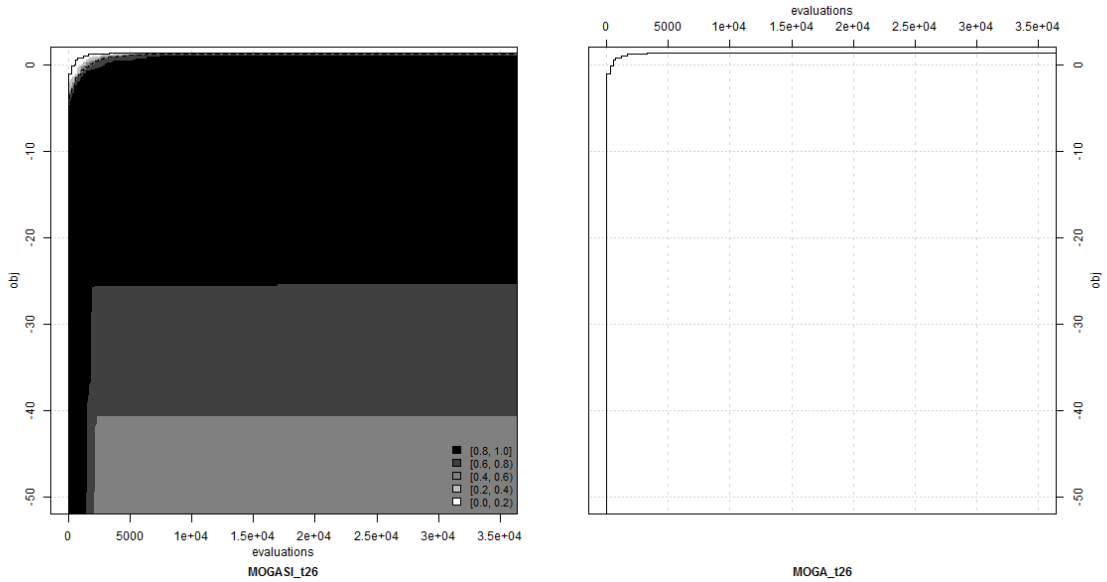


Figure 3.24: T26 problem: MOGASI vs MOGA-II

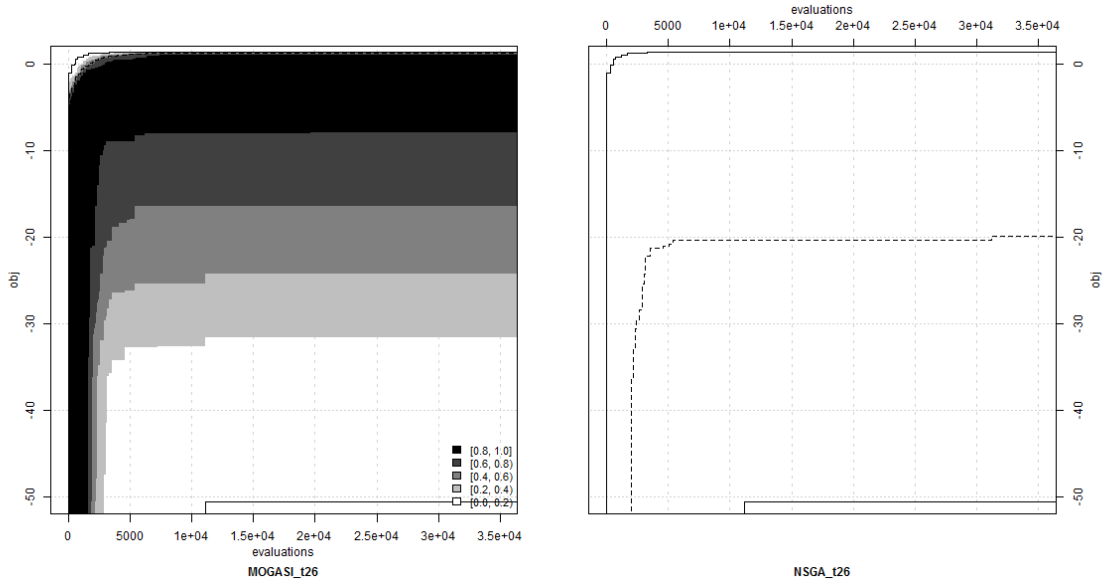


Figure 3.25: T26 problem: MOGASI vs NSGA-II

The charts in Figures 3.24–3.25 confirm the previously described inadequacy of generic purpose GAs in the presence of equalities, which irreversibly undermine their average performance. This test, however, shows a trend change between the two competitors already visible before. More specifically, different objective landscapes result in a better performance of either MOGA-II or NSGA-II, as is the case. This is a clear evidence of their different

computation capabilities and different problems they are designed to handle in spite of the fact that they are classified in literature in the same category of global optimization GAs. Again, a detailed comparison of the competitors with MOGASI is not necessary as it presents the same characteristics as in the previous test problems.

3.10.3 Multi-Objective Optimization: Deb Benchmarks

Seven test problems are presented in this Section as in the previous one. These benchmarks have been chosen because they are universally accepted for testing of general-purpose GAs. Their mathematical formulations can be found in Deb et al. (2002). Here as well for each problem instance each algorithm was run fifty times to obtain robust results without changing any input parameter other than the seed of the random sequences, which were used to generate the initial populations of each run. The mean IGD value obtained after fifty iterations is used to rank and compare the performance of the algorithms. Even in this case, the parameters of all algorithms are set to their default values for all problem instances (see Tables 3.5–3.6) to avoid possible performance gaps that might occur due to parameter tuning. Again, it was decided to use a reasonable set of parameter values for MOGASI, as shown in Table 3.7. Again, no effort was made to find the best parameter combination.

Table 3.5: Multi-Objective Problem: NSGA-II parameters

Description	Value
Population Size	100
Num.Generations	200
Max Evaluations	20,000
Crossover prob.	0.9
Real Mutation prob.	1.0/nvar
Distribution Index for operators	20.0

Table 3.6: Multi-Objective Problem: MOGA-II parameters

Description	Value
Population Size	100
Num.Generations	200
Max Evaluations	20,000
Directional Crossover prob.	0.5
Mutation prob.	0.1
DNA Mutation Ratio	0.05
Selection prob.	0.05

Table 3.7: Multi-Objective Problem: MOGASI parameters

Description	Value
Population Size	100
Num.Generations	200
Max Evaluations	20,000
Operators prob.	uniform
Auto scaled op.prob.	true
Q ind. distribution	0.1
Inizialization	random
Replacement prob.	0.15
Ind. prob. distribution	cumulative

All optimization problems analyzed in this section have minimization objectives. The focus is on the comparison of the non-dominated fronts found by the algorithms. As already discussed in the SO case, a time-based comparison of algorithm performance is usually interesting in case of different algorithm structures. However, all three algorithms share the same nature and require no computationally intensive operations, so the predicted computational cost is negligible and is almost the same also for the MO case. Moreover, it is important to stress again that a pure time-based comparison would not have been entirely fair due to the fact that MOGASI has been implemented in a batch standalone module, whereas the two competitors are part of a commercial optimization platform with many options and graphical support that slow down their execution. Hence, MOGASI would easily win in a time-comparison, which would be misleading, and for this reason no results related to execution times are presented.

3.10.3.1 SCH

SCH is a simple unconstrained problem which involves a single variable with a very large range of variation and two non-linear objectives, as shown in Eq. 3.18. This problem presents a simple convex front.

$$\begin{aligned} f_1(x) &= x^2 \\ f_2(x) &= (x-2)^2 \\ x &\in [-10^3, 10^3] \end{aligned} \tag{3.18}$$

Table 3.8 contains the average IGD, the calculated Variance and Standard Deviation obtained after fifty iterations of each algorithm. MOGA-II registered a high stability level

over the fifty iterations in terms of variance and standard deviation whose values are extremely low. Its IGD value is perfectly comparable with the NSGA-II, but both of them are outperformed by MOGASI by two orders of magnitude.

Table 3.8: Final IGD values on the SCH problem

	MOGA-II	NSGA-II	MOGASI
IGD	1.67E-02	1.65E-02	4.84E-04
Var.	1.08E-34	4.67E-06	1.03E-07
Std. Dev.	1.04E-17	2.16E-03	3.20E-04

Figure 3.26 shows the behavior of the algorithms in form of IGD values for different steps of the optimization expressed against the number of evaluations. These two visualization tools are useful to differentiate in the former the quality of the general results collected by the algorithms, and to further highlight in the latter the characteristics of the optimization trend, such as the convergence rate. Indeed, already after 2,000 evaluations MOGASI surpassed the results reached by the competitors that would improve only very slightly by the end of the run.

Even though the absolute value of the IGD metric is rather low for all algorithms, MOGASI was able to better refine the SCH front within 20,000 evaluations, as shown in Figure 3.26. The difference between the average performance in terms of the order of magnitude is not influenced by the algorithm variance, nor by the standard deviation.

3.10.3.2 KUR

The second unconstrained multi-objective problem presented in this Section, called KUR, has two non-linear objective functions and three variables (see Eq. 3.19). In this case the Pareto front is non-convex.

$$
\begin{aligned}
& f_1(x) = \sum_{i=1}^{n-1} \left(-10exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2} \right) \right) \\
& f_2(x) = \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5sinx_i^3 \right) \\
& x_i \in [-5,5] \\
& i = 1,2,3
\end{aligned}
\tag{3.19}
$$

Table 3.9 shows that all three algorithms have achieved the same IGD orders of magnitude with similar robustness levels, determined by comparable values of variance and standard

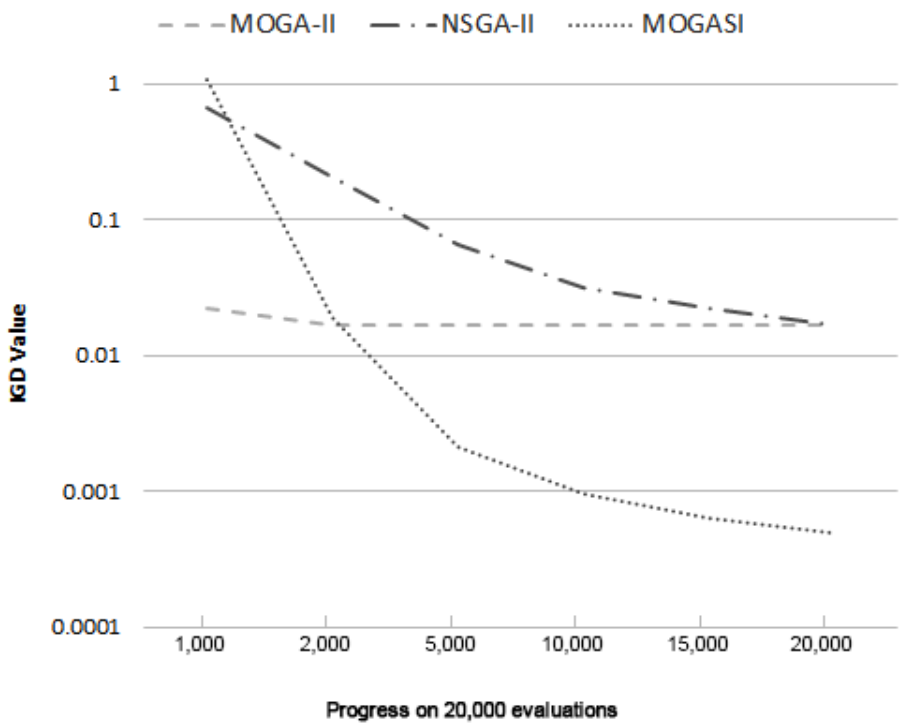


Figure 3.26: SCH Problem: Performance Comparison

deviation. MOGA-II has considerable difficulties in refining the Pareto front as shown by the relative stagnation of the IGD value. This can be observed over 10,000 evaluations in Figure 3.27.

Table 3.9: Final IGD values on the KUR problem

	MOGA-II	NSGA-II	MOGASI
IGD	4.22E-02	1.27E-02	1.51E-02
Var.	7.32E-05	1.08E-06	1.35E-05
Std. Dev.	8.56E-03	1.04E-03	3.67E-03

Another interesting behavior can be observed in Figure 3.27. In particular, MOGASI and NSGA-II present perfectly the same trend at 20,000 evaluations and this similarity is probably due to the selection mechanism which MOGASI inherits from NSGA-II as described in Section 3.7. At the same time it can be presumed that the resulting gap between their absolute IGD values is due to the different genetic operators that those two algorithms use internally. Nevertheless, the collected results are entirely comparable, which translates as a success of MOGASI. The reason is that MOGASI contains specific additional mechanisms that are normally activated in the presence of certain problem characteristics, such as

the linear constraints, and improve its convergence. This is a generic problem so these characteristics are not present, but MOGASI is still able to reach the same result quality levels as the two competitors.

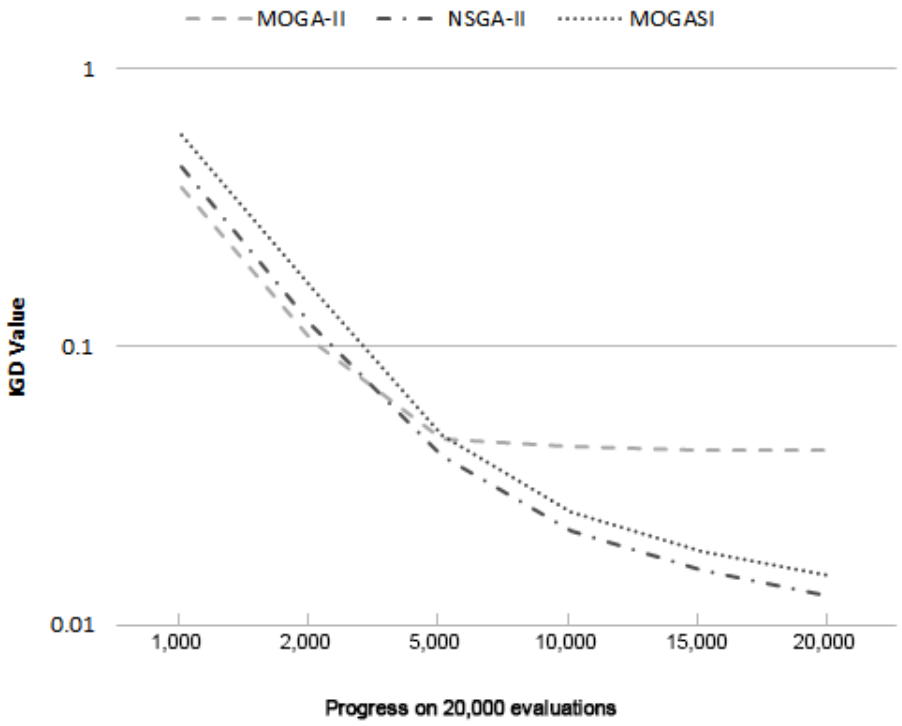


Figure 3.27: KUR Problem: Performance Comparison

3.10.3.3 DEB

The DEB problem is a well-known constrained multi-objective problem, widely used in GA literature. As presented in Eq. 3.20 it has two mixed objectives (one linear and one non-linear) and two variables, and it is constrained by two linear inequalities.

$$
\begin{aligned}
& f_1(x) = x_1 \\
& f_2(x) = (1 + x_2)/x_1 \\
& g_1(x) = x_2 + 9x_1 \geq 6 \\
& g_2(x) = -x_2 + 9x_1 \leq 1 \\
& x_1 \in [0.1, 1.0] \\
& x_2 \in [0, 5]
\end{aligned}
\tag{3.20}
$$

In this case, as in the previous one, very low IGD values can be observed for all three algorithms, leading to the conclusion that each of them achieves a good coverage of the real Pareto front. The only relevant difference is given by the comparison of the calculated variances shown in Table 3.10, where a worse performance of MOGA-II is visible from the order of magnitude of the IGD value compared to the others.

Table 3.10: Final IGD values on the Deb problem

	MOGA-II	NSGA-II	MOGASI
IGD	1.94E-02	4.54E-03	1.24E-03
Var.	1.43E-05	3.25E-08	2.96E-07
Std. Dev.	3.78E-03	1.80E-04	5.44E-04

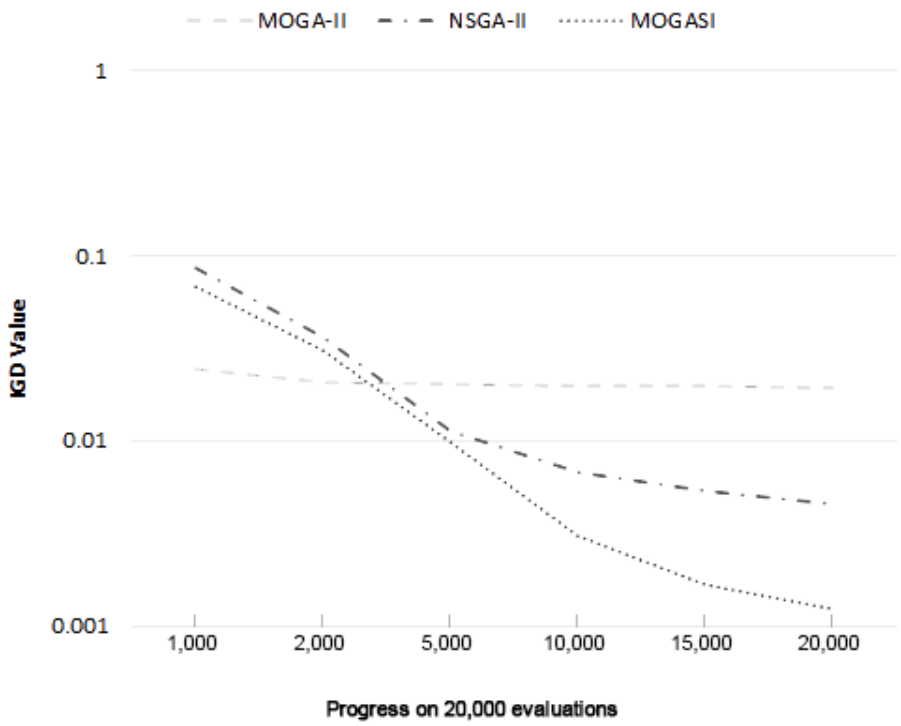


Figure 3.28: DEB Problem: Performance Comparison

An analysis of Figure 3.28 shows that MOGA-II had the best average IGD values up to approximately 2,000 evaluations, but from that point on it stagnates and improves very little, unlike the competitors. The figure also shows a continued decrease of the MOGASI IGD values throughout the evaluation sequence. It can be presumed that MOGASI earned a clear advantage from the arithmetic crossover operator (Michalewicz and Janikow, 1996) for the

systematic search space exploration on a convex set such as this one, which could be the reason why its convergence is longer and deeper than the competitors' before entering the stagnation phase.

3.10.3.4 SRN

The SRN test function has been chosen because it has both a linear and a non-linear constraint (see Eq. 3.21 for its mathematical formulation). It is a non-linear multi-objective problem with two objectives and only two variables.

$$
\begin{aligned}
& f_1(x) = (x_1 - 2)^2 + (x_2 - 1)^2 + 2 \\
& f_2(x) = 9x_1 - (x_2 - 1)^2 \\
& g_1(x) = x_1^2 + x_2^2 \leq 225 \\
& g_2(x) = x_1 - 3x_2 \leq -10 \\
& x_i \in [-20, 20] \\
& i = 1, 2
\end{aligned}
\tag{3.21}
$$

It is relevant for this work to verify whether in a simple problem, such as this one, the difference between the mean performance of MOGASI and NSGA-II is not so significant but it is nonetheless present in favor of MOGASI. The difference between MOGASI and MOGA-II, as shown in Table 3.11, is much higher again in favor of MOGASI.

Table 3.11: Final IGD values on the SRN problem

	MOGA-II	NSGA-II	MOGASI
IGD	1.79E-01	7.98E-02	4.48E-02
Var.	3.69E-03	1.15E-05	3.52E-04
Std. Dev.	6.07E-02	3.39E-03	1.88E-02

Figure 3.29 shows the IGD evolution after 20,000 evaluations for the three algorithms. The capability of MOGASI to approximate the front is as good as expected. Furthermore, it shows the IGD progress over the total evaluations: MOGASI reaches within half of the total available evaluations a precision comparable to the final NSGA-II IGD value.

3.10.3.5 TNK

TNK is a non-linear constrained problem with two very simple objectives and two variables. The exact formulation is shown in Eq. 3.22. This problem enabled the first verification of

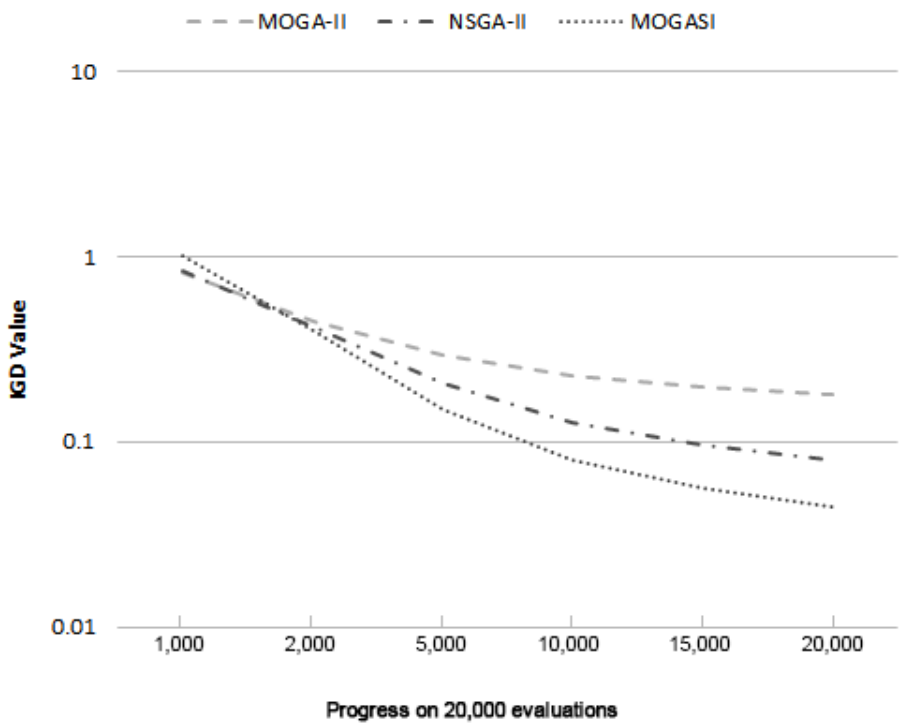


Figure 3.29: SRN Problem: Performance Comparison

whether MOGASI is able to effectively explore the feasible region defined by non-linear constraints for a multi-objective space, or not.

$$
\begin{aligned}
& f_1(x) = x_1 \\
& f_2(x) = x_2 \\
& g_1(x) = -x_1^2 - x_2^2 + 1 + 0.1\cos(16\arctan(x/y)) \leq 0 \\
& g_2(x) = (x - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5 \\
& x_i \in [0, \pi] \\
& i = 1, 2
\end{aligned} \tag{3.22}
$$

Table 3.12: Final IGD values on the TNK problem

	MOGA-II	NSGA-II	MOGASI
IGD	2.97E-03	3.09E-03	1.55E-03
Var.	1.52E-07	4.43E-08	3.28E-08
Std. Dev.	3.90E-04	2.11E-04	1.81E-04

As in the previous cases, it can be observed from the line chart in Figure 3.30 that MOGASI achieved the best performance. The behavior of MOGA-II and NSGA-II is perfectly comparable. On the other hand, MOGASI's line shows a deep descent up to 5,000 evaluations where it changes convexity. Table 3.12 shows a series of equally good average IGD values but with no relevant variance nor standard deviation. TNK test results play in favor of MOGASI because this kind of problem does not belong to the category of problems MOGASI has been designed for. This was in fact a test of the general algorithm behavior on problems with no specific data structures. For this reason this case can be considered as the worst case scenario for MOGASI.

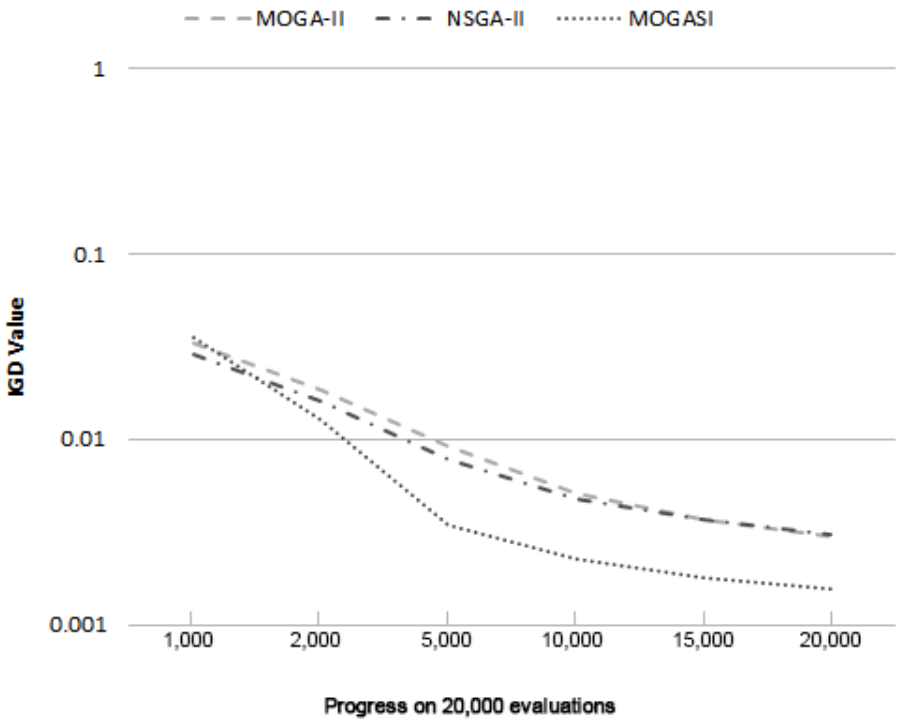


Figure 3.30: TNK Problem: Performance Comparison

3.10.3.6 OSY

The constrained optimization problem OSY (Osyczka and Kundu, 1995) has six variables, two objectives and six constraints. The constrained space is composed of four linear inequality constraints and two non-linear constraints, as shown in Eq. 3.23. However, the variables are not evenly divided among constraints. Two variables are used by the linear constraints, whereas the remaining variables are used by the non-linear constraints. This can lead to

both advantages and disadvantages for an algorithm with specific behavior for different constraint types, such as the one implemented in MOGASI.

$$
\begin{aligned}
& f_1(x) = (25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2), \\
& f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2, \\
& g_1(x) = x_1 + x_2 - 2 \geq 0, \\
& g_2(x) = 6 - x_1 - x_2 \geq 0, \\
& g_3(x) = 2 - x_2 + x_1 \geq 0, \\
& g_4(x) = 2 - x_1 + 3x_2 \geq 0, \\
& g_5(x) = 4 - (x_3-3)^2 - x_4 \geq 0, \\
& g_6(x) = (x_5-3)^2 + x_6 - 4 \geq 0, \\
& x_1, x_2, x_6 \in [0,10] \\
& x_3, x_5 \in [1,5] \\
& x_4 \in [0,6]
\end{aligned}
\tag{3.23}
$$

The original formulation is modified to obtain a more balanced test case. The Pareto front of this problem has five segments which can be found by choosing $x_4 = x_6 = 0$ and one of the combinations exemplified in Eq. 3.24.

$$
\begin{aligned}
& x_1 = 5, \quad x_2 = 1, \quad x_3 \in [1,5], \quad x_5 = 5, \\
& x_1 = 5, \quad x_2 = 1, \quad x_3 \in [1,5], \quad x_5 = 1, \\
& x_1 \in [4.056,5], \quad x_2 = (x_1-2)/3, \quad x_3 = 1, \quad x_5 = 1, \\
& x_1 = 0, \quad x_2 = 2, \quad x_3 \in [1,3.732], \quad x_5 = 1, \\
& x_1 \in [0,1], \quad x_2 = 2 - x_1, \quad x_3 = 1, \quad x_5 = 1.
\end{aligned}
\tag{3.24}
$$

Moreover, these combinations touch the variable bounds in many parts and it is possible to shift from a Pareto point to another one by simply changing the value of a single variable. To avoid these unwanted phenomena while maintaining a smooth mathematical structure, a rotated and translated set of input variables $\mathbf{y}$ such that $A\mathbf{y} + (1,1,1,1,1,1) = \mathbf{x}$ is adopted. The matrix A applies a $\pi/6$ rotation to the first two variables, a $\pi/4$ rotation to the third and fourth variables and a $\pi/3$ rotation to the last two variables.

$$
\mathrm{A}= \begin{pmatrix}
\cos(\pi/6) & -\sin(\pi/6) & 0 & 0 & 0 & 0 \\
\sin(\pi/6) & \cos(\pi/6) & 0 & 0 & 0 & 0 \\
0 & 0 & \cos(\pi/4) & -\sin(\pi/4) & 0 & 0 \\
0 & 0 & \sin(\pi/4) & \cos(\pi/4) & 0 & 0 \\
0 & 0 & 0 & 0 & \cos(\pi/3) & -\sin(\pi/3) \\
0 & 0 & 0 & 0 & \sin(\pi/3) & \cos(\pi/3)
\end{pmatrix}
$$

As a result of the rotation it can be observed that the problem is much more difficult to solve: fixing the target to the usual maximum evaluation number (i.e. 20,000) is insufficient to robustly reach the true front. Therefore a greater average values of the IGD coefficient can be expected for all algorithms with respect to the previous test problem.

Table 3.13: Final IGD values on the OSY problem

	MOGA-II	NSGA-II	MOGASI
IGD	6.90E+00	9.07E+00	8.07E+00
Var.	1.13E+02	2.10E+02	4.67E+00
Std. Dev.	1.06E+01	1.45E+01	2.16E+00

Data contained in Table 3.13 show that initial expectations were justified: none of the algorithms scored a good average IGD. The absolute IGD values are more distant one from another than in the previous tests but the results have the same order of magnitude and are sufficiently comparable. Figure 3.31 shows the optimization trends on 20,000 evaluations. MOGASI has a lower improvement rate but in very few evaluations it reaches the stagnation zone just like its competitors.

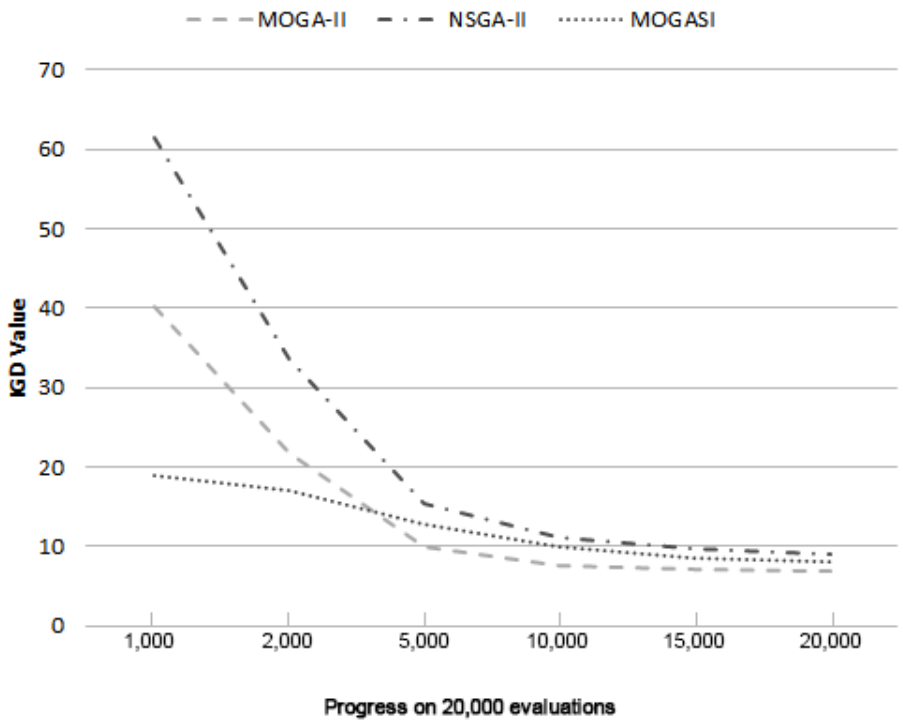


Figure 3.31: OSY Problem: Performance Comparison

MOGASI has low IGD values from the beginning owing to the use of the double population, specifically the search population (the one satisfying all linear constraints) which gives this algorithms a clear advantage. Indeed, the variable space is reduced and better explored. However, MOGASI loses this initial advantage in the next phase when it has to refine the non-dominated front. Here the difficulties are linked to the variables subject to the non-linear constraint.

The truly positive result that emerged from this test is the robustness of the average MOGASI behavior in the rotated OSY. Table 3.13 also shows a huge gap between the variance and the standard deviation values of the three algorithms. The reason behind this phenomenon could be that during the fifty iterations the classic GAs often get stuck in the local minima far from the real front. However, there is no significant indication of clear convergence advantages of one algorithm over another.

3.10.3.7 WATER

Since the performance of most multi-objective methods severely deteriorates with an increase in the number of objectives, a further distinction is made when referring to problems with four or more objectives (Khare et al., 2003). Multi-objective problems with more than three objectives are often referred to as many-objective problems (Ishibuchi et al., 2008). MOGASI is also tested on a many-objective problem taken from the library of constrained test problems used in Deb et al. (2002). WATER has five objectives controlled by three variables subject to seven non-linear constraints (see the mathematical formulation in Eq. 3.25).

As in the TNK test case, problems of this kind do not belong to the category of problems MOGASI has been designed for, so it cannot benefit from any specific strategy internally implemented to handle these problems. However, the algorithm robustness was confirmed by analyzing the IGD evolution during the optimization shown in Figure 3.32. The performance of MOGASI was perfectly comparable with the competitors, even for the values of computed variation and standard deviation, as summarized in Table 3.14.

Table 3.14: Final IGD values on the WATER problem

	MOGA-II	NSGA-II	MOGASI
IGD	2.04E-02	2.59E-02	1.84E-02
Var.	2.22E-06	1.64E-06	5.47E-06
Std. Dev.	1.49E-03	1.28E-03	2.34E-03

$$
\begin{aligned}
& f_1(x) = 106780.37(x_2 + x_3) + 61704.67 \\
& f_2(x) = 3000x_1 \\
& f_3(x) = (305700)2289x_2/(0.06 \cdot 2289)^{0.65} \\
& f_4(x) = (250)2289exp{-}39.75x_2 + 9.9x_3 + 2.74) \\
& f_5(x) = 25(1.39/(x_1x_2) + 4940x_3 - 80) \\
& g_1(x) = 0.00139/(x_1x_2) + 4.94x_3 - 0.08 \leq 1 \\
& g_2(x) = 0.000306/(x_1x_2) + 1.082x_3 - 0.0986 \leq 1 \\
& g_3(x) = (12307)/(x_1x_2) + 49408.24x_3 + 4051.02 \leq 50000 \\
& g_4(x) = (2098)/(x_1x_2) + 8046.33x_3 - 696.71 \leq 16000 \\
& g_5(x) = (2138)/(x_1x_2) + 7883.39x_3 - 705.04 \leq 10000 \\
& g_6(x) = 0.417x_1x_2 + 1721.26x_3 - 136.54 \leq 2000 \\
& g_7(x) = 0.164/(x_1x_2) + 631.13x_3 - 54.48 \leq 550 \\
& x_1 \in [0.01, 0.45] \\
& x_2 \in [0.01, 0.10] \\
& x_3 \in [0.01, 0.10]
\end{aligned}
\tag{3.25}
$$

The expectations have been met in the many-objective case, as in Figure 3.32, which shows the algorithm behavior during the optimization. A quick check of the absolute average IGD values from Table 3.14 confirm the better algorithm performance compared to the competitors even if in the same order of magnitude.

3.10.4 Benchmarks Outcome

As explained in the introduction of Section 3.10, the benchmarks have been chosen to cover a wide selection of problem characteristics and hence to enable a correct validation of the proposed algorithm. For this reason both constrained and unconstrained problems are included, and the former contain all types of handled constraints in different combinations. In particular, constraints are of both linear and non-linear type, with a further distinction between linear equalities and inequalities. In Section 3.10.2 and 3.10.3, two separate sets were run to test the algorithm respectively on Single- (SO) and Multi-Objective (MO) instances. Many-Objective (MaO) problems were not considered in the scope of this thesis, but a single benchmark in this context was nevertheless performed due to the considerable interest that this class of problems has recently received in the scientific community.

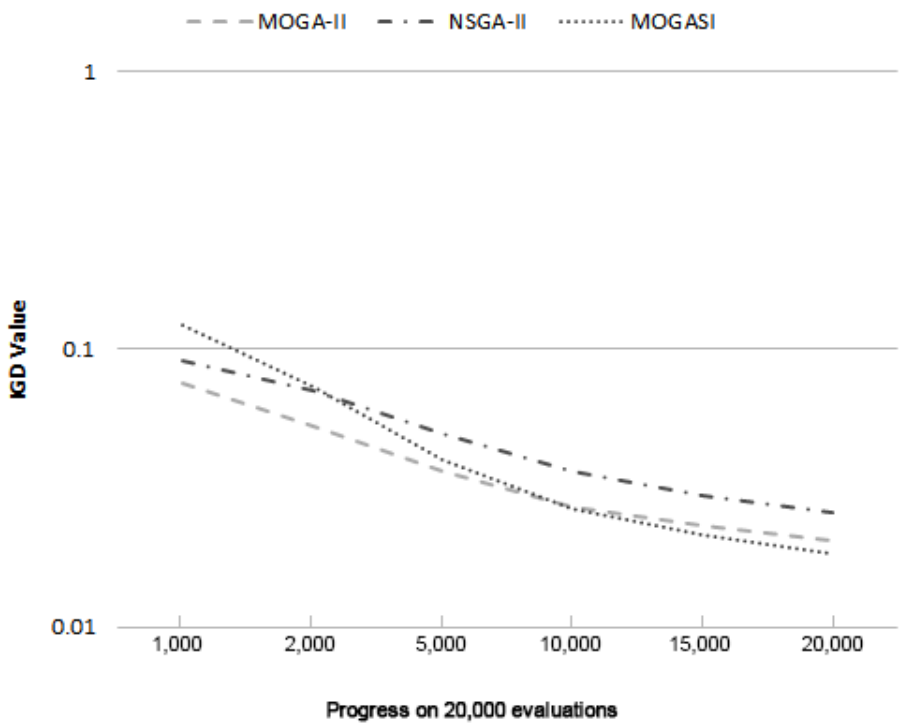


Figure 3.32: WATER Problem: Performance Comparison

A summary of the main MOGASI features to be validated by the benchmark tests is presented:

- As detailed in Section 3.4, the solutions are coded in double-precision floating-point format in a space with real values R^q. This is a huge difference from the MOGA-II and NSGA-II coding (for details refer to the respective papers: Poloni and Pediroda (1997) and Deb et al. (2002)).

- The double population mechanism described in Section 3.5 can have both advantages and drawbacks. It was devised to improve the algorithm performance in case of considerable differences between the search and the reference populations (e.g. the existence of many linear constraints and in particular linear equality constraints). On the other hand, this mechanism has a certain computational cost and requires Black Box evaluations that can be wasted in case of problems where such differences are not that pronounced;

- No less than eight GA operators are used in MOGASI, as explained in Section 3.4, but they are intended for and were taken from very different scenarios. Since no

fine-tuning was performed, the probability of their application is uniform;

- MOGASI has no dedicated Parent Selection mechanism. Instead, this task is accomplished by a simple probability-based extraction of new parents from the existing population, as explained in Section 3.9.

It is important to highlight that no data structure modules have relevance for benchmarking purposes and they are not discussed in this thesis, as already explained in Section 3.2. For further details in this regard refer to Costanzo et al. (2014).

The main target of Section 3.10 was to verify the MOGASI capability to reach a comparable performance level on generic problems as the competing algorithms and to surpass them on certain problems. The results show that the double population mechanism coupled with the GA operators leads to a remarkable performance gap between MOGASI and the competing algorithms on six problems. These problems are all SO and subject to linear equality constraints, hence the differences between the search and reference space is very pronounced. A doubt arises, that is whether these result differences can be caused by the SO nature instead of by the presence of equalities. Three observations can be made: firstly, both competitors can be used to solve SO problems without any modification similarly to the proposed algorithm; secondly, MOGASI reached at least comparable results also on all MO problems; finally, the MOGASI results on t01 and t02 problems are respectively equally comparable and much worse than the competitors. This last observation in particular and the absence of linear equality constraints dispel the expressed doubt despite the SO nature of t02. One last question can arise about the double population mechanism, that is whether it burdens the algorithm to the extent of hindering convergence in presence of many local minima, such as in the t02 problem. The other results, however, reveal this as an isolated case, probably due to the search space complexity. It may be deduced that the use of the double population as elite set when its main purpose is not accomplished (as explained in Section 3.5) does not have a negative impact on the MOGASI performance. This deduction is confirmed by the fact that the MOGASI disadvantage with respect to the other two GAs does not reappear in any of the six MO tests, nor in the MaO problem.

The remaining six MO benchmarks reveal that in two of them MOGASI reaches comparable results with respect to the competitors. In other three the IGD value decreases quickly to the competitor levels, but then MOGASI is able to better refine the Pareto front and lowers it further. The sixth, unconstrained benchmark test SCH (described in 3.10.3.1), is an MO outlier in which MOGASI reaches an IGD value of an order of magnitude better than the

competing GAs. Satisfying results have also been obtained in the MaO problem, although at levels comparable to the GAs.

The goal of the algorithm validation presented in Chapter 3 has been reached. This indicates that the MOGASI modular structure is qualitatively comparable with leading competitors in spite of the mechanism differences highlighted at the beginning of this chapter. In fact, the most invasive mechanisms (i.e. the selection and the double population) play in favor of MOGASI when dealing with target problems and do not otherwise burden its search capabilities.

CHAPTER 4

BI-LEVEL PRICING PROBLEMS

Many real-world problems involve a hierarchical relationship between two decision levels, for instance in areas like management (facility location, environmental regulation, credit allocation, energy policy, hazardous material regulation), economic planning (social and agricultural policies, electric power pricing, oil production) or engineering (optimal design, structures and shape). In these models the choice of a decision-maker always leads to some reaction within a particular market or social entity. Such problems in mathematical programming can be formulated as Bi-level Programming Problems (Colson et al., 2005), or BPPs.

4.1 Bi-level Programming: An Overview

A BPP is a hierarchical optimization problem in which part of the constraints translates the fact that some of the variables constitute an optimal solution to a second optimization problem (Labbé and Violin, 2013). There are many examples of real-world problems that would fit the BP framework, but the number of actual implementations is limited (Dempe, 2002). The main reason is the absence of dedicated algorithms, especially those capable of solving large-scale problems of this kind. Many different approaches have been proposed over time in literature, but none of them is universal to this problem class and thus cannot guarantee convergence, performance or optimality at the same time for all BP problems. The first works on BPPs appeared in 1970's, but due to the nonexistence of sufficiently powerful computers this field remained quite unexplored until mid-eighties of the 20^{th} century. More

precisely, these problems were introduced by Bracken and McGill (1973) as mathematical programs with optimization problems in the constraints, whereas the terms bi-level and multi-level were later introduced by Candler and Norton (1977).

Bi-level models represent a sequential game between two decision levels, commonly referred to as *Leader* (upper level) and *Follower* (lower level), where the leader has complete knowledge of the follower's strategy, and is therefore able to anticipate it, while the opposite does not hold. The follower can therefore only react to the leader's action. For given values of the first level decision variables, the second level problem may have multiple optimal solutions. In this case, different approaches can be proposed depending on the follower's behavior. A cooperative behavior leads to an optimistic solution, such that when there are multiple solutions the leader assumes that the follower's choice is always the one most favorable to him/her. On the other hand, an aggressive or non-cooperative behavior leads to a pessimistic solution, where a follower, faced with multiple optimal solutions, selects the one least favorable to the leader, who in turn must protect himself against such reaction (Colson et al., 2005). A more complete discussion on these issues can be found in Dempe (2002).

By denoting x and y respectively the leader's and follower's decision variables vectors, BP Problems can be described mathematically according to the optimistic approach as:

$$\min_{x,y} F(x,y) \tag{4.1}$$

$$\text{s.t. } G(x,y) \leq 0 \tag{4.2}$$

$$y \in \arg\min_{y} f(x,y) \tag{4.3}$$

$$s.t. g(x,y) \leq 0 \tag{4.4}$$

Note that in the formulation above multiple optimal solutions for the lower level problem can occur. If the solution selected among them is the most profitable for the leader, this is known as the optimistic approach, as opposed to the pessimistic approach where the leader opts for the worst case scenario as a means of protection. Both scenarios have been investigated in literature (for a detailed bibliography see Heilporn, 2008).

In general, BPPs are similar in principle to the *Stackelberg game* (see Stackelberg, 1952) and their solution is referred to as *Stackelberg equilibrium*. This is in contrast with the *Nash's equilibrium* (introduced in Nash, 1950) where both players have complete knowledge of each other's strategy. In order to underline the differences a simple example with two players, called 1 and 2, is shown in Table 4.1, taken from Violin (2014). Both players have

two strategies, denoted by A and B, and they want to maximize their own gain. Gains for all players and strategies are reported in the table. If Player 1 is the leader and Player 2 is the follower, the Stackelberg solution consists of strategy B for both players, leading to a gain of 3 for Player 1 and of 1 for Player 2. On the contrary, if Player 2 is the leader and Player 1 the follower, the Stackelberg solution consists of strategy A for both players, leading to a gain of 2 for both of them.

		Player 2	
		Strategy A	**Strategy B**
Player 1	**Strategy A**	2 , 2	4 , 1
	Strategy B	1 , 0	3 , 1

Table 4.1: Stackelberg vs Nash example: payoff matrix (Violin, 2014)

As already mentioned, another well known solution concept in game theory is the Nash equilibrium (see e.g. Nash, 1950), which is appropriate for a simultaneous game (i.e. players play at the same time, without any hierarchical structure) or games that are repeated many times. Specifically, a Nash equilibrium consists in a solution in which no player has an interest in changing his/her choice of strategy, as it cannot lead to a better gain. For the game described by the payoff matrix in Table 4.1, the Nash equilibrium is unique and given by strategy A for both players, leading to a gain of 2 for both of them. It is therefore interesting to underline the conceptual difference between a Stackelberg and a Nash equilibrium: they correspond to different assumptions in the game rules, as the former is sequential with a precise hierarchical structure. They generally lead to different solutions. One can also point out that, under mild conditions, there is always a Stackelberg solution (Stackelberg, 1952). On the contrary, a Nash equilibrium may not exist, as it could occur that, for all possible pairs of strategies, at least one player would always have an interest in changing their choice. For the scope of this work only the cooperative behavior is considered.

Just like single-level optimization problems, Bi-level Optimization Problems (BOPs) can have the real-value encoded solutions (continuous BOPs) and solutions encoded with discrete variables. The latter are known as combinatorial BOPs. Problems including both continuous and discrete variables are known as mixed BOPs. From a computational point of view, BOPs are intrinsically difficult and demanding, so most research has focused on the simplest cases, i.e. those with properties which were the easiest to handle, such as linear, quadratic or convex objective and/or constraint functions and continuous variables (particularly in the follower level). The reason was not only the development of linear programming and the relative ease to solve exactly the follower level problem, but also the

extensive availability of practical real-world cases that could be formulated as such. Other approaches included new ideas to the more complex BP problems (for example, those with non-linear variables), such as re-formulation and operative research methods, which mask the bi-level aspect of the problem. One of the first approaches was to replace the lower level by its optimality conditions, the Karush-Kuhn-Tucker (multiple optimal solutions) conditions. In this way the BOP is transformed into a single-level optimization problem with complementary constraints and the KKT conditions are used to determine if a solution is the optimum for a constrained optimization problem. The works of Colson et al. (2005); Dempe (2002); Dempe and Dutta (2012); Dewez et al. (2008) propose different alternative ways for reducing BPPs to a single level, which can be solved with exact linear solvers. Even these approaches, however, are case-specific. The studies on BP expanded over the years to more complicated subclasses, such as BP problems with discrete variables (Vicente and Calamai, 1994), nonlinear BP problems (Colson et al., 2005) and mixed-integer BP problems (Kalashnikov et al., 2015). This last category in particular received significant attention due to the possibility to formulate a number of problems in the fields of energy and networks as mixed-integer bi-level problems.

In general, BOPs, even the simple class of a BP problem where the objective functions and the constraints are linear and continuous, have been shown to be *NP-hard* by Jeroslow (1985). Vicente and Calamai (1994) show that even merely checking strict local optimality and local optimality in linear BP problems are NP-hard problems. Furthermore, Hansen et al. (1992) prove the strong NP-hardness of the problem. Considering this complexity, the research of BPs followed two main approaches: continuous and combinatorial. The former studies the optimality conditions and focuses on algorithms which can find at most a local solution. The combinatorial approach has a global perspective and focuses on algorithms which can find a trade-off global optimal solution, but are limited to problems with certain characteristics. Due to the difficulty of BPs, solution methods and algorithms generally focus on particular cases where functions have convenient properties, such as linearity and convexity, so their structure favors the development of efficient solution methods.

4.2 Metaheuristics for Bi-level Optimization Problems

The term metaheuristic has been introduced by Glover (1986). Metaheuristic search methods can be defined as upper-level general methodologies that can be used as guiding strategies in designing underlying heuristics to solve specific optimization problems. They have received increasing popularity in the last 20 years. Their uses in many applications show their

efficiency and effectiveness to solve large and complex problems.

In recent years, the application of metaheuristic techniques to solve multi-level and particularly BOPs has become an active research area. A wide variety of metaheuristics have been proposed in literature to solve such hierarchical optimization problems. A BOP can be seen as a multi-level problem with two levels. Compared to single-level optimization, the difficulty in solving BOPs lies in the following general facts:

- **Evaluation of the solutions at the upper-level problem:** It is not easy to evaluate the upper-level objective function of a BOP. The objective function at the upper-level has no explicit formulation, since it is conditioned by the lower-level optimization problem. In other words, the upper-level decision maker cannot optimize their objective without regards to the reactions of the lower-level decision maker;

- **Complex interaction between the upper-level and the lower-level optimization problems:** the lower-level can be seen as a non-linear constraint and the whole problem is intrinsically a non-convex programming problem. Even if the objective function and the constraints of the upper-level and lower-level optimization problems of a BOP are all linear, the BOP is neither continuous everywhere nor convex for the objective function of the upper-level problem.

4.2.1 Evolutionary Algorithm for BOPs

Evolutionary Algorithms for bi-level optimization were first proposed back in 1990s. One of the first EAs for handling BOPs was proposed by Mathieu et al. (1994). This algorithm used a nested strategy, where the lower level was handled with a linear programming method, whereas the upper level was solved with a genetic algorithm (GA). Nested strategies are indeed a popular approach for handling bi-level problems because of their simplicity. Further details on these strategies are given in Section 4.3.3. For every upper level decision variable configuration, a lower level optimization task can be executed, but this can also make them computationally expensive and therefore they are not adequate for bi-level problems requiring a high computational effort in both leader and follower levels. A different nested approach was proposed in Yin (2000): here the upper level is a GA and the lower level is handled with the Frank-Wolfe algorithm, a reduced gradient method. The algorithm was successful in solving non-convex BOPs and the authors claimed it to be better than the classical methods. In both of these approaches for every upper level solution a lower level optimization task was executed that emphasizes the nested structure approaches.

Oduguwa and Roy (2002) proposed a co-evolutionary approach based on two populations (this kind of approach is discussed in more detail in Section 4.3.4). The first population handles the leader level decision variables, whereas the second population handles the follower level variables. These populations interact with one another in order to converge towards the optimal solution. Another study in this direction can be found in Legillon et al. (2012) where the authors solve a bi-level problem with linear constraints with a new presented algorithm, Cobra. It handles population-based algorithms on each level, each cooperating with the other to provide solutions for the overall problem. An experimental analysis is conducted on a bi-level distribution planning problem, where multiple manufacturing plants deliver items to depots, and where a distribution company controls several depots and distributes items from depots to retailers. The authors claim that the results reveal significant enhancements over the lower level, especially compared with hierarchical approaches.

Wang et al. (2005) proposed an evolutionary algorithm with a constraint handling scheme which was successfully used to solve a number of standard test problems. Their approach was able to find better solutions in comparison to what is reported in literature. This algorithm handles non-differentiability at the upper level objective function, but it is not clear whether it can handle it in the constraints or the lower level objective function. Later on Wang et al. (2010) proposed an algorithm able to handle non-convex lower level problem and non-differentiable upper level objective function. The algorithm was shown to perform better than the previous one.

The evolutionary algorithm proposed in Li et al. (2006) uses particle swarm optimization to solve bi-level problems, where for each particle's movement it solves the follower level optimization problem in a classic nested approach. The authors show that the approach is able to produce good results on standard test problems with a small number of variables. However, they do not report the computational cost of the nested procedure for large scale problems. A hybrid nested approach proposed in Li and Wang (2007) adds a crossover operator to the evolutionary strategy in the leader level to improve the feasibility of the individuals. It is based on the simplex method in which the best individuals generated so far are used to point to a good evolution direction. This method successfully solves a number of standard test problems. The authors report the generation number and population sizes that can be used for the leader level function evaluations, but they do not explicitly explain the total number of evaluations required at the lower level. Other example studies where authors have relied on a nested strategy include Angelo et al. (2013); Sinha et al. (2014). In both of these studies an evolutionary algorithm has been used at both levels to deal with the

bi-level nature of the problems.

Calvete et al. (2008) proposed a GA for linear bi-level problems, also claiming that their approach is capable of handling quasi-concave bi-level problems with the assumption of a linear objective function in the follower level. Researchers in the field of evolutionary algorithms have also tried to convert the BOP into a single level optimization using the Karush-Kuhn-Tucker (KKT) conditions (Wang et al., 2008). However, such conversions are possible only if the lower level is smooth and the KKT conditions can be easily produced. Further details on this issue are given in Section 4.3.1. It is noteworthy that utilization of KKT conditions restricts the algorithm's applicability to only a special class of bi-level problems. To overcome this drawback, researchers are looking into meta-modeling based approaches where the lower level optimal reaction set is approximated over generations of the evolutionary algorithm. Furthermore, many authors have been focusing their work to multi-objective bi-level optimization using evolutionary algorithms. Deb and Sinha (2010); Ruuska and Miettinen (2012); Shi and Xia (2001) are some of the studies attempting to tackle BOPs with multiple objectives in each level with evolutionary algorithms.

4.3 Metaheuristic Strategies for BOPs

As demonstrated in Section 4.2.1 there are many different versions of EAs used for solving BOPs. These strategies can be seen more generally applied to metaheuristics and categorized according to Talbi (2013). For the purpose of this work they can be classified into the following types of strategies, as shown in Figure 4.1:

- **Single-level transformation approach:** the preliminary step to the application of this class of metaheuristics is the reformulation of the BOP as a single-level optimization problem. At this point any traditional single-objective metaheuristic strategy can be applied;

- **Transformation to Multi-objective approach:** in this class the preliminary step is the transformation of the BOP as a Multi-objective Optimization Problem. In this case too any traditional multi-objective metaheuristic strategy can be used;

- **Co-evolutionary approach:** this can be seen as the most simple methodology to solve BOPs (not requiring any reformulation) in which a metaheuristic strategy is applied to tackle a single level of the problem. At this point all metaheuristics co-evolve in parallel and exchange information solving the different levels of the problem;

- **Nested sequential approach:** this class of metaheuristic strategies is generally applied when the computational time of the lower level is low. Two different metaheuristics are applied: the lower level optimization problem is solved in a nested and sequential way to evaluate the solutions generated at the upper-level of the BOP.

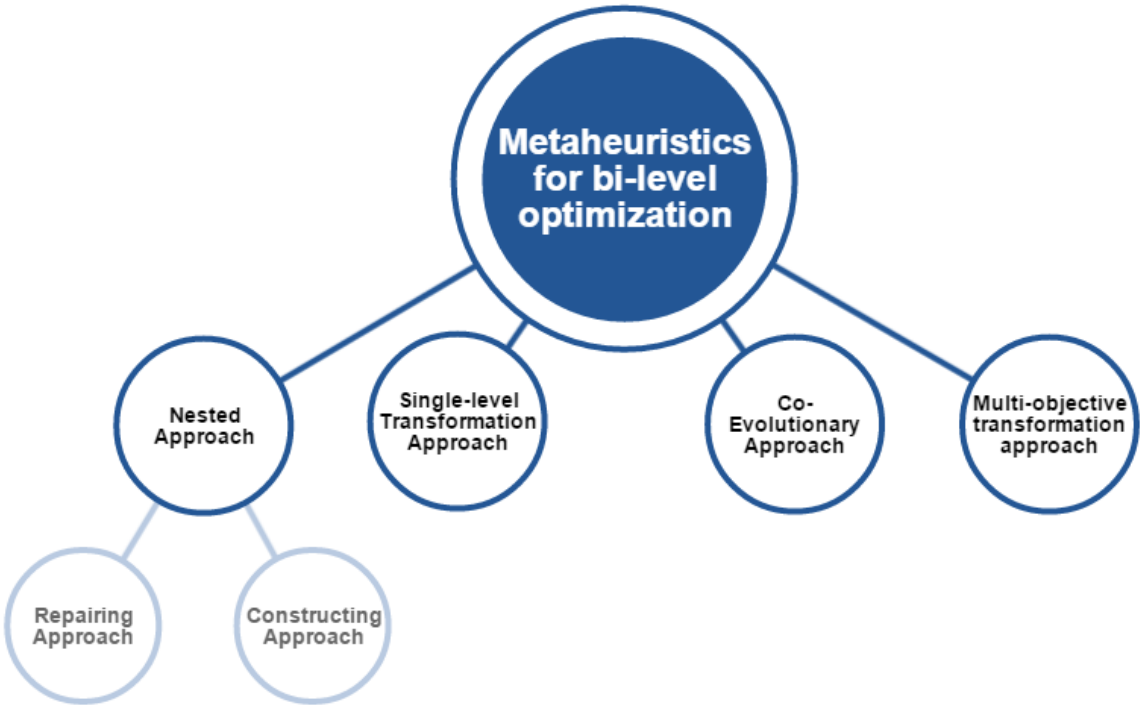


Figure 4.1: Classification of metaheuristic strategies for Bi-level Optimization

4.3.1 Single-level transformation approach

A common approach to solve BOPs is to transform them in a single-level optimization problem by using approximate or exact methodologies which replace the lower-level optimization problem (Dempe et al., 2015). Then the problem can be solved with an appropriate metaheuristic. Several approaches, for example, enumeration methods, penalty methods, marginal function method and trust-region methods, have been proposed for BOPs, under the assumptions that all functions are convex and twice differentiable. In the case of differentiable objectives and constraints in the lower-level problem, a popular approach is to use the Karush-Kuhn-Tucker (KKT) conditions of the lower-level problem as constraints in the upper-level optimization problem (Wang et al., 2008). Additional variables in the upper-level problem are represented as the Lagrange multipliers at the lower level problem.

Other conditions must be satisfied to ensure that the KKT solutions are optimal solutions. For more details refer to Dempe (2002).

4.3.2 Transformation to MOP approach

Considering a problem with more objective functions, a Pareto-based multi-objective approach can be a logical choice for tackling the BOPs. However the different structure of a BOP has to be taken into account. Indeed, the optimal solution of the BOP is not necessarily a Pareto optimal solution of the multi-objective problem, and vice-versa, so solving the BOP using Pareto dominance may not work at all. In fact, in spite of many attempts, no conditions have been found which guarantee that the optimal solution of a BOP is the multi-objective problem Pareto optimality. On the contrary, several authors (Candler, 1988; Clark and Westerberg, 1988; Wen and Hsu, 1989) have demonstrated that they are two different concepts.

Fliege and Vicente (2006) propose a multi-objective optimization approach consisting in the transformation of a BOP into an equivalent MOP. A specific cone dominance concept is used to achieve that. Hence, one can use any metaheuristic for multi-objective optimization to solve the problem. However, this approach is limited to certain assumptions and they can be rarely satisfied. In particular, the mathematical formulation of the MOP uses the derivatives of the objectives of the original BOP, so it is applicable to differentiable problems only.

4.3.3 Nested sequential approach

As detailed in Talbi (2013), hierarchical optimization algorithms try to sequentially solve the two levels in a nested approach. In each level the solution are improved to find a good overall solution for both levels. These algorithms can be divided in two categories:

- **Repairing approach:** the lower-level problem is considered as a constraint only and it is solved during the evaluation step of the leader-level optimization (Koh, 2007). This works on the assumption that the lower optimization problem has a certain structure enabling an efficient problem solving;

- **Constructing approach:** on each level an improving optimization algorithm is used on the population. This is sequentially iterated until a termination criterion (e.g. given number of generations) is satisfied (Li et al., 2006).

As first step of the repairing approach, a solution for the upper-level is generated (i.e. (x, y)). This solution (x, y) becomes the initial solution of the lower level and it is sent thereto, see Figure 4.2. At this point, an optimization algorithm dedicated to the solution of the follower level searches for a "good" solution y^*. The variable x is used as a fixed value parameter at the lower level. The solution of the lower-level (x, y^*) is sent back to the upper-level and the entire solution of the upper-level is replaced by (x, y^*) to compute the upper-level objective value. Those three steps are iterated sequentially until the termination criterion is reached.

Otherwise in the constructive approach, the lower-level problem is solved from a different perspective. The optimization process tries to improve a population of solutions (x, y) generated at the upper-level using the objective function F. This approach is generally used in population based metaheuristics. At this point, the population at the lower-level is improved using the objective function f where the decision variables x are fixed. The improved population of solutions (x, y^*) becomes the initial population at the upper-level. This process is iterated until a termination criterion is reached.

The computational cost and complexity are the main weaknesses of the nested approach because it needs to solve an optimization problem (i.e. lower-level problem) for each solution generated at the upper-level. Furthermore, this approach may produce unfeasible solutions during the interaction between levels. The efficiency of this class of strategies depends strongly on the difficulty of the lower-level problem. For hard lower-level problems, more efficient metaheuristic strategies with more coordination between the two levels must be designed.

4.3.4 Co-evolutionary approach

Methodologies based on the single-objective reformulation, multi-objective reformulation or nested approaches cannot be used or are practically inefficient in some cases. The reason is that usually the traditional approaches target specific types of BOPs or make specific assumptions (e.g. differentiability at upper-level or lower-level problem, low-level structured problems, convex feasible region, upper-level reduced search space). Due to these shortcomings, the said approaches cannot solve complex real-life applications (e.g. BOPs with non-differentiable objective functions, complex combinatorial BOPs, etc.). Therefore general BOPs can be solved as BPPs without any transformation with certain co-evolutionary based metaheuristic approaches developed for this purpose.

In co-evolutionary metaheuristics an optimization strategy is applied at each level, which run in parallel. Commonly the optimization strategy is a population-based metaheuristic.

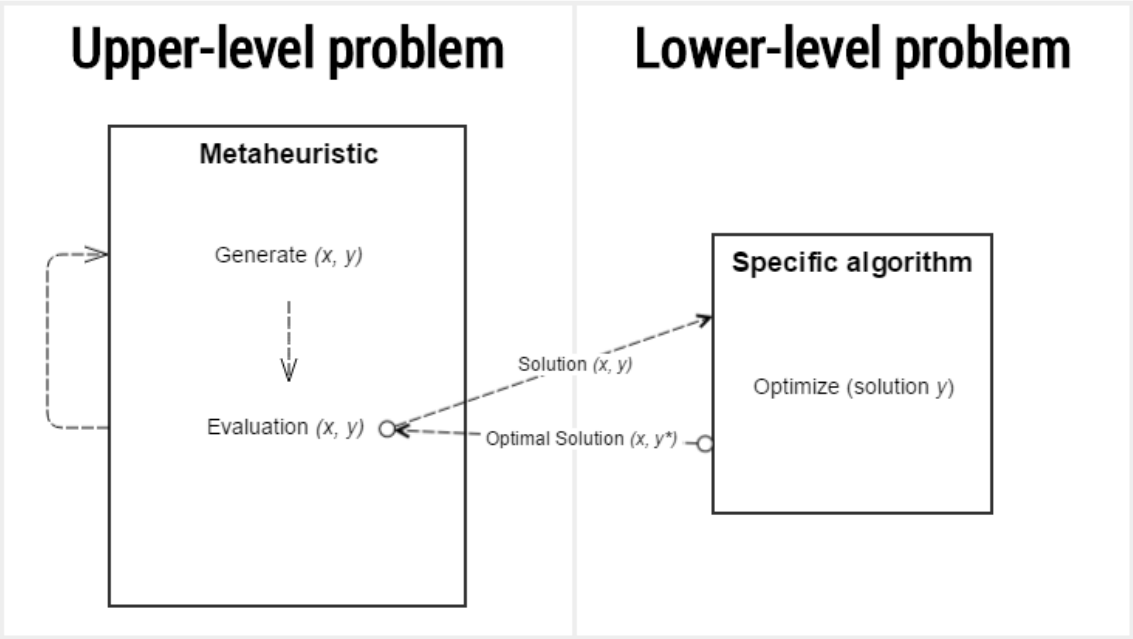


Figure 4.2: Metaheuristics nested repairing approach for solving BOPs

Each level tries to maintain and improve its own population separately. The two populations evolve in parallel. Each population evolves a portion of the decision variables, so the complete solutions are built by means of a cooperative exchange of individuals from populations. Hence, the two levels exchange information to keep the global view of the BOP (for example refer to Oduguwa and Roy (2002) or Legillon et al. (2012)). In general designing co-evolutionary metaheuristics to solve BOPs is challenging, but it can exploit the availability of parallel computing resources (e.g. multi-core, clusters, grids, etc.).

4.4 Pricing Problem Applications: an Overview

Many real world problems can be actually modeled as BPPs. In Chapter 5 and 6, two different MOGASI applications to such problems are presented. They belong to the same category of economic planning and are referred to as Pricing Problems. In general a company may be confronted with a best price strategy problem: it wishes to determine the price of a set of products in order to maximize its revenue. The reaction of potential customers has to be taken into account: if prices are too high, they may decide not to buy the products,

whereas if prices are too low, the company looses revenue. Pricing problems have been widely present in the economic literature for many decades. Furthermore, over the past two decades, this type of problem has attracted increasing attention from researchers in the operational research community, interested in using it to model a range of different applications, from product pricing to highway systems and telecommunication services.

The pricing problems can be efficiently configured using the BP paradigm as a sequential cooperative game between a company and potential customers. In a pricing problem the leader (upper level) sets taxes or charges for some activities and the follower (lower level) selects activities from among taxed and untaxed ones to minimize operating costs. It is assumed that there are n_1 taxed and n_2 untaxed activities. By setting $T \in \mathbb{R}^{n_1}$ as the tax vector, $x \in \mathbb{R}^{n_1}$ and $y \in \mathbb{R}^{n_2}$ as the vectors associated with taxed and untaxed activities respectively, f and g as the objective functions of the leader and of the follower respectively, and $\Pi \subset \mathbb{R}^{n_1+n_2}$ as the feasible solution set, the pricing problem can be formulated as:

$$\begin{aligned}
&\max_{T} && f(T,x,y), && \text{(4.5a)}\\
&\text{s.t.} && (x,y) \in \arg\min_{x,y} g(T,x,y), && \text{(4.5b)}\\
& && (x,y) \in \Pi. && \text{(4.5c)}
\end{aligned}$$

Transportation planning is a typical domain in which hierarchical pricing structures are construed as a Stackelberg game, where the first level corresponds to the network operator who seeks to improve the performance of the network and the second level corresponds to network users making their travel choices. This framework fits many applications, such as the toll optimization of highways, truck toll systems, the pricing of express mail delivery, passenger transportation systems (railways and airlines), and various other pricing schemes (hotel rooms, car rental, travel and tourism packages and telecommunications packages). Migdalas (1995) provides a review of BPPs in the transportation sector, for instance network design, signal setting and origin/destination matrix adjustment problems. These problems usually contain the so-called network equilibrium problem as the second level. Van Hoesel (2008) gives an overview of Stackelberg pricing problems applied to networks. Toll optimization schemes for highways using bi-level programming have been studied by Labbé et al. (1998), Dewez et al. (2008) and Heilporn et al. (2011), with the leader owning the highway and setting tolls on their arcs, and the followers traveling either on the highway or on national toll-free roads.

An extension of the pricing problem in a *product* pricing problem, which applies in a revenue management context: a company wants to determine the optimal prices for a

set of products to maximize the total revenue, taking into account that customers make their choices to maximize their utility (for instance the difference between the benefit they perceive by having the product and the cost of buying it). References on this problem can be found for instance in Shioda et al. (2011). Heilporn et al. (2010a) show that a parallel between this problem and the highway tolling problem described above can be established.

Applications of pricing models to telecommunication networks can be found in Bouhtou et al. (2007). The authors approach the pricing problem of a telecommunications operator owning part of a network in which clients want to route their flows at minimum cost, whereas they consider a restriction of it, considering that each client uses at most one of the operator's arcs (as if the operator should own bridges over a river and wanted to optimally price them). They prove that this restricted problem is *APX-hard*.

Brotcorne et al. (2000) consider a bi-level model for the freight tariff-setting problem, where the leader is a carrier seeking to maximize their revenue by setting optimal tariffs on the subset of arcs it controls, and the follower is a shipping company willing to send a prescribed quantity of goods from origin nodes to customers at minimum cost. They assume that the leader is not a dominant player of the market, implying that the total demand is not influenced by the leader's prices. In fact competitors (i.e. other carriers) do not react in the short term to the leader's prices. The authors first show how the bi-level model can be reformulated as a single level bilinear program. They then propose metaheuristics that explicitly take into account the structure of the network. The efficiency of these algorithms is assessed by comparing their solutions to the optimal solutions obtained from a mixed integer reformulation of the model using a commercial solver. Extensions of these heuristic algorithms have been proposed by Brotcorne et al. (2001). They present and test an algorithmic scheme that can solve toll-setting problems of significant sizes to near optimality within reasonable computing times.

Brotcorne et al. (2008) consider the problem of pricing in a network together with the design issue: this situation is realistic where the network design can be changed in the short term together with the pricing, for example, in telecommunications networks. In this case, the leader is a telecommunications operator, who wants to simultaneously determine the connections to be opened and the tariff to be applied to them. The followers are users who send flows along cheapest paths connecting their respective origins and destinations. Brotcorne et al. (2008) propose a bi-level formulation for this problem and discuss its characteristics. In particular, they show how the capacity constraints (ensuring that positive flows are routed only on open links, and that they respect capacities) present at the lower level can be moved to the upper level without affecting the optimal solution. From an

economic point of view, this behavior can be interpreted as follows: capacity constraints imposed on the users can be enforced through a suitable and finite tariff schedule, and this can be achieved without affecting the leader's revenue. The authors therefore use this property to develop an efficient algorithm, as the lower-level problem is reduced to a set of independent shortest path problems. Finally, they provide some numerical results on both randomly generated and real data.

Marcotte and Patriksson (2007) consider toll setting policies to mitigate the risk related to transportation of hazardous materials. Often a network design approach is used in the literature to tackle this problem, with an authority (e.g. the government) which can prohibit the use of some arcs of the network to transport certain types of hazardous materials so the carriers must choose among the remaining routes in the network. The objective of both the authority and carriers is to minimize a weighted sum of the risk caused by the hazardous materials transportation and the transportation costs with different weights assigned. A simpler version may be constructed considering a zero weight for the transportation cost in the authority objective function and/or for the risk factor in the carrier objective function. Marcotte et al. (2009) propose a new approach to this problem using tolls to channel the shipments of hazardous materials transportation. In this setting the authority does not shape the network by prohibiting the use of certain arcs, but imposes tolls on them and therefore affects the transportation cost and the route choices of the carriers. The objective functions of both the authority and carriers remain the same in a bi-level framework. Marcotte et al. (2009) show that the toll problem is not equivalent to the network design problem, unless there is only one shipment to be done. They also point out that toll setting policies give more flexibility compared to network design policies, as they permit to differentiate between carriers. Different formulations for both toll setting and network design versions of the problem are presented, together with efficient solution methods.

Cardinal et al. (2013) consider a pricing problem where the follower is looking for the minimum spanning tree on a graph and the leader owns a subset of arcs and puts prices to them maximizing his/her revenue. This framework too is applicable to telecommunication problems, for example where a company owns and sells several point-to-point connections between locations and a customer wants to buy a network connecting their locations in the form of a spanning tree. The market is composed of the company and its competitors who own the other connections. Cardinal et al. (2013) show that this problem is APX-hard even if there are only two different cost values on the arcs owned by the competitors. They use an approximation algorithm, provide an integer linear formulation of the problem and study the relation between them.

In Castelli et al. (2013) a bi-level pricing model has been proposed to determine en-route charges in the context of Air Traffic Management (ATM) and re-distribute air traffic and reduce congestion. This application in particular is one of the topics that are a subject of this work and is extensively explained in Chapter 6. Castelli et al. (2013) propose a sequential solution procedure and test it on real air traffic data. They show that a national ATM agency could realistically use this approach to determine the best en-route charge combination for peak and off-peak hours for its airspace. This approach can be extended to other cases of proportional tolls (for instance a per-kilometer toll on a road network).

Finally, the pricing paradigm appears to be an adequate framework for the Principal/Agent Problem (Van Ackere, 1993). In this problem, widely studied in economics, there is a principal who wants to delegate a task to an agent with a retribution. The principal optimizes their utility (e.g. by minimizing the reward offered to the agent) whilst ensuring that the agent accepts the task and performs it in a satisfactory way. In the classical example of the agency theory, the principal is the owner of a company and the agent is the company manager, who should be motivated to act in the interest of the principal even if the principal cannot directly monitor their work. In the basic model the outcome of the task performed by the agent depends on two elements both of which are beyond the principals control: the level of effort exerted by the agent and a random factor, which the agent learns after selecting their level of effort. If the agent refuses the task, they obtain a certain utility level, which should be smaller than the expected utility level in case of acceptance. Details on how these elements can be computed and variations of this basic scheme are explained in Van Ackere (1993). The author considers the model with different risk scenarios and different levels of knowledge of the involved players, and also describes some applications of this model in accounting (for budgetary control systems and variable cost allocation), in industrial organization (for capital structure, disciplining of the product, labor and capital markets and the role of supervision), in marketing (for the relationship between a sales manager and a salesman) and in finance (for the relationship between the shareholders and the manager of a company). Van Ackere (1993) also presents some management science problems in which this framework arises, such as centralized versus decentralized production planning, scheduling of rare resources and selection of batch size. Aside from all the details inherent to each individual real situation to be considered, the Principal/Agent problem could be modeled as a BP pricing problem, where the principal is the leader who wants to determine the minimum reward to offer to one or more agents (followers) to guarantee that the tasks are carried out with good results.

4.4.1 The bilinear pricing problem

This section focuses on pricing problems in which the objective functions at both levels are linear once the choices of the other level are fixed. All constraints are linear as well. The second level objective function has coefficient vectors $c_1 \in \mathbb{R}^{n_1}$ and $c_2 \in \mathbb{R}^{n_2}$, which represent the fixed costs of taxed and untaxed activities respectively. Matrices $A_1 \in \mathbb{R}^{m \times n_1}$ and $A_2 \in \mathbb{R}^{m \times n_2}$, and vector $b \in \mathbb{R}^m$, represent the coefficients of the m constraints of the follower's feasible solution set. The bilinear pricing problem can be modeled as follows:

$$\max_{T} \quad Tx, \tag{4.6a}$$

$$\text{s.t. } (x, y) \in \arg\min_{x,y} \quad (c_1 + T)x + c_2 y, \tag{4.6b}$$

$$s.t. \quad A_1 x + A_2 y = b, \tag{4.6c}$$

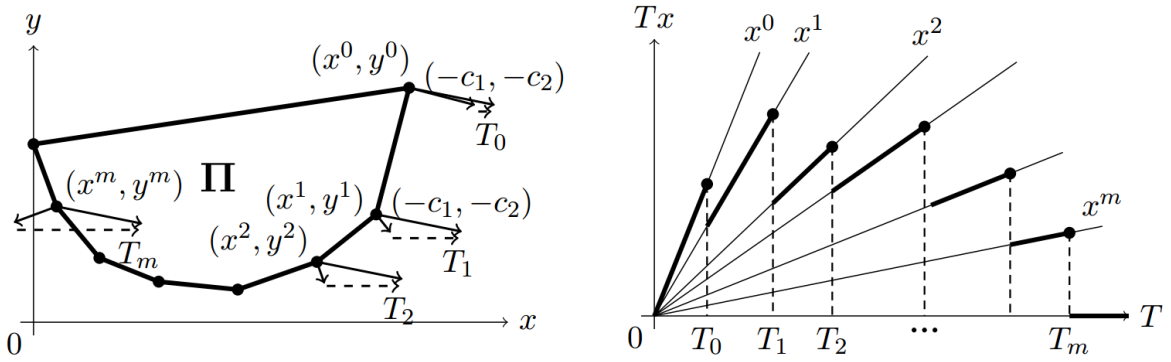
$$x, y \geq 0. \tag{4.6d}$$

To guarantee the existence of a bounded solution, it is assumes that the follower's feasible solution set $\Pi_1 = \{(x, y) : A_1 x + A_2 y = b, x, y \geq 0\}$ is not empty and bounded, and that the follower's set of feasible solutions using only untaxed activities $\Pi_2 = \{y : A_2 y = b, y \geq 0\}$ is not empty. In fact, if Π_1 is not empty and bounded, the second level problem will always have a finite solution. The non-emptiness of Π_2 guarantees the existence of a tax-free solution for the follower, which is necessary to prevent the leader from imposing an infinite tax on their activities leading to an infinite revenue.

For illustration purposes an example where the second level has only two decision variables is presented. This means that the followers can choose between a taxed activity and a free activity, and the leader has to determine a tax value T. The formulation is the same as described above, with decision variables $T, x, y \in \mathbb{R}$, parameters $c_1, c_2 \in \mathbb{R}$ and vectors $A_1, A_2, b \in \mathbb{R}^m$. A graphical representation of the problem is shown in Figure 4.3 and the optimal solution can be found using a relatively straightforward procedure.

Figure 4.3a shows the second level objective function with the set of feasible solutions Π. Each vertex of Π represents a potential optimal solution for the follower. Based on the linear programming theory, it can be easily concluded that a vertex of Π is optimal if the opposite of the objective function coefficient vector $(-(c_1 + T), -c_2)$ belongs to the cone generated by the coefficient vectors of the active constraints at that vertex. This enables to determine for each vertex the values of T for which it is optimal. For instance, vertex (x^0, y^0) is optimal for $T \in [0, T_0]$, (x^1, y^1) is optimal for $T \in [T_0, T_1]$, and so forth. The first level objective function Tx is shown in terms of T in Figure 4.3b. It can be seen that this function is discontinuous

and piecewise linear with slopes x^0, x^1, etc. The optimal solution in this simple example is given by T_1 and (x^1, y^1). For further details on this case see Labbé et al. (1998).



(a) Second level - Feasible solutions

(b) First level - Objective function (in bold)

Figure 4.3: Graphical example of the objective functions of the (bi)linear pricing problem in a two-dimensional case (Labbé et al., 1998)

4.5 MOGASI: Framework Extension

In the pricing problem presented in the next chapter, once the leader decisions have been taken the follower quickly reaches a solution. Hence, the nested approach is identified as the most appropriate action since the assignment of the follower level optimal values is done by means of a custom solver designed for each of the target applications discussed in this work.

This choice can be exemplified as follows. Consider a pricing problem on a network (Chapter 5) with an authority owning a subset of arcs and imposing tolls on them and users traveling on the network. The authority's objective is to maximize its revenue from the tolls, whereas network users want to minimize their total traveling costs and so will always travel using paths which incur the lowest costs. Moreover, users choose their best paths only after the authority has fixed the tolls, thus having a complete knowledge of all costs of the network. The application of MOGASI to solve this kind of problem follows exactly this explanation because it fits the description of a simple nested procedure. The adoption of the MOGASI strategy for managing the authority decision variables is proposed, i.e. the leader level, and the exact assignment of each user to the cheapest available route, i.e. the follower level. The algorithm requires that the follower level optimization task be solved for every new set of leader level variables produced using the genetic reproduction phase. The method relies on a real-coded generational GA and an exact assignment algorithm in the two respective levels.

A simplified description of the proposed strategy is given below, while a detailed algorithm description is presented in Chapter 3:

- **Pre-processing**
  Step 0 - It is a sequence of simple operations iterated on the entire set of linear constraints and variable bounds with the aim of identifying a possible simplified set.

- **Initialization step**
  Step 1 - The target of this step is to provide a set of initial configurations. MOGASI offers four alternatives for the initialization: randomly generated single solution or entire initial population, user-defined single solution or entire initial population.

- **Selection at leader level**
  Step 2 - A selection is performed with a cumulative probability prioritization to determine the best sub-set of candidate parents among all the available individuals.

- **Evolution at leader level**
  Step 3 - Solution Generation phase (i.e. application of Genetic operators) creates the new individuals. This provides the leader level assignment for each new individual.

- **Follower level optimization**
  Step 4 - The lower level optimization problem for each offspring, i.e. minimal cost assignment, is solved exactly with a custom optimizer. This provides the follower reactions to the assignment of the leader level.

- **Evaluation**
  Step 5 - The leader level variables are matched with the corresponding follower level optimal variables for each individual. At this point all quantities required for the optimization problem (leader objectives and constraints) are computed and collected in the complete individuals.

- **Population update**
  Step 6 - From the current pool of complete individuals the most promising set is selected according to the approach described in Section 3.7, at maximum *population size* individuals. The follower level is cleared for the next iteration.

- **Termination criteria**
  Step 7 - If the termination criteria, i.e. maximum number of generations and/or evaluations, is not reached, the algorithm proceeds to the next generation (Step 3).

Bi-level problems have been defined and explained in detail, together with different methods used for tackling them. In this work the MOGASI nested approach has been chosen as the most appropriate method and it has been extended to bi-level problems. To demonstrate the validity of this decision it has been applied to two pricing problem cases discussed in the following Chapters. The first is the Network Pricing Problem and this application consists in an intensive testing of a large number of instances to verify the effectiveness of the proposed method. The second application is aimed at contrasting the air traffic congestion at the European level and involves large scale instances. This study has been conducted in the framework of a European project the details of which are presented in the dedicated Chapter.

CHAPTER 5

NETWORK PRICING PROBLEM

The Network Pricing Problem (NPP) is a pricing problem on a network, with an authority which owns a subset of arcs and imposes tolls on them and users (often referred to as commodities) who travel on the network. The authority is the leader who wants to maximize his/her revenue whereas network users are the followers who want to minimize their costs, and so will always travel on their minimum cost path. Moreover, users choose their best paths once the authority has fixed the tolls, thus having the complete knowledge of all costs of the network.

5.1 NPP on a General Transportation Network

The transportation network is defined as a set of nodes linked by a set of (directed) arcs. N is the set of nodes i, P_1 is the set of toll arcs (i.e. arcs owned by the leader on which he/her can impose tolls) and P_2 is the set of the others, or toll free arcs. p is denoted as the generic arc. The set K represents the commodities k, which are groups of network users traveling from an origin to a destination. To avoid a trivial solution it is assumed that for each commodity there is a toll-free path between its origin and destination, i.e. a path which does not pass through any of the arcs owned by the authority, as otherwise the authority could impose an infinite toll on their arcs to obtain infinite revenues.

An example of such network is shown in Figure 5.1 in which a single commodity has to travel from A to E. The authority owns arcs B-C and D-E (dashed arcs) with fixed arc costs reported on the graph. The toll free path is A-C-E with a total cost of 22, so this is the upper

bound for the total travel cost the commodity is willing to pay. A-C-E is not the shortest path from origin to destination. The lower bound of the total cost is given by the cost of the shortest path on the network if the authority would impose no tolls. This path is A-B-C-D-E with a cost of 6. Hence, an upper bound for the authority's revenue can be calculated as the cost of the shortest toll free path minus the cost of the shortest path if all tolls were set to zero (in this example: $22-6=16$). The toll-free path cost can be computed as the cost of the shortest path on the network in which all tolls are set to infinity. If $\gamma^k(T)$ is the cost of the shortest path for commodity k and toll vector T, this upper bound on the authority's revenue can be written as $UB^k = \gamma^k(\infty) - \gamma^k(0)$. This bound is not always reached, as can been seen in this example: the toll on arc B-C must be at most 5, and on arc D-E at most 10. So the authority's revenue will never exceed 15. In fact, these values provide an optimal solution.

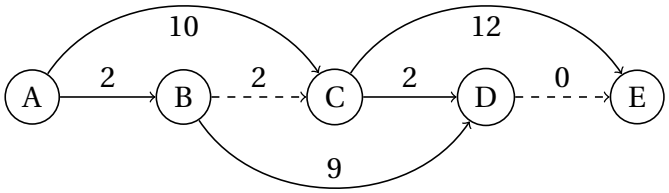


Figure 5.1: Simple example of an NPP (Dewez et al., 2008)

In this work tolls are supposed to be non-negative. Furthermore, there are some cases in which the follower has multiple optimal solutions for a given toll vector. In such cases the commodities are presumed to take the choice which is most profitable for the leader. This assumption is not restrictive, because the leader could modify some tolls of the most profitable solution by a small value, making that solution the only optimal solution for the follower.

5.1.1 Arc pricing

In this section, the definition of arc pricing is presented. This concept means that the leader imposes a toll on each of his/her arcs, and these values can be different.

The notation of arc pricing is given in Eq. 5.1a–5.1d and can be interpreted as follows: for each commodity $k \in K$, let η^k be its demand and o^k and d^k be its origin and destination respectively. Moreover, c_p is defined as the travel cost on arc $p \in P_1 \cup P_2$. The leader wants to set a toll T_p on each toll arc $p \in P_1$, such that his/her total revenue is maximum, and followers will seek their minimum cost path on the network, fixing flow variables x_p^k on

toll arcs and y_p^k on toll free arcs (these variables are equal to 1 if commodity k uses arc p, 0 otherwise). Later in the description the short notation (i, j) is used for the sake of clarity to indicate an arc p of the network, where i is the tail node and j the head node of the arc. The NPP for the arc pricing can therefore be modeled as follows:

$$\max_{T \geq 0} \quad \sum_{p \in P_1} T_p \sum_{k \in K} \eta^k x_p^k, \tag{5.1a}$$

$$\text{s.t. } (x, y) \in \arg\min_{x,y} \quad \sum_{k \in K} \left(\sum_{p \in P_1} (c_p + T_p) x_p^k + \sum_{p \in A_2} c_p y_p^k \right), \tag{5.1b}$$

$$\text{s.t.} \quad \sum_{p \in i^+} (x_p^k + y_p^k) - \sum_{p \in i^-} (x_p^k + y_p^k) = b_i^k \qquad \forall k \in K, \forall i \in N, \tag{5.1c}$$

$$x_p^k, y_p^k \geq 0 \qquad \forall k \in K, \forall p \in P_1 \cup P_2, \tag{5.1d}$$

where i^- and i^+ denote the sets of arcs with i as head or tail respectively, and b_i^k is equal to -1 if i is the origin node of commodity k, 1 if it is the destination node, and 0 otherwise.

As the second level is a shortest path problem (Equations (5.1b), (5.1c) and (5.1d)), whose linear programming formulation has the total unimodularity property, there is no need for integrality constraints on the decision variables. This bi-level NPP for a multicommodity network was first introduced by Labbé et al. (1998).

5.2 Path pricing

In contrast to the previous problem, where tolls are set on each arc and the commodity is paying for each toll arc used in its shortest path, the problem of path pricing is now considered. In this problem tolls are associated to paths. A toll path is defined by the subset of toll arcs contained in it, and a toll is associated directly to the path, i.e. to the subset of toll arcs. Note that each toll path is priced independently and that the toll of a path is not imposed equal to the sum of the tolls of arcs belonging to the path. This path pricing version of the NPP is different from the arc version introduced in Section 5.1.1.

In fact, consider an authority owning a set P_1 of toll arcs on a network $(N, P_1 \cup P_2)$. Solving the arc NPP provides an optimal toll for each arc of P_1. However, the authority may instead decide to set tolls on paths containing toll arcs. Since the authority controls only the toll arcs, setting a price for a path is in fact equivalent to setting a price for using the subset of toll arcs of that path, and all paths containing the same subset of toll arcs should have the same toll. Hence, when several paths contain the same subset of toll arcs, only the path with

the smallest fixed cost will be considered for pricing. Furthermore, the authority may decide to price only some subset of arcs, e.g. those constituting a simple path in the network (i.e. visiting each node at most once). This special path NPP may yield an optimal revenue higher than its arc version.

5.2.1 Path Pricing vs. Arc Pricing

The example shown on Figure 5.1 is an example of arc pricing where the tolls imposed by the leader on their arcs are fixed regardless of the route the commodity is taking from an origin node to reach a destination. Path pricing, on the other hand, presumes that the tolls that the leader imposes on the arcs can be different based on the path that the commodity has chosen to follow. If the above example is considered from the path pricing perspective, the situation is as follows. There are four paths including at least one toll arc plus one toll free path of cost 22 (path A-C-E):

- A-B-C-D-E, with a fixed cost of 6
- A-C-D-E, with a fixed cost of 12
- A-B-D-E, with a fixed cost of 11
- A-B-C-E, with a fixed cost of 16

It is clear that the second toll path should not be considered because whatever the toll the leader chooses for D-E, this path will always be more expensive than the third toll path (the paths share the same toll arc, but the third path has a lower fixed cost). This is an example of path domination. In this simple example the leader may choose a toll of 16 for the first toll path (the one with the smallest fixed cost, i.e. 6) and a large toll on all other paths, such that the shortest path for the commodity will be the first toll path. For this network, the leader's optimal revenue for the path pricing model is thus larger than the one for the arc pricing model, which was 15.

The second example, shown on Figure 5.2, considers three commodities, each of them with unit demand and traveling from o_1 to d_1, from o_2 to d_2 and from o_3 to d_3 respectively. Toll arcs 1-2 and 2-3 are connected as is the case of a highway. Arc pricing gives an optimal solution value of 10 for the leader's revenue with a toll of 3 on arc 1-2 and a toll of 2 on the arc 2-3 for all three commodities. On the other hand, path pricing yields a leader's revenue of 11, with a toll of 3 on path o_1-1-2-d_1, a toll of 2 on path o_3-2-3-d_3 and a toll of 6 on path

o_2-1-2-3-d_2. Again, the leader's revenue for the path pricing problem is larger than (or at least equal to) for the arc pricing problem, regardless of the considered network.

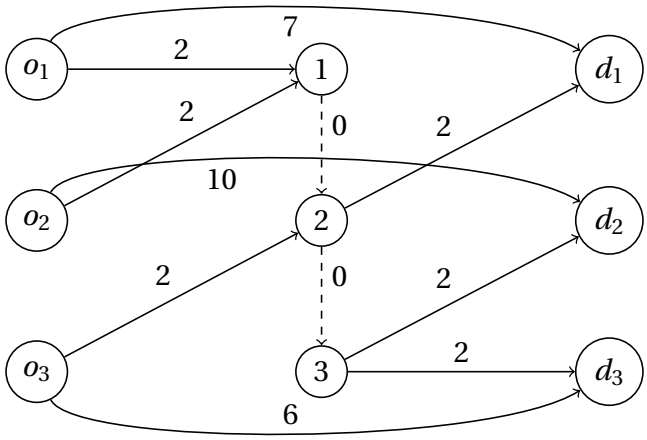


Figure 5.2: Example of arc vs path pricing on a network with connected toll arcs

5.3 Complexity of the Network Pricing Problem

In Labbé et al. (1998), the authors prove that the general problem is NP-hard, while some particular instances are polynomially solvable. In computational complexity theory *non-deterministic polynomial-time problems* (NP-problems) refers to a class of computational decision problems solvable in polynomial time by a theoretical non-deterministic Turing machine, or problems for which a given solution can be verified as a solution in polynomial time by a deterministic Turing machine. The latter means problems for which the instances where the answer is "yes" have efficiently verifiable proofs in polynomial time. NP-hard problems are at least as hard as the hardest problems in NP. Polynomial time refers to the increasing number of machine operations needed by the algorithm relative to the size of the problem, and it can therefore be regarded as the measure of efficiency of the algorithm.

Roch et al. (2005) strengthen the results of Labbé et al. (1998) and prove that the NPP is strongly NP-hard, even for a single commodity and/or when negative tolls are allowed. The question they base their research on is the following: given an instance of the NPP and a constant R, does a toll vector T exist such that $Tx \geq R$, and such that (x, y) is an optimal flow for the second level in reaction to T ? Given a fixed toll vector T, optimal flow variables can be easily determined with a shortest path algorithm, so it can be verified in polynomial time whether the toll vector satisfies the above condition. This places the NPP in the NP complexity class.

In case of a single commodity, Roch et al. (2005) also provide a polynomial approximation algorithm, with an approximation factor of $\alpha = \frac{1}{2} log_2 |A_1| + 1$, where $|A_1|$ is the number of toll arcs in the network. This means that such an algorithm is guaranteed to compute a feasible solution with an objective value of at least OPT/α, where OPT is the optimal leader's revenue.

Recently, Joret (2011) shows that the single commodity NPP is APX-hard, with a reduction from 3-SAT-5 (i.e. 3-SAT where each variable appears in exactly 5 clauses) and a network structure similar to the one developed by Roch et al. (2005) for the NP-hardness proof.

5.4 Bi-level Programming Formulation

The mathematical notation of the path pricing version of NPP is given in the following paragraphs. P represents the set of toll paths whose number is polynomial or fixed. For a general network with n nodes, the maximum number of paths is exponential. For this reason the path pricing problem is considered only for situations or networks where the number of toll paths is polynomial or fixed, as is the case with the highway case discussed in Section 5.5. For each commodity each toll path has a fixed cost of c_p^k, which is the sum of the fixed costs of the arcs belonging to the path. For each commodity the shortest toll-free path is then considered, with cost c_{od}^k and the associated binary decision variable y_{od}^k. Using this notation, the NPP can be described as follows:

(HPBP)

$$\max_{T \geq 0} \quad \sum_{k \in K} \eta^k \sum_{p \in P} T_p x_p^k, \tag{5.2a}$$

$$\text{s.t. } (x, y) \in \arg\min_{x,y} \quad \sum_{k \in K} \left(\sum_{p \in P} (c_p^k + T_p) x_p^k + c_{od}^k y_{od}^k \right), \tag{5.2b}$$

$$\text{s.t.} \quad \sum_{p \in P} x_p^k + y_{od}^k = 1 \qquad \forall k \in K, \tag{5.2c}$$

$$x_p^k, y_{od}^k \in \{0, 1\} \qquad \forall k \in K, \forall p \in P, \tag{5.2d}$$

where constraints (5.2c) allow one path choice for each commodity. In this formulation variables need to be binary (it can be easily seen that this condition could be relaxed for y variables).

The multicommodity path NPP has been proved to be strongly NP-hard by Heilporn et al. (2010b), whether toll arcs are single or bidirectional, based on a reduction from the problem 3-SAT (the proof is similar to the one described in Section 5.3 for the arc NPP).

Contrary to the arc version, the path NPP with one commodity is polynomial (Dewez et al., 2008) and can be expressed as follows:

$$\max_{x,T} \quad \eta \sum_{p \in P} T_p, \tag{5.3a}$$

$$\text{s.t.} \quad \sum_{p \in P} x_p = 1, \tag{5.3b}$$

$$T_p \leq M_p x_p \qquad \forall p \in P, \tag{5.3c}$$

$$x_p \geq 0 \qquad \forall p \in P, \tag{5.3d}$$

where $M_a = \max\{0, c_{od} - c_a\}$, for all $p \in P$.

Indeed, the toll path yielding the largest revenue for the leader can be found in polynomial time, searching for the shortest path on the network without any toll. The toll on this path p is set to M_p, whereas tolls on the other paths are set to sufficiently large values such that the commodity is not interested in taking them. This remains valid even if the number of paths to price is not polynomial. Heilporn et al. (2010a) provide a complete description of the convex hull of solutions in this case.

5.5 Highway Problem: NPP with Connected Toll Arcs

This work considers a particular case of NPP, as discussed in Heilporn et al. (2010a, 2011), where toll arcs are connected in such way that they constitute a single path, as for instance in a highway (see Figure 5.3), with pairs of entry and exit nodes and a toll associated to the sub-path delimited by each of these pairs. If the assumption is made that users who have left the highway do not re-enter it, paths considered for toll can be uniquely determined by their entry and exit nodes. As a consequence, for n nodes in the highway, the number of total tolled paths will be n^2. Due to the completeness of the toll subgraph, this problem is also called the clique pricing problem (Heilporn et al., 2011).

Figure 5.3 shows the so-called Complete Toll NPP, introduced in Heilporn et al. (2010b): toll free arcs are inserted between origin and destination nodes, as well as from the origin and destination nodes to the highway, representing shortest toll free paths between the corresponding nodes. Each pair of entry and exit nodes of the highway is linked by a toll

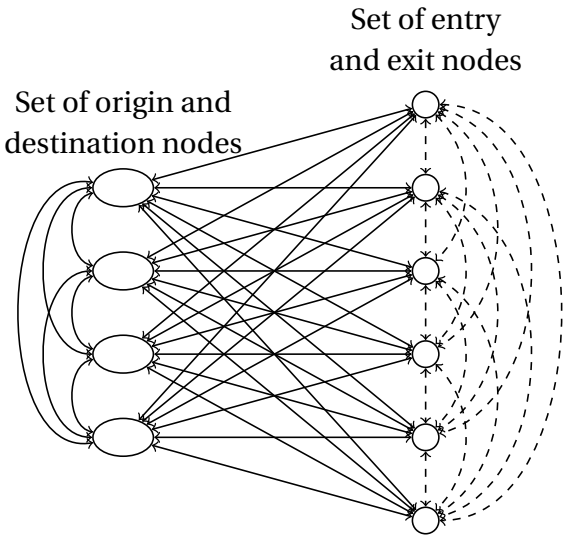


Figure 5.3: Complete toll NPP (Heilporn et al., 2010b)

sub-path $p \in P$, where P represents the set of all toll sub-paths/pairs entry-exit nodes, with $|P| \leq n^2$. Each toll sub-path can be represented by a single artificial toll arc (dotted arcs). In this a case additivity conditions are not considered, meaning that the toll of a path might not be equal to the sum of tolls on sub-paths composing it. For each of these toll sub-paths p, and each commodity k, the fixed cost is denoted as c_p^k and is calculated as the sum of the cost of the shortest toll free path from o^k to the entry node of the toll path $t(p)$, the fixed cost of the toll sub-path and the cost of the shortest toll free path from the exit node of the toll path $h(p)$ to d^k.

Decision variables are defined as follows: the authority wants to set a toll T_p on each toll sub-path $p \in P$ to maximize their revenue, whereas the network users will seek their minimum cost path on the network, fixing flow variables x_p^k on toll paths, and y_{od}^k on their toll free path (these variables are equal to 1 if the commodity k uses paths p or od respectively, and 0 otherwise).

Computational experiments revealed that triangle inequalities and monotonicity constraints on the toll variables may not be satisfied by the optimal solution if they are not explicitly included. The guarantee of these conditions is important for real applications, as triangle inequalities prevent a commodity being able to pay less using two (subsequent) highway arcs instead of the direct one from the same origin and destination, and monotonicity constraints imply that the toll of a path cannot be smaller than the toll of any of its sub-paths. The highway problem without triangle inequalities and monotonicity constraints can be modeled as the path pricing introduced above (Equations (5.2a) to (5.2d)). Both

versions, with and without these constraints, have been proved to be strongly NP-hard (Heilporn et al., 2010b).

5.6 One level Reformulation

The formulation presented with Equations (5.2a)–(5.2d) can be reformulated as a single level optimization problem. First, the follower's objective function can be separated for each commodity, and the lower level optimization problem can be replaced by constraints stating explicitly that the used path is the shortest one (Eq. (5.4b) and (5.4c)). Moreover, variables y_{od}^k, associated to toll free paths can be eliminated using constraints (5.2c).

A new set of variables $p_p^k = x_p^k T_p$ is introduced, having only one non-linear set of constraints in the model (5.4e). These variables represent the toll paid by commodity k for the path p, and they are equal to the toll of the path if the commodity is using it, 0 otherwise. Therefore, the non-linear single level HP can be written as follows (Heilporn et al., 2010b):

(HPNL)

$$\max_{T,x,p} \quad \sum_{p\in P}\sum_{k\in K} \eta^k p_p^k, \tag{5.4a}$$

$$\text{s.t.} \quad \sum_{p\in P}\left[(c_p^k - c_{od}^k)x_p^k + p_p^k\right] - T_q \le c_q^k - c_{od}^k \qquad \forall k \in K, \forall q \in P, \tag{5.4b}$$

$$\sum_{p\in P}\left[(c_p^k - c_{od}^k)x_p^k + p_p^k\right] \le 0 \qquad \forall k \in K, \tag{5.4c}$$

$$\sum_{p\in P} x_p^k \le 1 \qquad \forall k \in K, \tag{5.4d}$$

$$p_p^k = T_p x_p^k \qquad \forall k \in K, \forall p \in P, \tag{5.4e}$$

$$x_p^k \in \{0,1\} \qquad \forall k \in K, \forall p \in P, \tag{5.4f}$$

$$T_p \ge 0 \qquad \forall p \in P. \tag{5.4g}$$

Finally this (HPNL) model can be linearized eliminating the non-linear set of constraints (5.4e), introducing the following set of constraints:

$$p_p^k \le M_p^k x_p^k \qquad \forall k \in K, \forall p \in P, \tag{5.5a}$$

$$T_p - p_p^k \le N_p(1 - x_p^k) \qquad \forall k \in K, \forall p \in P, \tag{5.5b}$$

$$p_p^k \le T_p \qquad \forall k \in K, \forall p \in P, \tag{5.5c}$$

Constants M_p^k represent upper bounds on p_p^k variables, i.e. the highest value that commodity k is willing to pay to use path p. A valid value is thus $M_p^k = \max\{0, c_{od}^k - c_p^k\}$, $\forall k \in K$ and $\forall p \in P$. Constants N_p represent upper bounds on the cost of a path for all commodities, and a valid value is $N_p = \max_{k \in K: p \in P} M_p^k$, $\forall p \in P$. Numerical results show that using these valid values instead of arbitrary large ones significantly improves the solution time of the formulation. The HP can be therefore reformulated as a mixed-integer linear problem as follows (Heilporn et al., 2010b):

(HPL)

$$\max_{T,x,p} \quad \sum_{p \in P} \sum_{k \in K} \eta^k p_p^k, \tag{5.6a}$$

$$\text{s.t.} \quad \sum_{p \in P} \left[(c_p^k - c_{od}^k) x_p^k + p_p^k \right] - T_q \leq c_q^k - c_{od}^k \qquad \forall k \in K, \forall q \in P, \tag{5.6b}$$

$$\sum_{p \in P} \left[(c_p^k - c_{od}^k) x_p^k + p_p^k \right] \leq 0 \qquad \forall k \in K, \tag{5.6c}$$

$$\sum_{p \in P} x_p^k \leq 1 \qquad \forall k \in K, \tag{5.6d}$$

$$p_p^k \leq M_p^k x_p^k \qquad \forall k \in K, \forall p \in P, \tag{5.6e}$$

$$T_p - p_p^k \leq N_p (1 - x_p^k) \qquad \forall k \in K, \forall p \in P, \tag{5.6f}$$

$$p_p^k \leq T_p \qquad \forall k \in K, \forall p \in P, \tag{5.6g}$$

$$p_p^k \geq 0 \qquad \forall k \in K, \forall p \in P, \tag{5.6h}$$

$$x_p^k \in \{0, 1\} \qquad \forall k \in K, \forall p \in P. \tag{5.6i}$$

Violin (2014) shows that, if all cost and parameters are integer, (HPNL) is a fully integer formulation, and therefore optimal toll values are integer. In fact, when flow variables x_p^k are fixed, the matrix associated to the constraints of the obtained problem is totally uni-modular. This is true for any choice of values for the flow variables, and in particular for the optimal values.

5.6.1 Valid inequalities

Finally, Heilporn et al. (2011) introduced some valid inequalities that amount to a strengthening of constraints (5.6b) and (5.6c). They are called "Strengthened Shortest Path Inequalities" (SSPI) and can be expressed as follows:

$$\sum_{p\in P}\left[(c_p^{k_1}-c_{od}^{k_1})x_p^{k_1}+p_p^{k_1}\right]-T_q\leq c_q^{k_1}-c_{od}^{k_1}+\sum_{p\in P\setminus(S\cup\{q\})}\left(p_p^{k_2}+(c_p^{k_1}-c_q^{k_1})x_p^{k_2}\right) \tag{5.7a}$$

$$\sum_{p\in P}\left[(c_p^{k_1}-c_{od}^{k_1})x_p^{k_1}+p_p^{k_1}\right]\leq c_{od}^{k_1}+\sum_{p\in P\setminus S}\left(p_p^{k_2}+(c_p^{k_1}-c_{od}^{k_1})x_p^{k_2}\right) \tag{5.7b}$$

$\forall k_1\in K, \forall k_2\in K, \forall q\in P$ and for any subset $S\subseteq P$

The number of subsets S is exponential, and hence the number of constraints (5.7a) and (5.7b), but the authors propose a separation procedure to determine the most violated constraint which has a complexity of $O(|K||A|\log|A|)$. These families define facets of the convex hull of feasible solutions for the two-commodity case (for more details about their validity and separation procedure see Heilporn et al. (2011)).

5.7 MOGASI: Framework Application to NPP

As previously mentioned in Chapter 4, this work proposes to solve bi-level problems with the MOGASI optimization framework applied in a nested approach (see Section 4.3.3). This can be done effectively when the required computational effort for the solution of the follower problem is low. As already explained in detail and as the name suggests, these approaches rely on two optimization algorithms, sequentially executed one within the other for each evaluation. First and foremost, it is important to decide to use either an evolutionary algorithm at both levels or an evolutionary algorithm at the leader level and an exact optimization algorithm at the follower level.

In this work, a single-objective GA was applied at the leader level and an exact custom solver at the follower level. The exact solver for the follower level was included directly in the source code of the framework. The follower problem is rather simple once the leader's decision variables are fixed. For this reason the required implementation time has been estimated to be comparable to the setup time of an integration with a suitable third-party software.

In this context a simple iteration loop has been defined, as shown in Figure 5.4. During the optimization loop of the GA managing the leader level, the algorithm assigns the values to the toll decision variables. In the "Evaluate individuals" phase these tolls are sent to the follower level. The exact solver, which manages the follower level, identifies the path with the lowest cost and assigns it to each network user. This flow variable assignment is sent back to the leader and will allow him/her to evaluate his/her objective function and the quantitative indexes of each solution. The leader will consequently be able to determine whether this

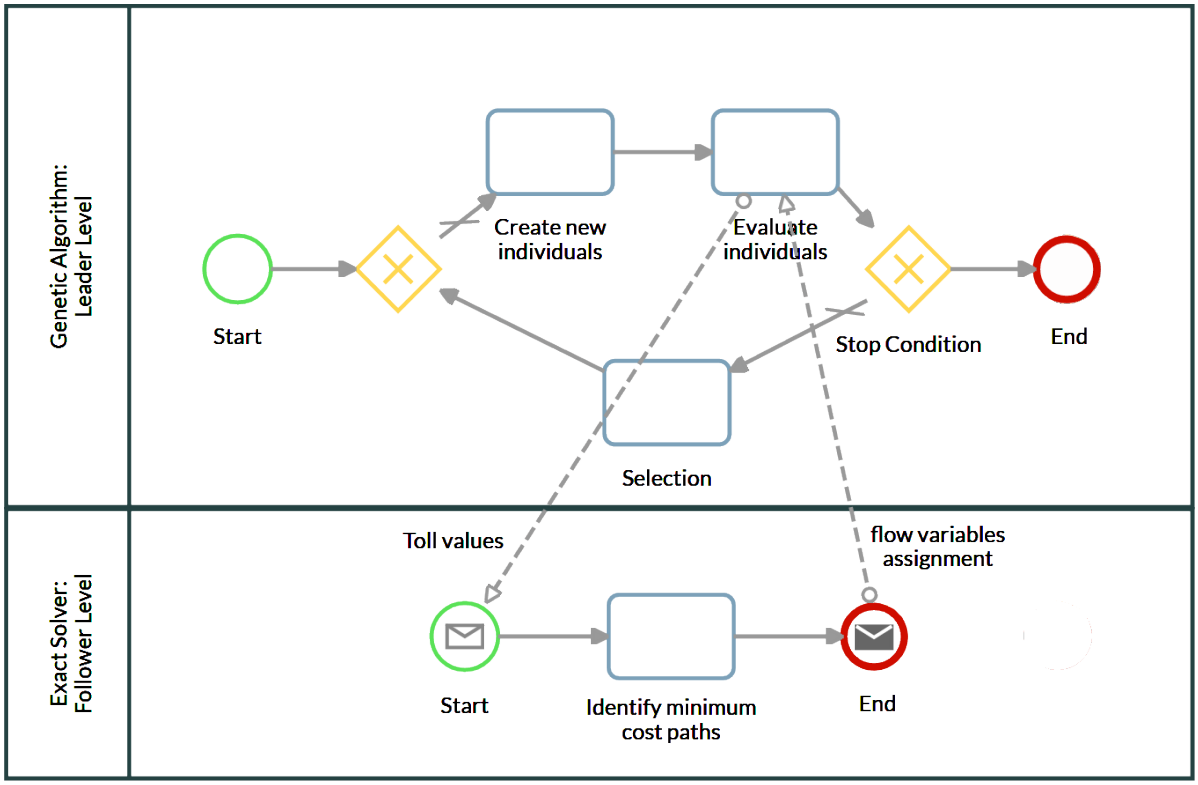


Figure 5.4: MOGASI nested approach for solving NPP bi-level optimization problems

solution is eligible for guiding the creation of the next generation of solutions. The entire loop is repeated until the termination criterion is reached, indicated as Stop Condition in Figure 5.4.

5.8 Experimental Design

This Section contains the details of the procedure used to create the instances tackled in this application, solution evaluation criteria and decisions taken to configure the optimization problem. Furthermore, the experimental data and the post-processing analysis thereof are presented. Due to the post-processing outcome, however, another set of benchmarks is performed. Focusing only on the large scale instances, the relative experimental data and results of the analysis are presented.

5.8.1 Instance Design

Two categories of instances are randomly generated, i.e. complete and partial networks, to test the proposed optimization framework. For both categories a set of five different problem dimensions is taken in consideration, i.e. instances with 20, 56, 90, 132 and 180 commodities and toll paths. These instances do not correspond to any underlying network, in the sense that costs and demand data for each toll path and commodity are randomly generated. Fixed costs on toll paths are randomly set between 1 and 20, and demand for commodities is randomly selected between 1 and 10. Complete network means that toll free path costs are randomly generated between 20 and 30 (and so are always larger than fixed costs of toll paths), such that all commodities could use all toll paths, whilst for a partial network the set of toll free path costs is randomly generated between 10 and 30, meaning that some toll paths will never be interesting for some commodities.

These kinds of instances have been introduced in Shioda et al. (2011) to test product pricing problems, and later used by Heilporn et al. (2011) to compare the NPP formulation and the product pricing formulation for solving product pricing problems. Shioda et al. (2011) address randomly generated instances with 40, 60 or 80 customers (commodities) and 20, 40 or 60 products (toll paths). The toll path network is complete in the sense that all commodities can use each toll path. In a previous work of Violin (2014) numerical tests revealed that these instances, despite having a similar size as the proposed ones, are easier to solve (they have a smaller gap at the root node). On the other hand, the randomly generated instance are identified as quite difficult to solve (they have a large gap at the root node), such that they are better candidates for heuristic algorithm benchmarking. It is relevant to underline that for standard solution methods an instance with dozens of tolls and commodities is considered a large scale problem.

5.8.2 Optimization Problem Setup

The bi-level formulation of the problem has not been modified with respect to the descriptions given in Section 5.4 and follows the explanation provided in Section 5.7 and distributed between two methods involved in the nested approach: the GAs and the custom exact solver.

An analysis of the problem definition from a practical point of view is as follows. The optimization at the leader level is set to maximize a single objective, whereas the follower is set to perform a comprehensive item-by-item enumeration of path alternatives in function of the assignment of paths entailing a minimum total cost to each commodity. The follower level requires a very low computational effort, which makes the actual computations rather

quick. In fact, even in the largest problems instances, i.e. involving a network of 180 tolls and 180 commodities, the time required for the computation is negligible.

Up to a certain extent the proposed nested approach in this particular application can be compared to the classic Black Box optimization. Once the GA has fixed the tolls (that are its decision variables), it has to call the Black Box and wait to retrieve the responses which will then be used to determine the quality of the variable configuration. This similarity with the Black Box is due to the hierarchical two-level decomposition of the problem, which has left to the follower the simple task of minimal cost path assignment. This makes the reaction of the follower always possible and entirely deterministic once the leader has made his/her decision.

The aim of this chapter is to assess the performance of MOGASI applied to bi-level problems with a nested approach with respect to the solution method traditionally used for these kinds of problems. The idea is to identify an application field, or at least a niche, in which a metaheuristics such as this one may compete against or even surpass the performance of traditional linear solvers. A high number of instances should be run to obtain information in this regard. These instances are generated by combining all tolls and commodities values (see Section 5.8.1). More specifically, there are five different values for tolls and five for commodities, which is translated into 25 value combinations for the NPP with connected toll arcs. For each of these combinations, 10 different random instances were generated in order to obtain the average behavior. This was necessary because the generation process is stochastic. Being also MOGASI stochastic, each instance is run 10 times with it in order to obtain average results. This procedure has been performed for both categories of the problem under examination, i.e. for both complete and partial networks.

The following Sections contain the results of the comparison of MOGASI with the solver of the commercial software *FICO Xpress Optimization Suite*. For this purpose the HPL model and the routines used for obtaining the initial solutions described in Section 5.8.4 were implemented in the Mosel language and solved with the Xpress solver v8.0.

The run configuration of both algorithms was fine-tuned using the *Iterated Race (Irace)* tool (López-Ibáñez et al., 2011) to obtain the best possible parameter settings for the branch and bound solution of the HPL problem and for the MOGASI framework. The fine-tuned parameters in the former include pre-solving routines, built-in cuts and in particular the branching choice parameter; the parameters of MOGASI have already been discussed in Chapter 3.

5.8.3 Algorithm Parameter Tuning: Irace Package

An algorithm is defined by different parameters, many with several possible values, leading to a big number of different configurations of the algorithm. The large number of possible parameter configurations makes a manual testing thereof virtually impossible. Instead, a parameter tuning was performed in the IRIDIA research group of the Université libre de Bruxelles (ULB) using Irace, which is a tool for the automatic configuration of algorithms (López-Ibáñez et al., 2011). Its main purpose is to automatically configure optimization algorithms by finding the best configuration of the parameters given a set of instances of the problem. Parameter values constitute a configuration: they can be numerical, boolean or categorical (i.e. they take discrete value without an implicit order), and it is also possible to specify conditional rules between parameters. A cost measure is used to assess the quality of a configuration over each instance of the set, which could be the best objective function value found within a given computational time or the computational time required to solve the instance, possibly bounded by a maximum cut-off time. After that, Irace searches for the configuration which optimizes a function of the configurations cost over the whole set of instances, which is typically the expected cost value or mean cost. It implements an iterated racing framework of a number of searches through the configurations space, which works in the following way. A race starts with a finite set of candidate configurations and is composed of several steps. At each step the candidate configurations are evaluated on each instance of the set, and after each step, statistical tests discard the candidates performing worse than the others. Then the race continues on the remaining candidate configurations, until the predefined minimum number of remaining configurations or the predefined computational budget is reached. Note that different tests can be used and that the predefined computational budget may be the overall computational time or number of experiments. For more details about the exact implementation and how to use it refer to López-Ibáñez et al. (2011). In order to achieve a proper tuning, a dedicated instance set is used. These instances are randomly generated with the same logic as those in Section 5.9, but the dimensions taken into consideration are higher and can reach 1,000 commodities and 180 tolls.

5.8.4 Initial Solution Creation

Providing good solutions as the initial set for a metaheuristic like the GAs is considered extremely beneficial for the subsequent optimization, especially in the case of constrained problems. Exploiting the known structure of the problem is a very easy method to obtain a

lower bound and a feasible solution in the HPL formulation. This method can be applied by assigning all commodities to the path that bears the minimum fixed cost for each commodity. This is either the path leading to the highest potential revenue (i.e., the one corresponding to Q^k) or the toll-free path in case every other path is more expensive than the toll free path for that commodity.

This assignment is feasible by definition, since infeasibility can occur only in case there exists a path that is always cheaper than the assigned one (i.e., the assigned path is dominated by another path for that specific commodity).

Once the commodity assignment $\overline{x}$ is given, the corresponding tolls can be obtained by solving the following LP (which is a single level version of HPBP with second level objective reformulated into cheapest path constraints).

(HPLP-$\overline{x}$)

$$\max_{T \geq 0} \quad \sum_{k \in K} \eta^k \sum_{p \in P} T_p \overline{x}_p^k \tag{5.8a}$$

$$\text{s.t.} \quad \sum_{p \in P} \left[(c_p^k - c_{od}^k + T_p) \overline{x}_p^k \right] \leq c_q^k - c_{od}^k + T_q \qquad \forall k \in K, \forall q \in P \tag{5.8b}$$

This Minimum-fixed cost path assignment is used for all the random generated problems in order to provide a good starting solution for the MOGASI algorithm. Obviously the solution does not change over the iterations of the same instance.

5.9 Experimental Results

This Section presents the results of 5,000 MOGASI runs on the corresponding 500 randomly generated instances also solved with Xpress. A meaningful and effective analysis of the results can be extremely problematic due to the sheer quantity of available data. For this reason only the highest level of result aggregation is presented in this Section. The benchmark results of complete and partial randomly generated instances are available in Table 5.1 and 5.2 respectively.

For each instance dimension, constituted by a commodity (k) and toll (t) value combination, the results reached by the GA and the Exact Solver are presented. As explained in Section 5.8.2 ten randomly generated instances were run for each dimension. In order to obtain meaningful average results an additional calculation was required. This consisted in the computation of the absolute distance from the values of the BEST BOUND, which were

taken as the reference values (see Eq. 5.9). In this way the results are presented as a deviation percentage (hereinafter referred to as Gap) of the found solution from the hypothetical optimal solution.

$$Gap_{k,t} = 1 - \left(\frac{\frac{1}{n} \sum_{i}^{n} z^{*}_{i,k,t}}{z^{b}_{k,t}} \right) \tag{5.9}$$

Numerical experiments presented in this Section have been run on a 64 bit Intel(R) Xeon(R) E5520 @ 2.27GHz quad core CPU computer with 16GB of RAM memory and Debian 8.0 operating system. Maximum execution time for the Xpress solver was set to 3,600 seconds and a termination criterion of 10,000 evaluations was set to the MOGASI.

5.9.1 Analysis of Complete Network Results

The purpose of this Section is to identify an area in which the GA can compete against or surpass the performance of the Xpress solver. For this reason the presented results refer to the average algorithm behavior considering the different tested dimensions. Since the bi-level problem is governed by the leader, the first results are presented with tolls being given a place of prominence. Figures 5.5 are composed of six line charts, each showing the Gap expressed as a deviation percentage between the obtained results and the BEST BOUND value. The Gap can be thus intended as a minimization objective represented on the y axis of each chart for different problem dimensions. Each chart shows the results obtained on instances with the same Toll dimension, which is indicated in the chart header, matched with different commodity dimensions represented on the x axis. The same color coding has been used on all charts and it is shown at the bottom right corner of the set. Its interpretation is as follows:

- Green – average Gap value reached by the Xpress solver on 10 instances
- Blue – average of 10 instances of the average results obtained by the GA in 10 runs for each instance
- Red – average of 10 best results, one for each instance, obtained by the GA in 10 runs for each instance

The first analysis presents the results of the complete category instances, see Figure 5.5, and shows that the two compared approaches have rather different characteristics. The Xpress exact solver, represented with green lines, easily reaches the optimal value in

Table 5.1: Complete NPP Instances: Average Gaps and Execution Times

Complete Instances						
		Average Gaps			**Execution Times**	
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
20	20	0,0747	0,0440	0,0000	0,73	3,86
20	56	0,0333	0,0133	0,0000	6,93	14,49
20	90	0,0250	0,0063	0,0000	16,55	42,04
20	132	0,0430	0,0294	0,0319	25,54	60,45
20	180	0,0689	0,0615	0,0707	36,04	68,97
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
56	20	0,1393	0,1052	0,0000	0,81	54,64
56	56	0,0823	0,0587	0,0244	6,73	3600
56	90	0,1130	0,0977	0,0831	14,10	3600
56	132	0,1391	0,1270	0,1245	20,52	3600
56	180	0,1547	0,1442	0,1476	28,70	3600
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
90	20	0,1521	0,1242	0,0000	1,07	3600
90	56	0,1098	0,0960	0,0554	6,61	3600
90	90	0,1537	0,1378	0,1192	14,02	3600
90	132	0,1704	0,1546	0,1390	22,71	3600
90	180	0,1566	0,1448	0,1448	29,50	3600
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
132	20	0,1881	0,1382	0,0000	1,58	3600
132	56	0,1564	0,1369	0,0837	6,43	3600
132	90	0,1606	0,1459	0,1140	14,01	3600
132	132	0,1804	0,1585	0,1443	22,09	3600
132	180	0,1707	0,1598	0,1480	27,36	3600
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
180	20	0,2038	0,1595	0,0000	2,61	3600
180	56	0,1616	0,1444	0,0959	6,85	3600
180	90	0,1789	0,1588	0,1272	16,28	3600
180	132	0,1821	0,1570	0,1363	23,56	3600
180	180	0,1815	0,1643	0,1572	31,16	3600

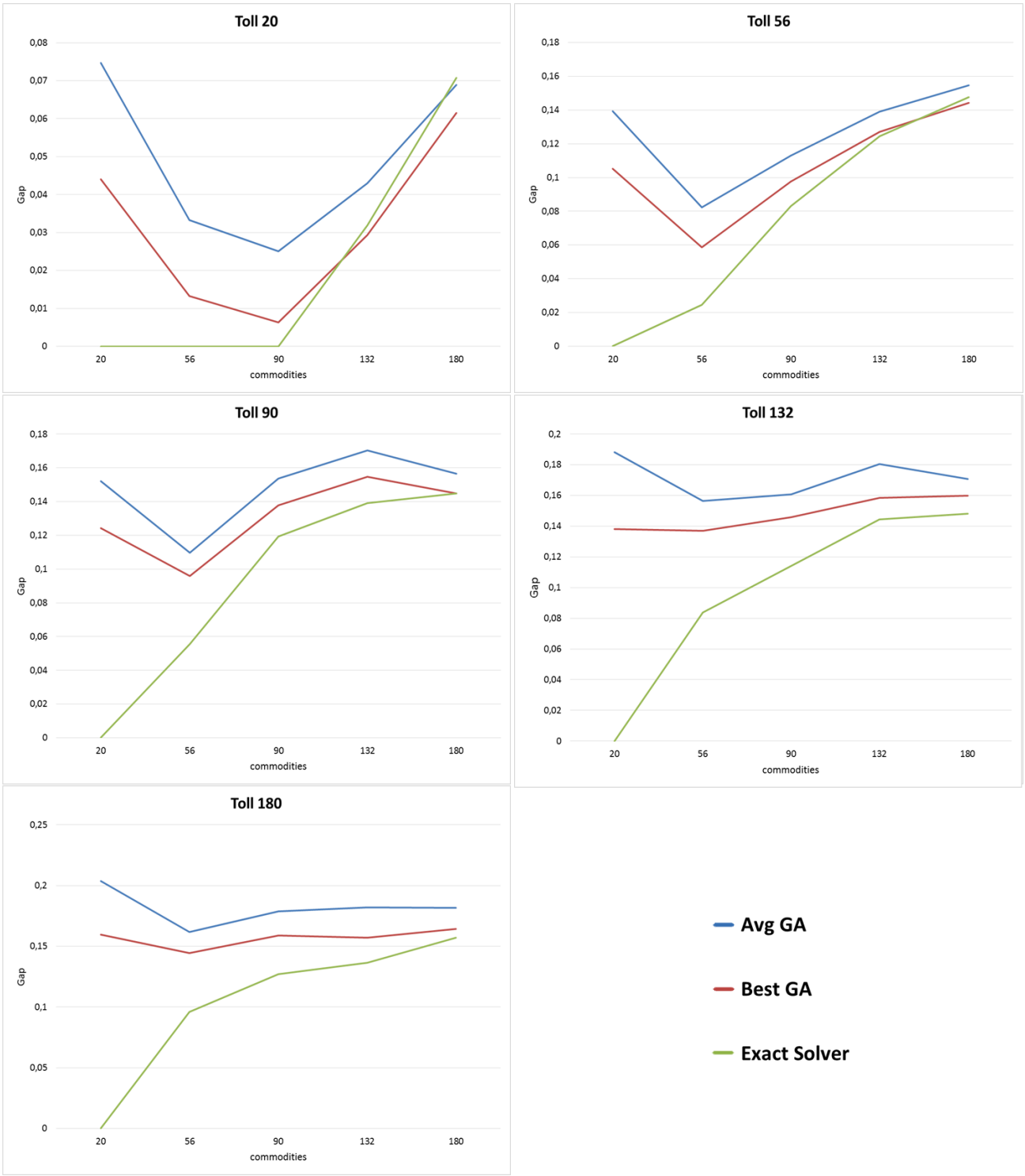


Figure 5.5: Complete NPP Randomly Generated Instances: Toll Perspective

instances with a low number of commodities, in particular those with 20 commodities. Instead, the GA has difficulties in respecting the defined computational budget for the same instances. However, as the number of commodities increases, this trend slowly changes and the three performance lines move closer one to another. In fact, the point in which they are closest corresponds to the maximum number of commodities. This behavior is consistent with the behavior described in literature and thus validates the experiments. As said before in Section 5.8.1, the instances with an increasing number of both tolls and commodities become more and more complicated to solve. This can be observed in particular on the chart Toll 20 where the performance of the exact solver drastically deteriorates for commodity classes 132 and 180. This deterioration, although in a lesser extent, can also be seen on the remaining Toll charts of this set.

In spite of the fact that both dimensions, i.e. tolls and commodities, contribute to the problem complexity, the commodities seem to have a larger impact on the Xpress solver performances. An analysis of the blue and red lines, that are those indicating the behavior of the MOGASI algorithm, shows an unusual change in convexity in relation to instances with a low number of commodities. This phenomenon is observable on all five charts in Figure 5.5. A hypothesis is that it is more difficult for the GA to solve problems with a low number of commodities because their distribution implies smaller shifts in the leader objective function values, which the GA is tasked with solving. This scenario can generate a multitude of equivalent solutions for the leader, which in turn translates into an objective landscape that strongly hinders a fast convergence of a generational algorithms as is the GA.

It can be generally observed that the Best GA has always some Gap advantage with respect to the Average GA. This is a further indicator of the problem multimodality, that has an impact on the stability of the solution quality of a GA, which in any case cannot be guaranteed.

The most promising cases for the GA with respect to the exact solver are the instances with the highest number of commodities, as shown in Figure 5.6. On the following two charts the results have been grouped in such way to give the commodities a place of prominence, instead of the toll as in Figure 5.5. The upper chart in Figure 5.6 shows the performance of the two algorithms in the case with 132 commodity instances, whereas the lower chart shows the 180 commodity case. In both cases only the tolls, plotted on the x axis, vary.

The color coding used in this Figure remains unchanged. Unexpectedly, the three lines on both charts are located much closer one to another than in Figure 5.5. Furthermore, the results of the Best GA are virtually identical to those of the exact solver for instances with few commodities. This is a further confirmation, although not definitive, of the aforementioned

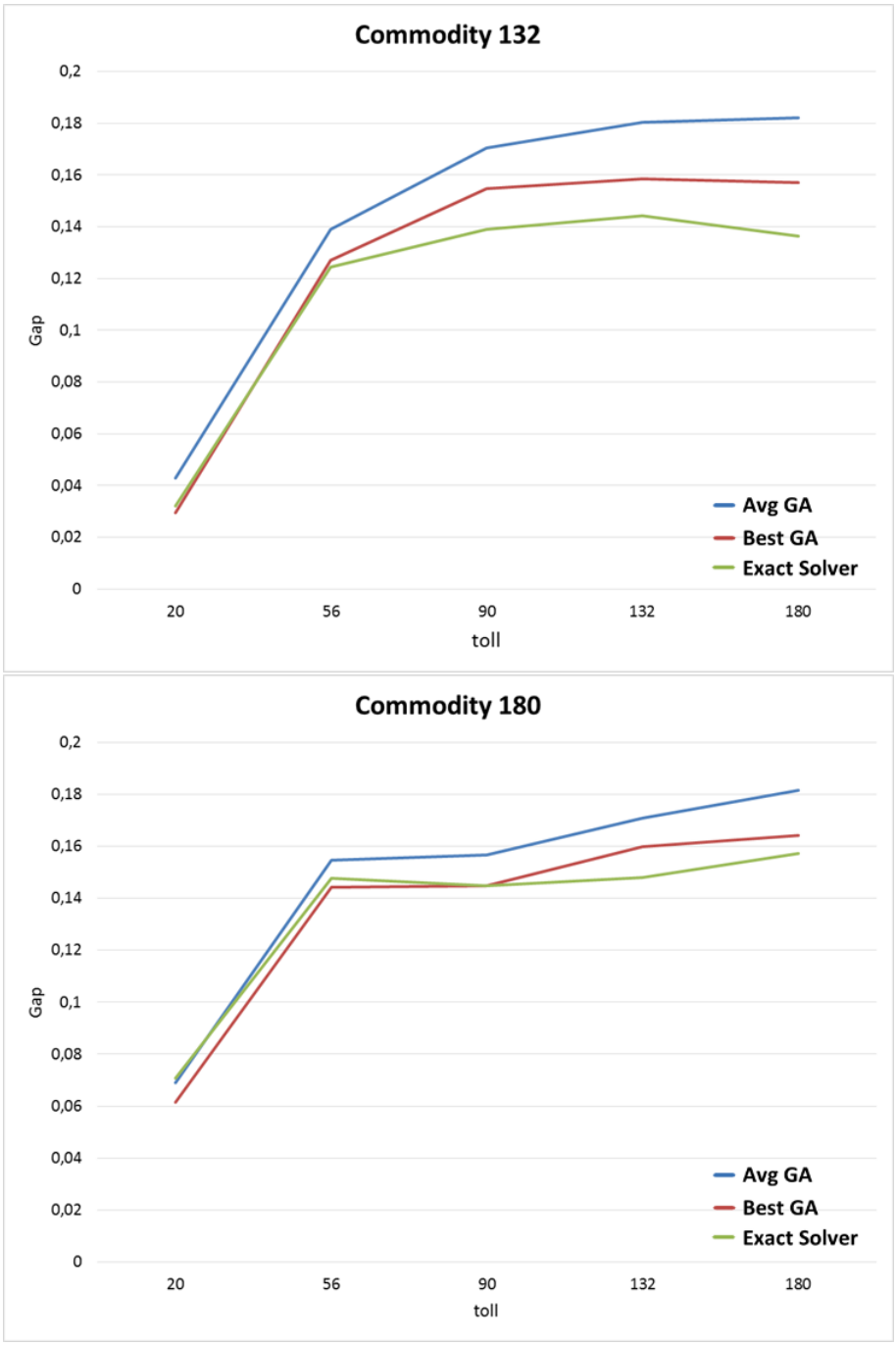


Figure 5.6: Complete NPP Randomly Generated Instances: Commodity Perspective

hypothesis that the GA performs better in the case of instances with a low number of tolls and many commodities.

5.9.2 Analysis of Partial Network Results

A similar analysis has been performed on the results of the random instances on the partial network. In some previous works, such as Violin (2014), these instances are considered harder to solve than the complete network category for exact solvers such as Xpress.

Table 5.2: Partial NPP Instances: Average Gaps and Execution Times

Partial Instances						
		Average Gaps			**Execution Times**	
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
20	20	0,1041	0,0653	0,0000	0,79	4,36
20	56	0,0579	0,0275	0,0000	5,92	32,55
20	90	0,0570	0,0318	0,0199	12,09	171,11
20	132	0,0928	0,0738	0,0748	20,79	372,80
20	180	0,1207	0,1112	0,1264	33,04	759,55
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
56	20	0,1664	0,1371	0,0000	0,83	151,48
56	56	0,1890	0,1660	0,1087	5,81	3600
56	90	0,2326	0,2147	0,1924	12,88	3600
56	132	0,3324	0,3200	0,3189	19,57	3600
56	180	0,3396	0,3283	0,3480	25,34	3600
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
90	20	0,2438	0,2134	0,0000	1,10	2610
90	56	0,2497	0,2241	0,1791	5,99	3600
90	90	0,3367	0,3210	0,2972	12,26	3600
90	132	0,3667	0,3522	0,3463	20,87	3600
90	180	0,3653	0,3569	0,3651	25,41	3600
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
132	20	0,2193	0,1913	0,0000	1,65	3600
132	56	0,3149	0,3004	0,2217	6,24	3600
132	90	0,3633	0,3435	0,3145	12,86	3600
132	132	0,3835	0,3706	0,3596	20,73	3600
132	180	0,3737	0,3642	0,3674	26,90	3600
toll	commodity	Avg GA	Best GA	Exact Solver	GA	Exact Solver
180	20	0,2566	0,2181	0,0000	2,22	3600
180	56	0,3298	0,3137	0,2620	6,30	3600
180	90	0,3674	0,3502	0,3280	13,45	3600
180	132	0,3769	0,3637	0,3551	20,92	3600
180	180	0,3877	0,3765	0,3778	28,36	3600

Figure 5.7 presents the partial network category, with all data organized in the same way as in Figure 5.5. The five charts show the results obtained by the two compared algorithms

grouped by toll. A preliminary analysis confirms that the partial category is considerably more difficult to solve than the complete category. In particular, this can be observed in the Gap relative to the Xpress solver that increases with the increase of both the number of commodities and tolls.

An analysis of the green line relative to Xpress shows that in principle the trend is very similar as in the case of the complete network but with one major difference, that is a significantly higher average Gap. A comparison of the GA performance on the partial network with respect to the complete network highlights the increasing difficulty of the problem with the increase of the number of commodities within each toll case. In fact, the GA lines (both blue and red) are steeper on this set of charts. Nevertheless, this increased problem complexity seems to have eliminated the previously observed stagnation effects that the GA endured in case of low-commodity classes. Unlike in the complete network category, the GA lines in the remaining four charts in Figure 5.7 are much less jagged. Furthermore, the initial very low performance on 20-commodity classes for all toll dimensions is absent, with the exception of the first case (20 tolls and 20 commodities).

In the partial network category the ability of the GA to catch up with the Xpress performance levels is much more visible than in the complete network. This is particularly true for instances with many commodities in all five toll cases. In the 20-Toll case the relations between the GA and Xpress have remained almost the same, but the differences in percentage among the lines has increased, especially in the 180-Commodity class. The 56-Toll chart in the upper left corner shows a steadier trend of the GA results with respect to the complete network. Furthermore, the difference between the Best GA and the Average GA is lower, which indicates a major stability of the algorithm in generating results. In the 56-Toll case with 132 commodities the results of the Best GA and Xpress are almost the same. On the other hand, both Average and Best GA clearly outperform Xpress in the 56-Toll case with 180 commodities. In the complete network such performance was achieved only by the Best GA, but the Gap percentage was rather small.

The general performance trends visible on the 90-, 132- and 180-Toll charts for partial network are very positive for the GA for the cases with the highest number of commodities in comparison to the complete network. The difference between the Best GA and Average GA is in fact very small in this case and the two lines are close to one another. The results of the Best GA and Xpress for all toll cases with 132 commodities have very similar Gaps, whereas those with 180 commodities are more favorable for the GA. In fact, in these instance Xpress is at least equaled or surpassed by either only the Best GA or even both Best and Average GA.

Given the observed improvements in favor of the GA emerging from the comparison

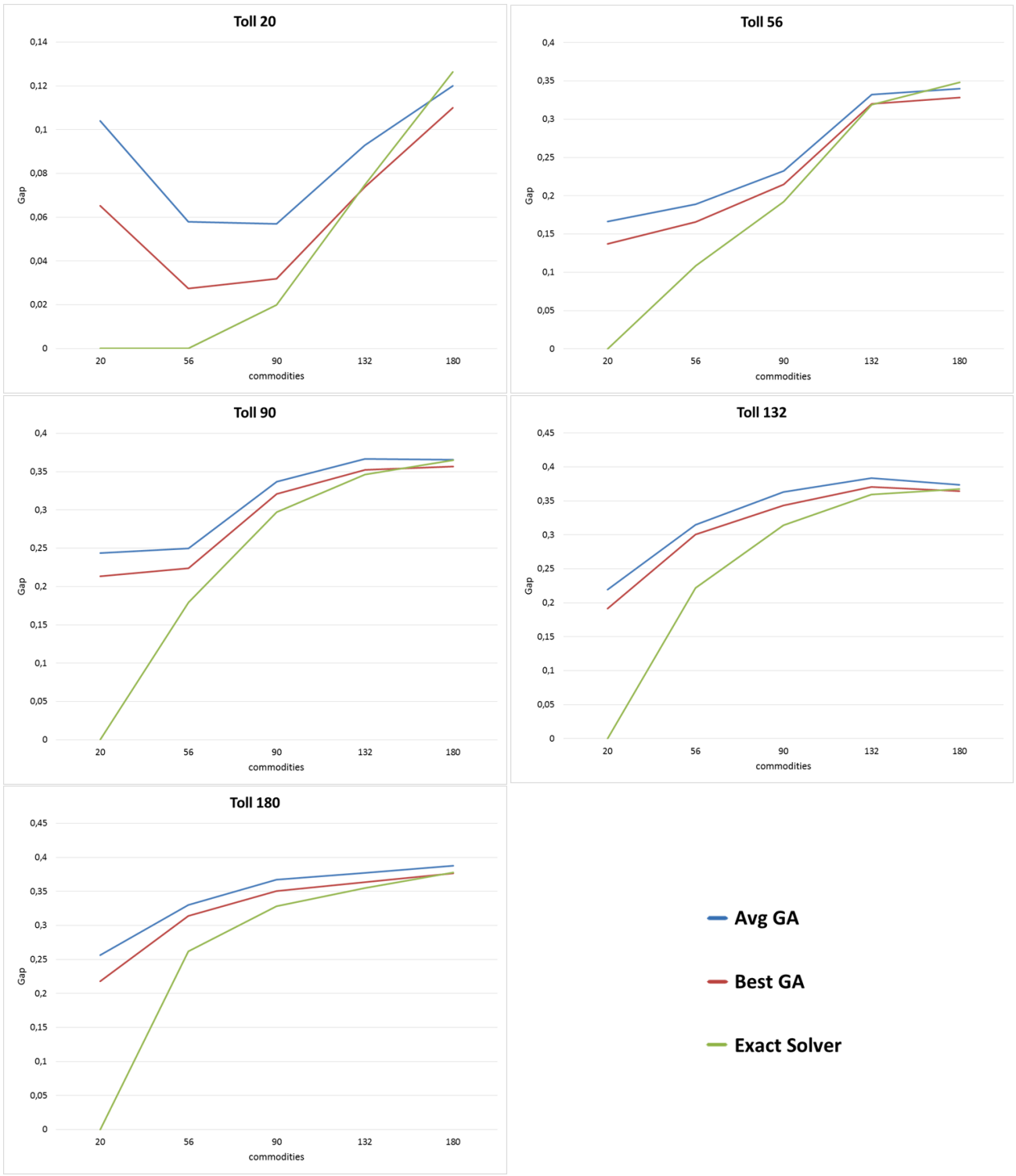


Figure 5.7: Partial NPP Randomly Generated Instances: Toll Perspective

between the Xpress exact solver and the GA in cases with a high number of commodities, an explanation approach similar to the one used to comment Figure 5.6 is adopted. Figure 5.8 is composed of two charts grouping the results according to the 132-Commodity and 180-Commodity cases. They have been plotted following the same logic and color coding as the previously mentioned chart pair. These can be considered as instances of particular interest since these partial instances with many commodities have very different Gap values for each toll class, but the overall trends shown on the two charts are very similar between the GA and Xpress. The 132-Commodity trend still presents differences, no matter how small, in favor of the exact solver. However, the 180-Commodity case for all toll classes represents the optimal set for the GA. In some of the previously analyzed charts the GA performance was better only on some instances, e.g. those with many commodities. In this particular case, however, the GA outperforms Xpress on all toll cases for the given number of commodities.

5.9.3 Execution Time Analysis

Given that the nature of the two algorithms is very different, as is their functioning, considerations should be made regarding the execution times in spite of the fact that this is generally not done in relation to GAs because their performance is usually assessed in terms of number of evaluations. Once the problem size that is most relevant for this work has been identified, a deeper analysis of this topic is necessary. Comprehensive execution times of the GA are not graphically presented in this work but only summarized in Tables 5.1–5.2. Nevertheless it is important to specify that it always takes the GA approximately the same amount of time to run 10,000 evaluations on the same instance dimension. It was observed that these small variations are strongly correlated with the number of tolls, but not with the number of commodities. The computational load at the follower level has been shown to be negligible, as stated earlier in Section 5.8.2.

Owing to this steadiness in execution time levels only the execution times for the 180-Commodity cases with all toll dimensions (plotted on the x axis) are presented in Figure 5.9. The left and the right y axes show the execution times represented as two value series expressed in seconds and refer to different elements of this combined chart. The blue vertical bars represent the average execution times of the GA for each toll class and refer to the value series plotted on the left y axis. Even in the 180-toll class, the maximum recorded time of the GA, is less than 30 seconds. The execution times of the exact solver are represented by the horizontal green line in the upper part of the chart and refer to the value series plotted on the right y axis. The execution times of the latter never vary; instead, they are fixed to 3,600 seconds. The reason is that the exact solver never managed to reach convergence

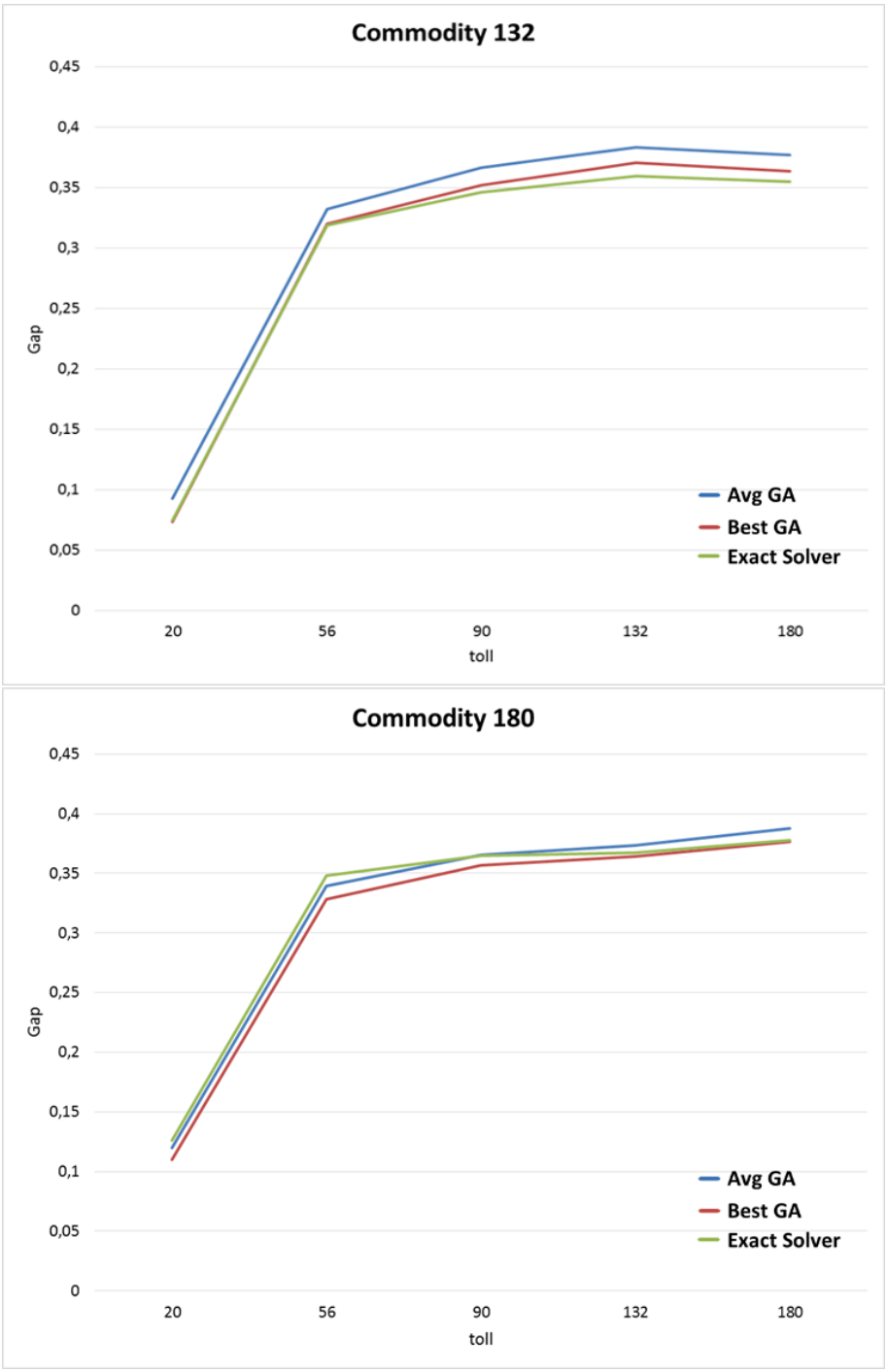


Figure 5.8: Partial NPP Randomly Generated Instances: Commodity Perspective

beforehand and thus interrupt the run, but instead the run continued until the secondary termination criterion was activated, i.e. the 1-hour time limit.

An analysis of the average execution times of the GA reveals a natural increase with the

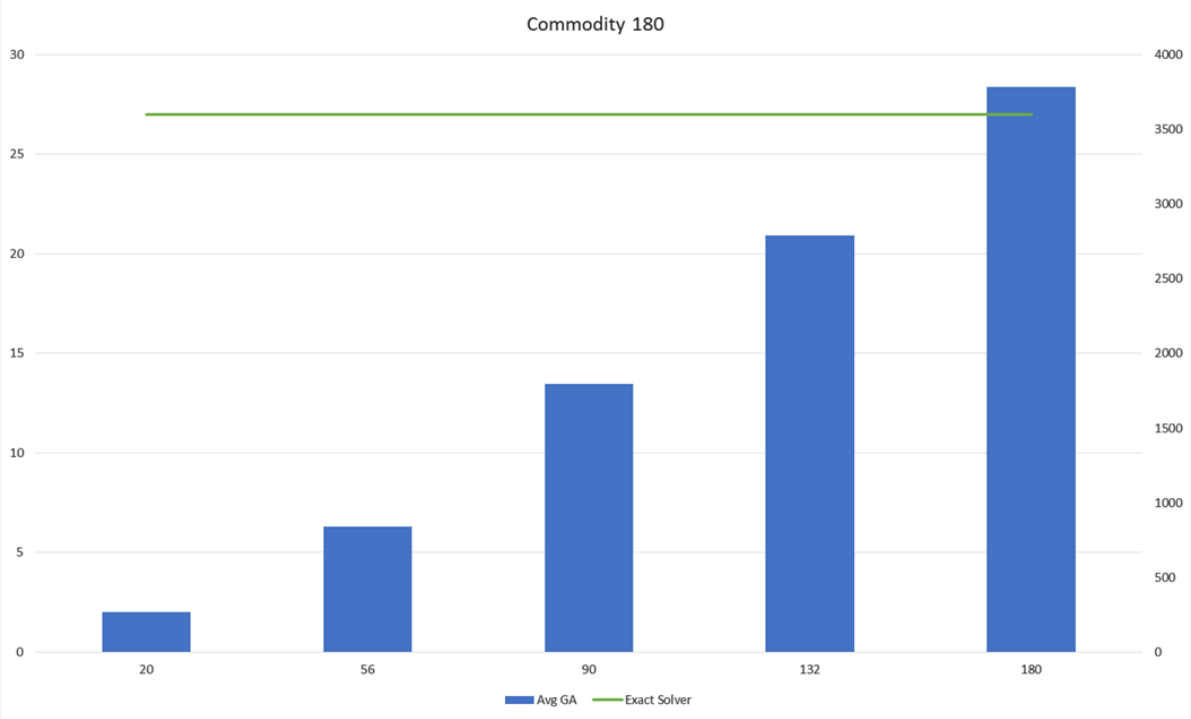


Figure 5.9: Execution times on 180-commodity partial instances

increase of the problem dimensions in terms of tolls. As said before, the analogous increase of the execution times in terms of the increase of commodities (not shown by any chart) is negligible. The absolute values of the execution times of the GA may represent a problem because the ranges of variation fluctuate between a few seconds up to approximately thirty seconds. For this reason batches of 10 repeated runs per instance and class represent a valid alternative to the exact approach, even if the available computational budget is increased. Furthermore, this strategy enables to consider the Best GA plotted with the red line on all previous charts as an effective competitor of the exact solver, considerably modifying the information extracted from the abovementioned tests. This particular fact gives further importance to the results achieved by the GA shown on the last chart in Figure 5.8. In this regard an analysis of Xpress time performance would not yield any real benefits because the obtained results are equal or worse then those of the GA, even with the assumption that the Xpress's convergence curve was very fast (i.e. few seconds). Whether Xpress reaches this result immediately and then stagnates or at the last second is irrelevant because such a scenario is not usable in practice. However, considering that the execution time differences are deemed rather high, a further analysis of execution times only is required (Section 5.9.4).

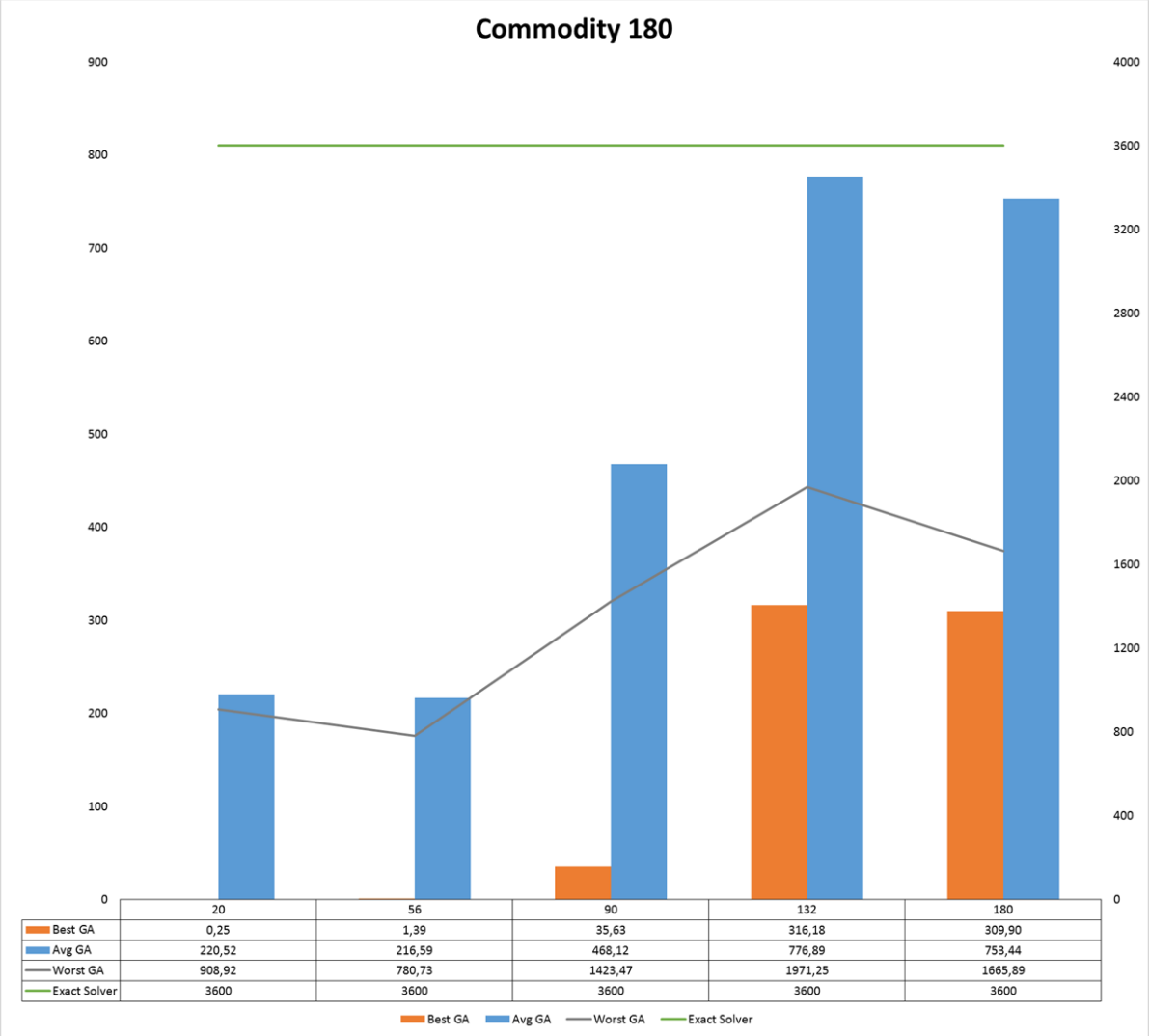
5.9.4 Time To Target Experiments

The goal of this Section is to perform a more comprehensive and fair comparison between the execution times of the two algorithms. In order to do that the result quality is fixed and used as a target to be reached. For this purpose the Time-to-Target approach was used. According to this approach, the focus is entirely on time so a defined objective function value is determined as the target to be reached by the algorithms and interrupt their runs when they do. After that the time it took them to reach this target value is measured.

In this case the analysis concerned the partial network 180-Commodity instances for 20, 56, 90, 132 and 180 tolls discussed in the previous Section. These are randomly generated instances so the optimal target values are not available. To overcome this particular problem and avoid an endless execution from the Xpress exact solver, the target for the GA became the objective value found by Xpress after a run of 3,600 seconds. A further safety termination criterion has been configured for the GA, i.e. maximum number of evaluations that it can perform. A limit of 100,000 evaluations was considered appropriate.

The results were obtained for five toll classes of partial networks and a fixed number of commodities (180), and for each of them 10 randomly generated instances were run. The use of such batch runs was necessary to obtain average results due to the GA stochastic nature. The results are presented in Figure 5.10. The complexity of plotted data, reflected mostly in different orders of magnitude of value series, required again the use of a combined chart. The average execution times of each batch of ten instances relative to the Best GA and the Average GA are plotted with orange and blue vertical bars respectively. The average execution times of the Worst GA and the reference time of 3,600 seconds are plotted with the gray and the green line respectively. As in the previous chart, two different execution times value series (in seconds) are shown on the left and on the right y axis: the former are referred to in relation to vertical bars, whereas the latter are referred to in relation to lines. The five toll classes, plotted on the x axis, also represent the header of the table below the chart which contains the exact values plotted on the chart. This specification was deemed necessary because of large differences in the order of magnitude of the plotted results, visible in particular in the absence of the first two orange bars due to their very small values.

Before the actual analysis of the results plotted in Figure 5.10 a last consideration should be made. The secondary termination criterion added to the GA beside convergence to target value was triggered on certain instances, meaning that the execution was interrupted before reaching the desired objective value. Indeed, this information would have a considerable impact on the interpretation of results if the percentage of failure to reach the target were high. For this reason it was deemed important for the sake of impartiality to report the



	20	56	90	132	180
Best GA	0,25	1,39	35,63	316,18	309,90
Avg GA	220,52	216,59	468,12	776,89	753,44
Worst GA	908,92	780,73	1423,47	1971,25	1665,89
Exact Solver	3600	3600	3600	3600	3600

Figure 5.10: TTT: Execution times analysis on 180-commodity partial instances

average success rates relative to all instances (see Table 5.3). These rates fluctuate between 75% up to a maximum of 86%, meaning that the GA was not able to hit the target in all runs within the 100,000 evaluation limit. However, it also means that roughly only one out of five runs was unsuccessful. Returning to Figure 5.10 this fact should explain the unusual relation among the average execution times of Best GA, Average GA and Worst GA runs.

Considering the success rates, it is certain that in each batch of tests for each toll class there is at least one run that failed to reach the target, which automatically determines the execution time of the Worst GA. Such runs continued until the exhaustion of the maximum evaluation budget and thus lasted much longer than the runs that hit the target within that limit, also causing an increase of the execution times of the Average GA. However, for

Commodity	Toll	Avg Hits/Tries
180	20	0,75 %
180	56	0,86 %
180	90	0,82 %
180	132	0,80 %
180	180	0,82 %

Table 5.3: TTT: Average Hits on total Tries in 180-commodity instances

consistency reasons the latter has not been excluded from the analysis. In the light of the above considerations it is possible to state that the application of the proposed methodology based on the MOGASI nested approach can yield results of the same quality levels as those obtained with the Xpress exact solver in at least one order of magnitude of time less. However, this performance cannot be guaranteed in a 100% of cases, but with a certain probability which, although high, requires a few repeated runs of the same instance. Considering that the average execution time of the case with the longest run duration is approximately $1/5$ of the time required by Xpress, this effort is deemed more that acceptable. If this approach should really be performed with repeated runs, a reduction of the maximum number of evaluations is advisable in order to prematurely interrupt those runs which got stuck anyway in a local optimum in the objective landscape. In this way, a lower maximum time allowed for the repeated runs would be guaranteed against a likely invariability of the success probabilities.

CHAPTER 6

CENTRAL PEAK LOAD PRICING PROBLEM

Air Traffic Management (ATM) in Europe is a considerably complex system, with many agents involved. The most relevant entities for this work are Eurocontrol, the *Air Navigation Service Providers* (ANSPs) and the airlines, i.e. *Airspace Users* (AUs).

The *European Organisation for the Safety of Air Navigation* (Eurocontrol) is a civil international organization which coordinates and plans air traffic control for all of Europe and as such has the role of Network Manager. It has 41 member states (plus the European Union, which is also a member) out of 44 belonging to the *European Civil Aviation Conference* (ECAC) area, as shown in Figures 6.1 and 6.2.

Among the many functional units that comprise this organization, two are particularly relevant for the purpose of this work: the *Central Flow Management Unit* (CFMU), which manages the air traffic flows of the ECAC states to ensure that the demand from the airlines does not overload the capacities of the available infrastructure and the *Central Route Charges Office* (CRCO) which charges airspace users for providing ATM services on behalf of the Member States through the route charges system.

Air Navigation Service Providers (ANSPs) are agencies that manage flight traffic at regional or national level. ANSPs are commonly administered as a government department depending on the state budget as autonomous bodies belonging to the public sector (still property of the state but separate from it) or as fully or partially privatized agencies. In any case, each European state is responsible for providing air traffic services as a public service and has sovereignty over the airspace above its territories. ANSPs are mainly funded through the air navigation charges collected by Eurocontrol from the airspace users. The charges are

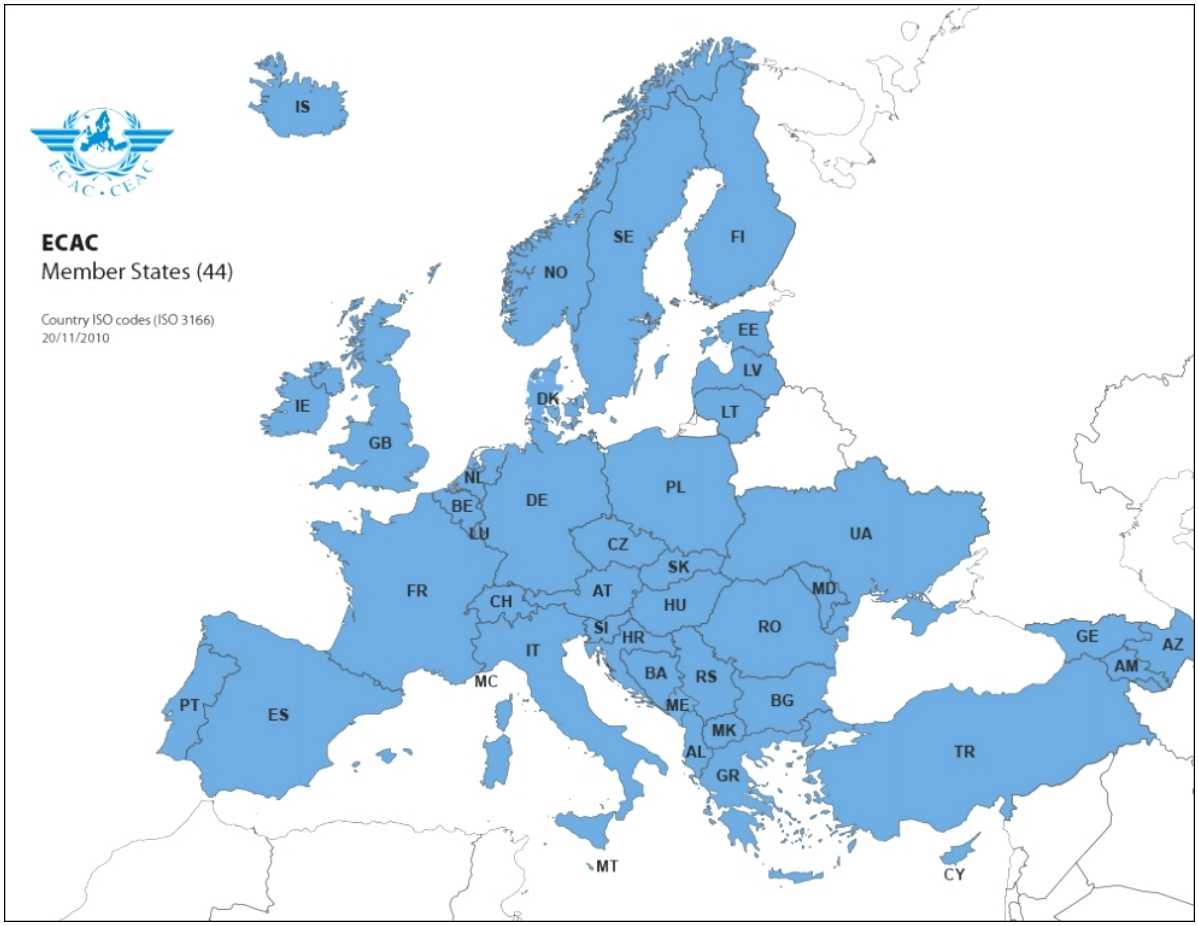


Figure 6.1: ECAC member states (source: Eurocontrol)

set and collected according to one of the following two principles (Eurocontrol, 2015a):

- *Full cost Recovery*: route charges are calculated based on the estimated costs and traffic for that same year. An adjustment mechanism is applied to ensure that only the actual costs of the service are eventually recovered.
- *Determined costs*: the costs are determined by the contracting states at the level of the charging zone and are estimated prior to the beginning of each reference period (which covers from three to five years) as part of the performance plan for each calendar year.

Airline operators are the final users of the ATM system, AU in short. According to Eurocontrol, every year more than nine million *Instrument Flight Rules* (IFR [1]) flights are performed by

[1] A pilot planning a flight can choose between two types of flight rules: *Visual Flight Rules* (VFR) and *Instrument Flight Rules* (IFR). Under VFR the pilot is responsible for maintaining visual contact with other airspace users and determining his route with the help of geographical landmarks. Under IFR the pilot uses

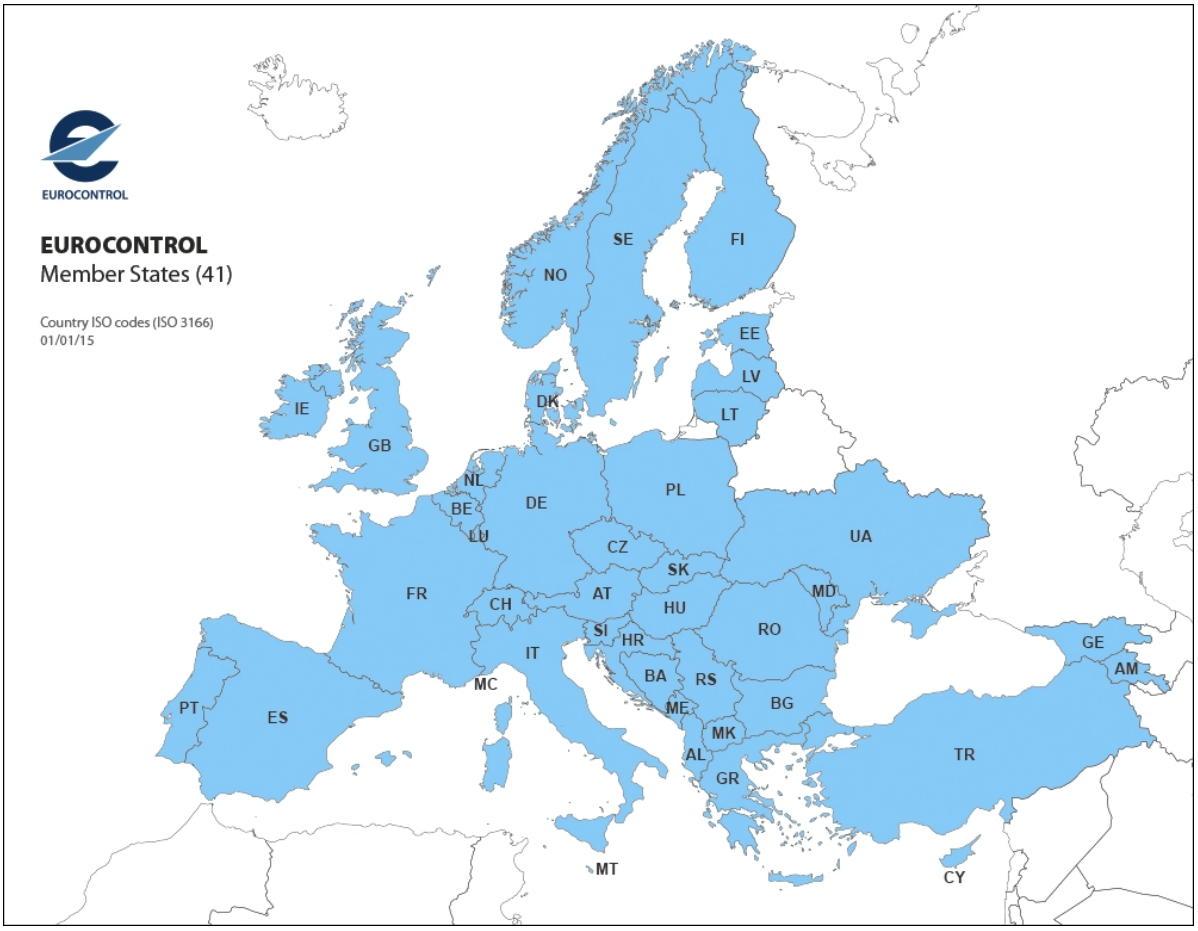


Figure 6.2: Eurocontrol member states (source: Eurocontrol)

scheduled and business airlines in the ECAC region, corresponding to an average of over 28 thousands flights per day with an average duration of one and half hour each (Eurocontrol, 2015d). For the purpose of this work only IFR flights are considered, both passenger and cargo. In the passenger segment there are currently over 50 AUs that move annually millions of passengers to, from and within European countries, as shown in Table 6.1.

Breaking down the operational costs of airline operators to assess their potential sensitivity to the modulation of air navigation charges is no easy task. In fact, AUs are highly heterogeneous with regards to market scope (e.g. regional carrier, international carrier, pure passenger carrier, pure cargo carrier, mixed passenger and cargo etc.) and business models (e.g. traditional full service carrier, low cost carrier, charter etc.). Estimates can be inferred from the annual investor reports that the AUs publish every year and are freely accessible on

the on-board instruments as navigation aids which, together with the indications transmitted by air traffic controllers, compensate the lack of visibility and provide the information required for the pilot to be aware of his position at all times (Cook, 2007).

Table 6.1: Largest airlines in Europe by total passengers carried in millions

Rank	Country	Airline/Holding	2015	2010	2005	Fleet	Destinations
1	Germany	Lufthansa Group	107,7	90,2	51,3	602	321
2	Republic of Ireland	Ryanair	101,4	72,7	33,4	319	186
3	United Kingdom/Spain	International Airlines Group	88,3	50,6	-	520	248
4	France/Netherlands	Air France-KLM	79	70,8	70	574	231
5	United	Kingdom easyJet	69,9	49,7	30,3	226	136
6	Turkey	Turkish Airlines	61,2	29,1	14,1	300	280
7	Russia	Aeroflot Group	39,4	14,1	8,1	239	189
8	Germany	Air Berlin Group	30,2	31,8	-	143	145
9	Sweden/Norway/Denmark	SAS Group	30,2	23,1	26,5	137	157
10	Norway	Norwegian Air Shuttle ASA	25,8	13	3,3	102	130

the companies' websites. For example, for Ryanair as the largest low-cost carrier in Europe the cost of fuel and oil account for 46.03%, whereas route charges account for 12% of total operating costs. An example of a traditional passenger carrier is Lufthansa: although this company does not provide separate figures for airport and route charges, the combined value amounts to 23%. Airport and route charges for Lufthansa Cargo as the largest cargo carrier in Europe account for approximately 12%. The Cargo flights are mostly performed during the night since airport charges are generally lower during those hours, it can be safely assumed that in this case route charges make up for the larger part of these costs. According to Eurocontrol (2015d), traditional, low-cost and pure cargo airlines account for 83.6% of flights and 86.3% of the collected charges.

6.1 Air Navigation Service Charges

The costs of ATM services in Europe (infrastructure, staff and other operational costs) are funded through air navigation charges. In particular, they are funded through the "user pays" principle, meaning that the AU are directly charged for the operated traffic. There are different sorts of air navigation charges: route charges, terminal navigation charges, and communication charges. *Air Navigation Services* (ANS) charges are imposed to recover the cost of providing ANS in three phases of flight: movements at and around airports (aerodrome control), approach and departure of flights, including initial climb and descent (approach control) and en-route phase (en-route/area control). Air navigation services provision in Eurocontrol member states is based on the principle of recovering an a-priori determined costs for a reference period (the so-called determined cost system, mandatory for the signatories of the *Single European Sky* (SES)[2] Initiative) or the full cost recovery

[2]Single European Sky (SES) is the legislative framework for European aviation constituted by the European Commission as a response to the growth in air travel witnessed in the last two decades. The main objective was

system (applied by the nine Eurocontrol member states which are not SES signatories).

ANS charges play a pivotal role in the economics of the European ATM industry: en-route charges represent 76% of all ANSP revenues (Performance Review Unit with ACE Working Group, 2014), with the remaining 14% accounting for terminal charges. ANS charges are a non-negligible operational cost for AUs, especially when fuel costs are low. For these reasons, understanding how much AU route choices depend on ANS charges (en-route charges in particular) and to what extent the charges could then be used as an effective tool to balance demand and capacity is of great importance.

Route charges represent the remuneration for the costs of en-route ANS provision, including Eurocontrol costs. There is a harmonized route charging system in the Eurocontrol area, the legal basis of which is the Multilateral Agreement relating to Route Charges. This Agreement dates back to 1981 and has set the basis for the common policy on ANS route charges in the Eurocontrol area. Regulation EC 1794/2006 (European Commission, 2006) laid down a common charging scheme for ANS services and introduced the notion of charging zones. An "en-route charging zone" is a volume of airspace for which the states establish a single cost base and a single unit rate. This "en-route charging" zone extends from the ground up to and including upper airspace. A Contracting State is permitted to establish a specific zone for a complex terminal area (a Terminal Maneuvering Area, TMA, which is a special type of airspace located above an airport designed to handle aircraft arriving and departing the airport(s) contained within it) after consultation with airspace user representatives. En-route charging zones are listed in Annex 1 of (Eurocontrol, 2011).

A single en-route charge is levied for each flight performed in the Eurocontrol airspace, regardless of the number of Member States overflown. The system for billing and collection of route charges is one and operated by Eurocontrol's Central Route Charging Office (CRCO). The billing of route charges is done on a monthly basis and charges income is disbursed weekly to the ANSPs. In principle, all flights are subject to route charges. However, there are several categories of flights exempted from payment in all Eurocontrol states (European Commission, 2006), namely:

- flights performed by aircraft with maximum takeoff weight inferior to 2t.
- mixed VFR/IFR flights in the charging zones where they are performed exclusively under VFR and where a charge is not levied for VFR flights.

to reform ATM in Europe to ensure a sustainable air traffic growth and operations under the safest, most cost- and flight-efficient and environmentally friendly conditions. The SES legislative framework consists of four Basic EC Regulations (the SES I Package, n. 549/2004, 550/2004, 551/2004 and 552/2004), which were revised and extended in 2009 with Regulation (EC) n. 1070/2009 which became the SES II Package.

- state flights performed exclusively for the transport of reigning monarchs, heads of state, government or ministers on official mission.
- search and rescue flights authorized by the competent body.

In addition, some states may exempt from payment of route charges all military flights, training flights performed for the purpose of obtaining a license or testing equipment, circular flights, humanitarian flights and flights performed by customs or police officers. Where exemption is granted, the state concerned bears the cost which would otherwise be chargeable to the flights.
The total charge per flight collected by Eurocontrol (R) equals the sum of the charges generated in the charging zones defined by states:

$$R = \sum_i r_i \tag{6.1}$$

The individual charge (r_i) is equal to the product of the unit rate (t_i) and the number of service units (s_i) in charging zone i for this flight:

$$r_i = t_i \cdot s_i \tag{6.2}$$

The number of service units (s_i) is defined as a product of the distance factor (d_i) and the weight factor (p) for a given flight:

$$s_i = d_i \cdot p \tag{6.3}$$

The distance factor, d_i, is equal to 1/100 of the great circle distance (expressed in km) between the aerodrome of departure within, or the point of entry into, the charging zone (i), and the aerodrome of first destination within, or the point of exit from, that charging zone. The distance to be taken into account is reduced by 20 km for each take-off and for each landing within a charging zone (i). The entry and exit points are the points at which the lateral limits of the charging zone are crossed by the route described in the flight plan filed by the operator and approved by the *Central Flow Management Unit* (CFMU) 30 minutes prior to take-off. The route description per flight is extracted from the flight plan filed by the operator and approved by the CFMU 30 minutes prior to take-off. This enables the *Central Route Charges Office* (CRCO) to calculate the distances flown in each State's airspace (Eurocontrol, 2014a).

The weight factor, p, is proportional to the *maximum takeoff weight* of the aircraft used and it is calculated according to following formula:

$$p = \sqrt{MTOW/50} \tag{6.4}$$

where the MTOW is the maximum takeoff weight of the aircraft, expressed in metric tonnes and rounded to one decimal place. MTOW values are calculated as follows: towards the end of each calendar year, AUs are required to submit to the CRCO a declaration of the composition of their fleet (registration markings, aircraft type, version and certified MTOW). Based on this information, the CRCO calculates the weight factor based on average weight of all aircraft of a basic type. These weight factors then become valid from the 1st January of the following year.

Unit rates (t) are calculated based on required equality of total costs and revenues of an ANSP, as a ratio of cost base (forecast or determined) and forecast number of service units. The cost base includes operating costs, depreciation costs, cost of capital, and a state share of Eurocontrol costs. Each Eurocontrol Member State establishes the unit rate of en-route charges (basic unit rate) for the airspace within its responsibility. Each November the enlarged Commission approves the basic unit rates for the following year. Basic unit rates are adjusted every month if the national currency of a Member State is not the Euro. The monthly unit rate is recalculated by applying an exchange rate between the Euro and the national currency. This exchange rate is the average of the Closing Cross Rate calculated by Reuters based on the daily Bid rate, for the preceding month.

Terminal changes are used to remunerate the cost of aerodrome control services and air traffic services related to the approach and departure of an aircraft within a certain distance (at present typically 20 km) of an airport. Terminal charges are not the subject of this work.

6.2 Air Traffic - Current Situation and Growth Forecasts

In 2014 the European ATM system controlled on average 28,000 flights daily; after the decrease between 2011 and 2013, flights in Europe increased again by 1.7% in 2014 with a positive medium term outlook. According to the latest Eurocontrol Seven Year Forecast (Eurocontrol, 2015b), flights are expected to grow by 1.5% in 2015 and to continue with an annual average growth rate of 2.5% between 2014 and 2021. Currently, European ATM costs an additional 2-3 billion euros every year compared to other similar systems in the world due to the inefficiencies in controlling a highly fragmented airspace with an uneven

distribution of traffic. In fact five largest ANSPs (DFS for Germany, DSNA for France, ENAIRE for Spain, ENAV for Italy and NATS for the UK) bear 60% of total European gate-to-gate service provision costs and operate 54% of European traffic.

According to Eurocontrol's Performance Review Report for the year 2014 (Eurocontrol, 2015c), after a steady improvement between 2010 and 2013, when the lowest level of en-route delay per flight on record was reached (0.53 minutes per flight), en-route delays in the Eurocontrol area increased again to 0.61 minutes per flight in 2014 (as shown in Figure 6.3). The performance deterioration in 2014 was mainly attributed to ATC capacity issues: while capacity constraints can occur from time to time, some area control centers generate high delays on a regular basis. In 2014 they were Warsaw, Lisbon, Canarias, Athinai/Macedonia, Reims, Brest and Marseille, accounting together for 54.6% of all such delays, but only for 17.8% of total controlled flight hours.

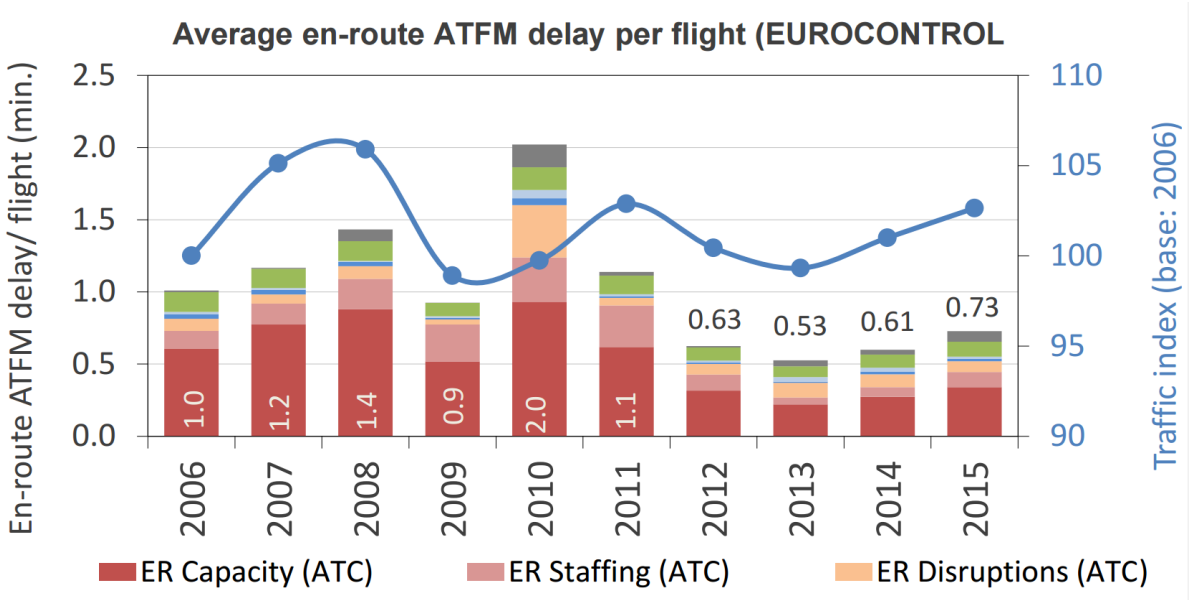


Figure 6.3: Average en-route ATFM delay per flight in the Eurocontrol area between 2006 and 2015 (source: Eurocontrol)

According to Eurocontrol forecasts (Eurocontrol, 2014b) for the period 2014-2019 at European network level the delay forecast based on the baseline traffic growth scenario shows that en-route delay will be above the target set at 0.5 minutes per flight, for each year of this period. The delay forecast is based on capacity plans agreed with all ANSPs during the period November 2013-April 2014 and on the baseline scenario of the 2014 Eurocontrol Seven Year Forecast of traffic demand (Eurocontrol, 2015b). Delays are likely to be above the target at some ACCs in Europe throughout the period in Cyprus, France, Greece, Poland,

Portugal and Spain. In most cases this is due to inflexible use of staff, shortage of qualified controllers in some areas and unresolved staff management issues despite the plans for capacity increase formulated in response to the capacity requirement profiles 2014/2019.

Eurocontrol's Challenges of Growth studies have been developed to deliver information to support long-term planning decisions, analyzing what are the challenges of growth for commercial aviation in Europe between now and both 2035 and 2050. According to the latest of these studies, completed in 2014, the five main challenges identified for 2035 are as follows:

- The continuing difficulty of delivering airport capacity when, where and at the price it is needed;
- The difficulty of delivering the required level of performance on a congested network, when airport delay increases on an average busy day by a factor of 5 or 6 to become a frequent major contributor to overall delay;
- Keeping the industry financially viable in an era of slower growth and more interesting opportunities for investors away from Europe where aviation will be growing more quickly;
- The amount of emissions from aviation, which are likely to increase even with a slow traffic growth. Development of competitively-priced low-carbon fuels may become a priority in this regard;
- Building resilience to climate change. A growing number of organizations are making resilience to climate change a routine part of their business or operational planning. However, more should be done to build local and network climate resilience. Some of the solutions are relatively low-cost (training and procedures), or happy side-effects of other investments. An early start should save money in the long run.

Different forecast scenarios have been devised on how air transport in Europe and the factors influencing it might develop. The most likely scenario envisages moderate economic growth with regulations reconciling the environmental, social and economic demands to address the growing global sustainability concerns. According to this scenario, there will be 14.4 million flights in Europe in 2035, 1.5 times the 2012 volume. That is an average of 1.8% increase per year, around half the historic rate from the 1960s to the peak of 2008. Traffic growth will slow down from 2025 as markets mature, economic growth decelerates and as the capacity limits at airports increasingly become an issue.

6.3 Pricing in Network-based industries

Network industries are generally priced according to the models and methods of the market for services. The market is generally composed of several actors, namely an infrastructure manager/owner, who may also act as a regulator, one or more service providers and consumers. The objective of the regulator is generally to maximize network efficiency, whereas the objective of service providers is to maximize revenues while minimizing costs. The objective of the consumers is to obtain the service at minimum cost.

When dealing with networks, efficiency and costs are generally related to the level of congestion of the network itself. Congestion is a negative externality representing an imbalance between demand and supply (that is, the available network capacity) which deteriorates the quality of the service due to delays and results in additional costs or non-provision of service.

Since users want to minimize their cost for using the network (in terms of time and money), and congestion generates externalities, marginal cost pricing is generally regarded as the way to internalize the cost of congestion. Marginal cost is the variation in the total cost that arises when the quantity produced is incremented by one unit, that is, it is the cost of producing one more unit of a good. In network terms, it represents the cost of providing an additional unit of capacity on a link or path or, equivalently, the externality incurred to each user of the network.

The so-called first-best pricing principle states that a toll equal to the user's externality (a Pigovian tax) should be charged on each link in order to obtain the optimal network traffic flow configuration (De Palma and Lindsey, 2011). This scheme represents theoretical optimality, where marginal revenues (revenue obtained by the service provider from each user) equal marginal costs.

However, estimating marginal costs for actual implementation is, in general, difficult. Required information includes customers' demand elasticity and cross demand. Customers are usually very reluctant to reveal their willingness to pay (which is, in general, higher when demand elasticity is low), as it is subject to strategic behavior. Moreover, if marginal costs are considered only in short term, they do not cover the costs of upgrading the infrastructure, potentially affecting the development of the industry (deficit). As a consequence, optimal marginal cost pricing schemes are rarely implementable in reality and second-best pricing regimes (i.e., models that deliver a solution that is sub-optimal but workable in reality) are generally preferred for pricing actual networks.

The theory of second best (Lipsey and Lancaster, 1956) in general states that in a system

where conditions are such that a Pareto optimum exists, if one condition is changed so that it is no longer at its optimum state (or if one or more condition are subject to uncertainty), a second best optimum (because the first best optimum cannot be reached) cannot be obtained by simply re-evaluating the uncertain condition and leaving the others to their first-best values. Instead, the theory of second best states that all the other conditions must be changed from their original first best optimum states. In general, the direction and magnitude of the changes necessary are not known.

An example of a second-best pricing scheme is Ramsey Pricing (RP). The aim of RP is deficit coverage, obtained by means of pricing markups to be added to marginal costs proportionally to customers' elasticities; the price markup should be inverse to the price elasticity of demand: the more elastic demand for the product, the smaller the price markup. This markup, generally a percentage on the marginal costs, is inversely proportional to the price elasticity of the demand of the customers at zero profit (inverse elasticity rule). RP is in general hard to implement: since it builds upon marginal costs, and therefore faces the same information restraints.

Another common second-best pricing scheme is Peak-load pricing (PLP), typically applied in utilities and public transports. In PLP users are charged for marginal and capacity costs in an environment where demand peaks (and therefore capacity shortages) are easy to predict. The resulting pricing scheme is usually divided in fixed time periods priced differently, namely off-peak, peak time and (possibly) shoulder periods.

It is generally argued that all types of congestion charging schemes are, in fact, discriminatory since they tend to penalize users with lower incomes. Hence, non-monetary pricing (NMP) schemes have been proposed as an alternative to first-best and second-best charging schemes in order to grant equal rights to all users. NMP schemes generally charge a certain amount of freely distributed credits or travel permits for traveling during peak times. The equity issue is then transferred to the initial endowment of credits or permits among users. An exhaustive overview of non-monetary pricing schemes proposed for road transport can be found in Fan and Jiang (2013).

6.4 Applicability of other pricing schemes to European ATM

In the current configuration the European airspace has a single Network Manager (Eurocontrol), who is responsible for collecting en-route charges and redistributing them to national ANSP. Charges are set by national ANSPs with the aim of recovering operational costs of providing air navigation services to the AUs. From the economic point of view, the current

environment is a monopolistic competition, where competitors (ANSPs) are differentiated on a location basis (country boundaries) and competitors' pricing policies are not taken into account. The configuration is summarized in Table 6.2 (Rigonat, 2016).

Environment	
1. Control	b. Decentralized control: unit rates are set by ANSPs and collected by Eurocontrol.
2. Pricing strategy objective	a. Objective of cost recovery: en-route charges are collected to recover operational costs of national ANSP for ATC services.
Pricing	
3. Type of tariff	c. Consumption-proportional: monthly adjusted unit rates.
4. Modulation of the tariff	c. Space dependent and time independent: unit rates vary on a country bases (although EU reg. 391/2013 art. 16 allows unit rate modulation).
5. Users classification	a. No differentiation among customers: all airlines are equal.
6. Price setting strategy	b. Prices are set according to resource value: cost of ATC services.
7. Payment	a. Monetary payment.
8. Quality of Service (QoS)	a. and b. Guaranteed service, capped by ATC sector capacity by imposing ATC delay (but this is applied on Day of Operation).

Table 6.2: Current configuration of the route charges system (Rigonat, 2016)

An alternative scenario is an environment in which prices are set and controlled by a single central authority whose objectives are the recovery of ANS expenses and reduction of network congestion. This alternative scenario is introduced as background for the centralized pricing models described in Section 6.6. This is a monopolistic environment with a Network Manager, ANSPs as operators and AUs as customers (Table 6.3) (Rigonat, 2016).

Flat pricing and user-class based options should be excluded a priori: the former because it is not an incentive sustainable traffic distribution, the latter because it clashes with the requirement of equity among AUs stated by EU Reg. 391/2013 art. 16: *Member States, [...] may, at national or functional airspace block level and on a non-discriminatory and transparent basis, modulate air navigation charges. Since pricing is considered at a strategic level, real-time pricing should also be excluded.*

Time dependent usage pricing (see Falkner et al. (2000)) and Time of Usage (see Borenstein et al. (2002)) are of the peak-load pricing type. Users are charged proportionally to resource consumption and the tariff varies according to the time at which the service is provided. Tariffs are set according to congestion level forecasts obtained from historical data and are adjusted periodically (e.g., once a month). There are no remarkable issues in making such a pricing scheme also location dependent, as proved by other transport modes that use time-and-place dependent peak-load pricing, so it is a viable option for European airspace

Environment	
1. Control	a. Centralized control: tariffs will be set and collected by a Central Planner.
2. Pricing strategy objective	c. Objective of cost recovery and congestion reduction.
Pricing	
3. Type of tariff	b. or c. Proportional to traveled distance, sector forecast capacity or both; either an exact marginal cost or a second best charging scheme is plausible.
4. Modulation of the tariff	c. or d. Always space dependent, could be either time dependent or invariant.
5. Users classification	a. Equity is a priority; hence, user classes among airlines would not be welcomed.
6. Price setting strategy	b. Prices are set according to resource value: cost of ATC services; a combination of resource and user-perceived value is also acceptable.
7. Payment	a. or c. Payment could be either monetary or hybrid.
8. Quality of Service (QoS)	a. and b. Guaranteed service, capped by ATC sector capacity (this could be applied in advance of Day of Operation).

Table 6.3: Centralized control with modulated charges configuration (Rigonat, 2016)

as well. De Matos (2001) investigates the issues and applicability of peak-load pricing and yield management techniques to European airspace.

Ramsey pricing based mechanisms, such as responsive pricing for telecommunications (Falkner et al., 2000), and simple charges for rail transport (Peter, 2003) set the congestion charge, or the congestion dependent component of the tariff, as inversely proportional to demand elasticity of the users. Critical peak pricing for electricity retail (Borenstein et al., 2002) is conceptually similar but maximum price for peak times is generally capped. It is legitimate to assume that demand elasticity varies among users also in the air transport industry. For example, an airline operating on a hub-and-spoke paradigm is likely to have a less elastic demand than one operating on a point-to-point basis due to the constraints imposed by connecting flights.

Critical Peak pricing with Incentive Rebates and Demand Reduction Programs (both from electricity retail, see Borenstein et al. (2002)) are variants of the Ramsey pricing scheme where the user pays a baseline charge and is offered an incentive for not consuming during peaks and/or cash-back for consuming less than what is provided by the baseline. From a model point of view, they are equivalent to Critical Peak pricing, the only difference lying in the way the scheme is presented in the contract to the customer. For this reason, it is reasonable to discard them from our analysis.

Time of Usage with demand charges (electricity retail, Borenstein et al. (2002)) is a peak-load pricing scheme where the consumer is charged an additional tax for his own peak usage. The tax is independent of the level of network congestion at that time. Since generation costs in the electricity industry raise exponentially when imbalances between demand and supply occur, such a scheme is designed to deter the customer from generating peaks of usage at all. This type of pricing is therefore tightly related to a peculiar characteristic of electric grids that is not shared by airspace and therefore has no meaningful applicability in air traffic.

Bid-price and auction-based mechanisms, such as smart-market pricing for telecommunications (Falkner et al., 2000) and pay-as-bid pricing for electricity wholesale may give rise to equity issues, since they favor economically sounder AUs that are likely to pay more. However, should the auctioning process be carried out without the use of real money, but rather with freely distributed non-monetary credit, it would be acceptable and still effective for reducing congestion. Such a system could be combined with the current distance-based en-route charges so that it would also guarantee ANS operational costs recovery.

As for other non-monetary schemes, both day permits and vehicle miles traveled (Fan and Jiang, 2013) are not suitable for managing congestion in air transport. Day permits would constrain AUs in scheduling flights on allowed days only; such a limitation would be unwelcome by both AUs and their customers, and would raise severe issues on equity. Vehicle miles traveled is a fair option for reducing pollution through emission charges; in fact it shares some traits with the already implemented EU Emissions Trading Scheme (ETS), whose aim is to mitigate the climate impacts of aviation. Such a system, however, charging only on traveled distance (as with current ANS charges), is ineffective for managing congestion.

The permit system that better fits the needs of air traffic congestion management is time-place specific allowances (Fan and Jiang, 2013). Under such schemes, the capacity of a resource determines the total quota of available permits. Since permits are defined for use in a specific period, their validity in time is also clearly defined and limited. In order to enforce capacity constraints for air routes or sectors, it is therefore sufficient to issue a limited number of permits for each resource and time period and distributing them with a strategy that grants equity among AUs (e.g., first-come-first-served or auctioning with credits). Credit allowances (Fan and Jiang, 2013), represent a very flexible option for dealing with demand/capacity imbalances in air traffic. Credits may be used in different ways, as a currency for congestion charges or auctions for resources, as previously suggested for bid-price and auction-based mechanisms. Provided that the initial endowment is fair, a charging system which uses credits instead of real money may partially mitigate the advantage that

larger and economically sounder AUs may have under several pricing schemes.

6.5 Route charge modulation in literature

Article 16 of EC Regulation 391/2013 states: *"Member States [...] may [...] reduce the overall costs of air navigation services and increase their efficiency, in particular by modulating charges according to the level of congestion of the network in a specific area or on a specific route at specific times. [...] The modulation of charges shall not result in any overall change in revenue for the air navigation service provider [...]"*. This feature gives Member States and hence, ANSPs, the opportunity to use pricing modulation as an instrument to reduce recurring congestion problems. The possibility of modulating ANS charges by modifying the unit rate has been previously investigated in scientific literature.

Modulation of ANS charges was initially investigated by Andreatta and Odoni (2001), who envisaged the change of the unit rate to mitigate airspace congestion: higher rate in congested sectors, and lower in non-congested ones, where the total collected charges correspond to the costs incurred by ANSPs. However, the proposed concept was not elaborated in detail. A similar approach was described in Deschinkel et al. (2002). The introduction of a fixed rate in peak periods is one of the four en-route modulation charging options analyzed in Steer Davies Gleave (2015), recently prepared for the European Commission. The study recommends that future work should investigate the impact of a fixed supplement to the current charging formula. Numerical examples to compare the various options are, however, very limited (e.g. only three routes), and more emphasis is given to other issues like implementation and policy recommendations.

In Raffarin (2004) an alternative pricing for ANS charges is introduced, by giving airlines economic incentives to modify their behavior, so that the resulting routing choices are optimal from both social and individual points of view. The reconciliation between system (social) and user optima is also addressed by Jovanović et al. (2014) who describe an anticipatory, time-dependent modulation of ANS charges to bring the traffic demand more in line with the available network capacities. The charges are modulated so as to minimize the total cost incurred to AUs by introducing a charge for the use of a premium resource (overloaded sector) and providing economically reasonable alternatives for users in excess. The collected charges can be used to finance the use of alternative, under-used sectors. The results of a medium scale case study indicate that this mechanism may yield a fairly equitable assignment.

Adequate modeling of en-route charge modulation needs to address the impact on AUs.

That is to say, the impact on the route choice that the modulation would bring along, even in a non-congested setting. In this context, Castelli et al. (2013) propose a bi-level programming pricing model for the maximization of ANSP revenues through en-route charge modulation. Results from a small-scale real-world test case suggest that the unit rate can be an effective instrument for modification of the route choices, thus being a starting point for development of a pricing model with modulated en-route charges with the aim to alleviate imbalances.

The quoted Article 16 of EC Regulation 391/2013 motivated the setting up of the SESAR WP-E project SATURN (Strategic Allocation of Traffic Using Redistribution in the Network). SATURN investigated the possibility of using pricing mechanisms to redistribute part of the demand for airspace capacity and therefore making the expected growth in air traffic more sustainable. The project showed how economic signals could be provided to airspace users and ANSPs to improve capacity-demand balancing by anticipating part of the flight planning process to the strategic phase, as opposed to the currently applied inefficient measures. Project SATURN is the largest study carried out to date on this subject.

Furthermore, the benefits of having a centralized planner compared to ANSPs' and airspace users' decentralized maximization of self-interests were also investigated. Different pricing approaches have been investigated within the project, both monetary and non-monetary (specifically, permit-based); among the monetary based approaches, charges modulated according to expected traffic peaks (peak-load pricing) or according to the time of flight plan submission (rewarding predictability) (see Figure 6.4).

The effects of these three main mechanisms were tested on one full day of European air traffic, *12 September 2014* (one of the most trafficked days of 2014 not affected by major non-capacity related limitations). The results showed that the modulation of en-route charges is indeed a viable option to redistribute air traffic in Europe and reduce congestion and delays. These modulations in fact influence the operational costs of the airline operators in the sense that it may encourage them to reroute some flights or request different departure times to avoid expensive areas or take advantage of reduced charges. The obtained results were naturally a trade-off between total delay, network capacity limitations and recovery of ANSPs' operational costs.

6.6 Centralized Peak Load Pricing

Peak Load Pricing (PLP) has been evaluated in the framework of the SATURN project as the most appropriate strategy for reducing network congestion in the European ATM. The objective was indeed to obtain a pricing policy that guarantees a more balanced load on

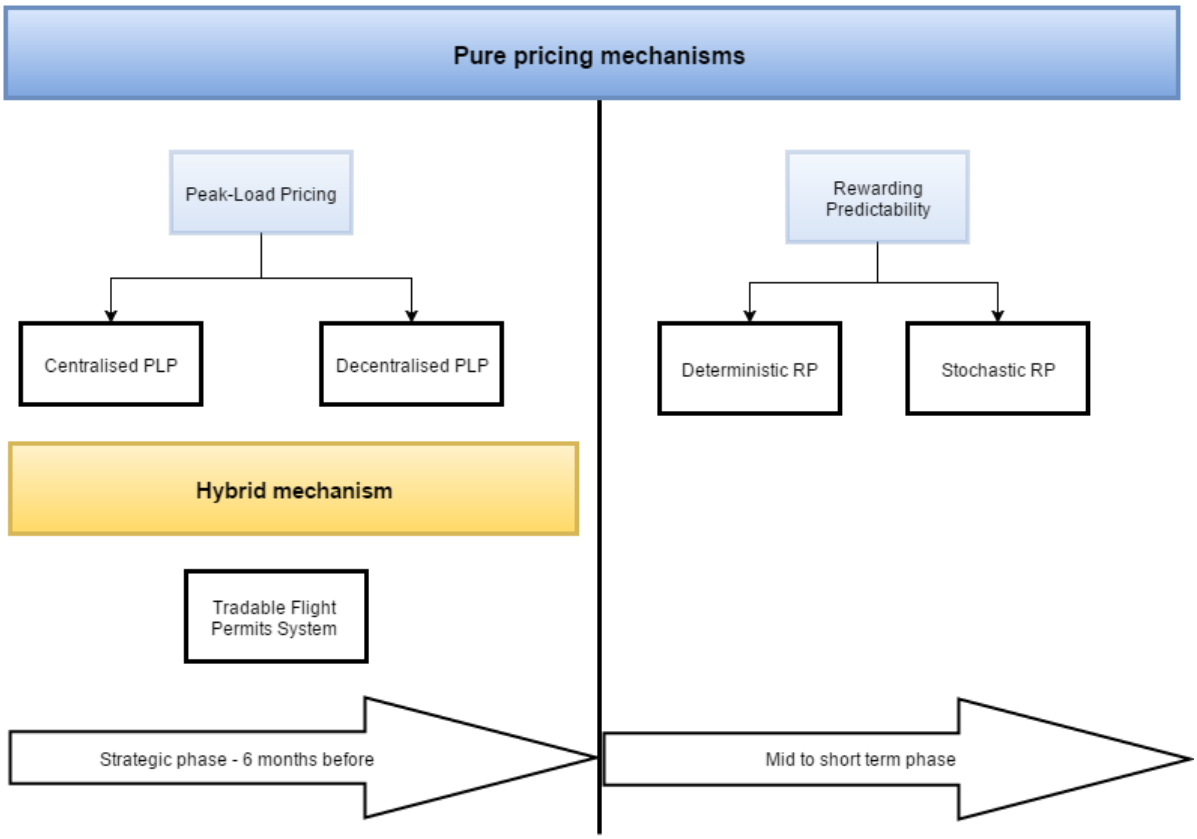


Figure 6.4: SATURN mechanisms and the time-line of their application (SATURN D.6.5)

network capacity by redistributing part of the traffic that would otherwise converge on high demand-capacity imbalance risk areas. Such a policy is expected to reduce this imbalance and consequent delay in the tactical phase by inducing airspace users to apply re-routing and strategic delay.

Peak-load pricing is a pricing mechanism commonly used in transport and utilities. It is a simplified form of congestion pricing with the fundamental assumptions that peaks in demand are occurring periodically, in both time and location (and are therefore predictable), and that demand has some degree of elasticity towards time and/or location of service consumption (and therefore is sensitive to price). Under these assumptions the PLP policy assigns a higher rate for times and/or locations where a peak in demand is expected and a lower rate for off-peak areas and times. By doing so, it is expected that part of the peak demand will divert their travel/consumption to a cheaper option. It is therefore essential that peak and off-peak prices are set adequately so that the pricing policy is effective with regard to both business sustainability and efficient capacity management. To achieve the former, total revenues should not be lower than total marginal costs. To achieve the latter

the peak rates should be greater than the users' willingness to pay in excess. It is expected that they will give preference to a cheaper off-peak option (in Borenstein et al. (2002) this principle is exemplified for the electricity retail market). PLP is also widely used in scheduled transport (public urban transport, railways, as in Peter (2003)) and is, in general, transparent and predictable to users, since peak times and prices are known in advance.

In the context of European ATM demand peaks in time are generally easy to predict, both at daily and seasonal level. Daily peaks usually occur in the morning between 7:00 and 10:00 and afternoon between 13:00 and 15:00 (UTC times); seasonal peaks occur close to holidays and special events, with the last week of June generally being the most trafficked in the year. As for price sensitivity, very few studies have been carried out on the sensitivity of airlines to route charges with regard to flight planning. Delgado (2015) carried out one such analysis on the traffic on 12 September 2014. The research presented in this paper shows that in the selection of a route by an aircraft operator the cost of the charging zones can be a factor. This impact is maximized around neighboring areas where differences in price are significant (e.g., Italy vs. the western Balcans) and alternative routes are similar with regard to other characteristics (e.g., route length, cruise altitude etc.). Therefore, some degree of sensitivity to price exists.

As already said in section 6.5, PLP is also compatible with the current unit rate setting guidelines by Eurocontrol and with EU Regulation No 391/2013 art. 16 stating that *"Member States [...] may [...] reduce the overall costs of air navigation services and increase their efficiency, in particular by modulating charges according to the level of congestion of the network in a specific area or on a specific route at specific times. [...] The modulation of charges shall not result in any overall change in revenue for the air navigation service provider. Over- or under recoveries shall be passed on to the following period. The modulation of air navigation charges means a variation of the en-route charge and/or the terminal charge [...]"*. This distinctive feature of Regulation 391/2013 provides Member States and therefore ANSPs with an operational instrument to manage demand in such way to help dealing with the recurring demand-capacity imbalance problems.

Project SATURN investigated a centralized approach to PLP (referred to as cPLP) where a central authority (or Central Planner, CP) is responsible for setting en-route charges on the whole network. cPLP has two phases. In the first phase congested airspace sectors and related peak and off-peak hours are identified. The identification can be done by analyzing the past traffic and route choice data (as said before, air traffic shows seasonal periodicity throughout the year), or by analyzing forecasted Origin/Destination (O/D) demand. Specifically, in either method, the traffic demand is counted for all the sectors, taking into account all

the flights and their routes, along a chosen time horizon (for example one hour). The ratio between hourly traffic count and nominal hourly capacity gives us the hourly load factor which can be used to assign the peak or off-peak label to a specific region for a specific hour. cPLP could in fact be applied at regional, national or local level, on the whole network or on selected regions only.

In the second phase, CP has to set en-route charges on the network and the AUs need to route each flight based on the set charges. These charges should guarantee that ANSPs are able to recover their operating costs and that AUs are able to perform flights, and at the same time it should mitigate the imbalance between demand and available airspace capacity. En-route charges are set by the CP to achieve a network level objective, i.e. to reduce the amount of delay on the network and thus the congestion. However, there is often a trade-off to be considered between the system level (CP) and the user level (AU) objectives. In general, allowing users to minimize, for instance, their individual delays (the so-called User Optimum assignment) does not lead to a solution where the global network congestion is also minimized. On the other hand, optimizing congestion at system level only (known as System Optimum assignment) would most likely penalize certain users more than others, which is also not ideal from an equity point of view.

Since in the current system en-route charges are set by the ANSPs and the AUs can only react to them by choosing alternative and cheaper routes, the relationship between the CP and the AUs has been modeled as a Stackelberg game where a leader (CP) makes his decision first, with the complete knowledge on how the follower(s) (AUs) would react to it. The Stackelberg equilibrium can be obtained by means of an optimization problem formulated as a bi-level linear programming model (see chapter 6.7.5) where the CP sets the rates for each sector/period and the AUs make their routing choice. Further constraints can be imposed to define the pricing scheme, ranging from allowing a different price for each sector/hour pair to imposing a single peak/off-peak couple of unit rates per ANSP, a depth analysis of these rates in the month of September over three years is shown in Table 6.6.

6.7 Centralized Peak-Load Pricing (cPLP) model

This section illustrates the Centralized Peak Load Pricing model (Bolić et al., 2017): Section 6.7.1 describes the assumption made, the used notation is presented in Section 6.7.2. Sections 6.7.3 and 6.7.4 present the formulation and constraints for the CP problem and for the AUs respectively, while Section 6.7.5 combines the two into a bi-level problem. Finally, Section 6.7.6 describes some relevant variants of the peak-load pricing scheme.

6.7.1 Modeling assumptions

1. *Fixed demand matrix*. Fixed number of flights between any airport pair in the network: the intention of the proposed pricing mechanism is to modify its spatial/temporal pattern to bring it in line with available capacities, not to scale down the total demand;

2. *Heterogeneous demand*, in terms of different aircraft types and associated cost coefficients;

3. The *infrastructure capacity constraints* are known in advance, in terms of airspace sectorization and maximum number of aircraft that can enter sectors per given period of time (i.e., nominal capacity). Since the mechanism is applied strategically, only nominal sector and airport capacities are considered, without variations introduced by regulations (which are caused by weather and other less predictable reasons, tactically);

4. *Finite set of possible (reasonable) 4D routes* for each Origin/Destination/Aircraft triple: users can select a route from a set of pre-determined routes (derived from actual traffic). Duration and profile of each route is assumed to be constant, for each aircraft type (i.e., speed profiles are assumed constant for each route/aircraft pair);

5. *Users are rational decision makers*. All AUs are assumed to choose the least-cost 4D route available. Flight cost components are attached to each route. AUs' routing decisions are therefore sensitive to modulations of en-route charges;

6. *Revenue neutrality* is established as a desired principle, meaning ANSPs' revenues are to be kept as close as possible to their operating costs: the adjustment of charges should not generate additional revenue (on top of the cost of ANS provision), nor deficit;

7. *Distance-proportional air navigation charges with sector-period based rates*. The pricing rule applied for air navigation charges is similar to the one currently in use but instead of a unique unit rate per country, two-level rates, namely *peak* and *off-peak*, differentiated by sector and period (i.e., one hour), are defined. A peak/off-peak rate pair is unique for each ANSP and therefore valid for all considered time periods and sectors in an ANSP;

8. *Peak times and locations are known in advance*. The expected load on a sector, during a specific time is estimated by analyzing initially submitted flight plans.

6.7.2 Model notation

F set of all flights, indexed by f

N set of all ANSPs, indexed by n

B set of all aircraft types, indexed by b

b_f aircraft type b used to operate flight f

W_{b_f} weight factor of aircraft type b used by flight f

A set of all airports, indexed by a

R set of all routes, indexed by r

R_f set of all routes that can be flown by flight f

S set of all sectors, indexed by s

S_n set of all sectors controlled by ANSP n

S_r set of all sectors crossed by route r

H time periods (hours), indexed by h

M time instants (minutes), indexed by m

MGS maximum ground shift (in minutes) allowed for a flight, i.e., the maximum difference between requested and allocated departure time (the formula applied is illustrated by eq. 6.7)

M_f possible departure time instants for flight f (i.e., $m \in [dt_f - MGS, dt_f + MGS]$)

T_h set of minutes m belonging to time period (hour) h

$Q_s^{(h)}$ capacity of sector s during time period (hour) h

$Q_{a,dep}^{(h)}$ departure capacity of airport a during time period (hour) h

$Q_{a,arr}^{(h)}$ arrival capacity of airport a during time period (hour) h

$Q_{a,gl}^{(h)}$ total (departures + arrivals) capacity of airport a during time period (hour) h

$D_{s,r}$ one hundredth of the great-circle distance flown in sector s if route r intersects sector s; 0 otherwise

$e_{s,r}$ estimated entry time since departure of route r in element (sector or airport) s

dt_f requested departure time for flight f

at_f requested arrival time for flight f

$adep_f$ departure airport for flight f

$ades_f$ destination airport for flight f

U_n actual (historical) en-route unit rate for ANSP n (€/service unit)

$GS_{f,r}^{(m)}$ global shift for flight f using route r and departing at time m

$OC_{f,r}^{(m)}$ airline operational costs for flight f using route r and departing at time m

$RC_{f,r}^{(m)}$ modulated en-route charges for flight f using route r and departing at time m

$TC_{f,r}^{(m)}$ total cost for operating flight f using route r and departing at time m.

6.7.3 Central Planner (CP) problem, upper level problem

Main decision variables: the decision process of the CP is illustrated by an optimization problem that identifies optimal rates for the considered airspace such that a user-optimal assignment of routes to flights will minimize the metric of network inefficiency represented by the CP objective function.
CP's decision variables are defined as:

$$RC_{f,r}^{(m)} \qquad \forall f \in F, r \in R_f, m \in M_f \tag{6.5}$$

en-route charges for flight f departing at minute m using route r

While AU's decision variables are:

$$x_{f,r}^{(m)} = \begin{cases} 1 & \text{if flight } f \text{ departs at minute } m \text{ using route } r \\ 0 & \text{otherwise} \end{cases} \qquad \forall f \in F, r \in R_f, m \in M_f \tag{6.6}$$

Route charges $RC_{f,r}^{(m)}$ can be defined in several ways, and their definition greatly affects the complexity of the model. The different pricing schemes analyses in this work and the

complexity of the resulting problem are described in Sec. 6.7.6

CP objective function: in general, the objective of the Central Planner is to minimize network inefficiency. Several metrics can be used to define efficiency. Delay minimization is usually a common choice in ATFM models (see for example Bertsimas et al., 2011) but since the cPLP models deal with strategic planning, the closer equivalent to tactical delay is in fact global shift minimization (meant as a measure of temporal displacement from the intentions stated by the AUs). The global shift for flight f using route r and departing at time m is defined as the sum of minutes of later-than-requested departure plus the number of minutes of earlier-than-requested arrival.

$$GS_{f,r}^{(m)} = \max\{0, m - dt_f\} + \max\{0, at_f - (m + e_{ades_f,r})\} \quad \forall f \in F, r \in R_f, m \in M_f \tag{6.7}$$

A second and third components to be minimized in the CP objective are respectively the violation of the revenue neutrality constraints ϵ and the violations of capacity constraints α, explained in the following. The complete CP objective function is therefore the following:

(6.8)

$$\min_x \left(\sum_{\substack{f\in F,\\ r\in R_f,\\ m\in M_f}} GS_{f,r}^{(m)} \cdot x_{f,r}^{(m)} + K_1 \cdot |\epsilon| + K_2 \cdot \sum_{s\in S, h\in H} \alpha_s^{(h)} + K_3 \cdot \sum_{a\in A, h\in H} \left(\alpha_{a,dep}^{(h)} + \alpha_{a,arr}^{(h)} + \alpha_{a,gl}^{(h)} \right) \right)$$

where K_1, K_2, K_3 are the weights to be assigned to the respective objective components.

Revenue neutrality constraint: the CP should optimize the rates so that ANSPs are guaranteed to recover their operational costs for providing air navigation services to AUs, as established by the cost recovery/determined costs system through which ANSPs are funded. Therefore, the rates chosen by the CP should be revenue-neutral, meaning that they should not generate additional revenues. By definition, the currently applied country-based unit rates represent the marginal cost for providing one ANS service unit. As illustrated in Section 6.5, an ANS service unit is defined as the product of a distance-related term (specifically, one hundredth of the Great Circle Distance between entry and exit points in the national airspace) and an aircraft-weight-related term (namely, the square root of one fiftieth of the maximum takeoff weight of the aircraft).
The revenue neutrality constraint therefore states that the revenues levied from the modulated route charges should not differ (in absolute value) from the revenues that would

be obtained from the historical unit rates by more than a variable term ϵ, which is to be minimized in the CP objective function through the $K_1 \cdot |\epsilon|$ term.

$$\sum_{\substack{f \in F, \\ r \in R_f, \\ m \in M_f}} RC_{f,r}^{(m)} \cdot x_{f,r}^{(m)} - \sum_{\substack{n \in N, f \in F, \\ r \in R_f, m \in M_f \\ s \in (S_n \cap S_r)}} U_n \cdot D_{s,r} \cdot W_{b_f} \cdot x_{f,r}^{(m)} = \epsilon \tag{6.9}$$

Sector and airport capacity constraints: under an effective peak-load pricing policy, the number of capacity breaches will be minimum. A breach occurs whenever the number of aircraft entering a sector or an airport during a certain time period (e.g., one hour) exceeds the declared capacity for that resource. Capacity constraints are therefore defined by stating that the number or aircraft entering a resource during a certain hour minus the declared capacity should not exceed a variable quantity α. These are defined for each capacitated resource and every hour when the capacity constraint is applied. In the model, they are represented by the following variables:

$\alpha_s^{(h)}$ = Number of flights exceeding capacity of sector s during hour h;

$\alpha_{a,dep}^{(h)}$ = Number of flights exceeding departure capacity of airport a during hour h;

$\alpha_{a,arr}^{(h)}$ = Number of flights exceeding arrival capacity of airport a during hour h;

$\alpha_{a,gl}^{(h)}$ = Number of flights exceeding global (departure and arrival) capacity of airport a during hour h.

The value of α variables could either be forced to zero (*hard capacity constraint*) or penalized in the objective (*soft capacity constraint*). While hard constraints would be a better choice in terms of formulation and could be used for cutting planes generation, soft capacity constraints are more realistic with regard to the application, and are therefore preferred in this case. In fact they better represent actual ATC practice, where a mild violation of nominal capacity is tolerated and handled by controllers. The formulation of the capacity constraints is therefore the following:

$$\sum_{f\in F, r\in R_f, h\in H, m\in M_f | m\in T_h} x_{f,r}^{(m)} - Q_s^{(h)} \leq \alpha_s^{(h)} \qquad \forall s\in S, h\in H \tag{6.10}$$

$$\sum_{\substack{f\in F | adep_f = a, r\in R_f, \\ h\in H, m\in M_f | m\in T_h,}} x_{f,r}^{(m)} - Q_{a,dep}^{(h)} \leq \alpha_{a,dep}^{(h)} \qquad \forall a\in A, h\in H \tag{6.11}$$

$$\sum_{\substack{f\in F | ades_f = a, r\in R_f, h\in H, \\ m\in M_f | (m+e_{a,r})\in T_h; e_{a,r}\neq 0}} x_{f,r}^{(m)} - Q_{a,arr}^{(h)} \leq \alpha_{a,arr}^{(h)} \qquad \forall a\in A, h\in H \tag{6.12}$$

$$\sum_{\substack{f\in F | adep_f = a \vee ades_f = a, r\in R_f, \\ h\in H, m\in M_f | (m+e_{a,r})\in T_h,}} x_{f,r}^{(m)} - Q_{a,gl}^{(h)} \leq \alpha_{a,gl}^{(h)} \qquad \forall a\in A, h\in H \tag{6.13}$$

$$\alpha_s^{(h)} \geq 0 \qquad \forall s\in S, h\in H \tag{6.14}$$

$$\alpha_{a,dep}^{(h)}, \alpha_{a,arr}^{(h)}, \alpha_{a,gl}^{(h)} \geq 0 \qquad \forall a\in A, h\in H \tag{6.15}$$

and their violation is penalized in the objective function by the following term:

$$K_2 \cdot \sum_{s\in S, h\in H} \alpha_s^{(h)} + K_3 \cdot \sum_{a\in A, h\in H} \left(\alpha_{a,dep}^{(h)} + \alpha_{a,arr}^{(h)} + \alpha_{a,gl}^{(h)} \right) \tag{6.16}$$

In order to reduce demand-capacity imbalance the CP will attempt to distribute the traffic evenly over the network, by minimizing the traffic flow considering the sector capacity ratio.

6.7.4 Airspace Users' (AUs) problem, lower-level problem

The decision process of the AU is modeled as an optimization problem where each AU aims at choosing the routes that minimize costs for each of its flights. AUs' decision variables are introduced in Equation 6.6.

Flight operational costs: the cost for operating a flight typically includes route and terminal charges, aircraft fuel and maintenance costs, staff costs. Most airline operators release an annual report with aggregated cost figures for each category (see for example Ryanair, 2015). Cook and Tanner (2015) identify and calculate the relevant cost coefficients per minute for fifteen reference aircraft types (estimated to cover 90% of European air traffic).The following coefficients are defined accordingly to the latter:

cam_b strategic cost of airborne maintenance for aircraft type b (€/min)

cgm_b strategic cost of ground maintenance for aircraft type b (€/min)
cf_b strategic cost of fleet utilization for aircraft type b (€/min)
cc_b strategic cost of crew utilization for aircraft type b (€/min)
afb_b average fuel burn for aircraft type b (Kg/min)
fc fuel cost (€/Kg)

Airborne operations costs include aircraft maintenance, fleet and crew utilization costs plus fuel costs. These costs are accounted for the duration of a flight with the chosen route. Ground operations costs include aircraft maintenance, fleet and crew utilization costs. These costs are accounted for every minute of strategic ground shift assigned to a flight. Note that terminal charges are excluded from this calculation because demand (i.e., origin and destination airports) is assumed as fixed. Aggregated strategic cost coefficients for airborne operations (ca_b) and ground operations (cg_b) are defined as follows:

$$ca_b = cam_b + cf_b + cc_b + fc \cdot afb_b \qquad \forall b \in B \tag{6.17}$$

$$cg_b = cgm_b + cf_b + cc_b \qquad \forall b \in B \tag{6.18}$$

Operational costs for operating flight f using route r and departing at time m is therefore given by the ground shift cost coefficient times the ground shift (i.e., minutes of later departure), plus the airborne cost coefficient, times the route duration.

$$OC_{f,r}^{(m)} = cg_b \cdot \left(GS_{f,r}^{(m)} - (e_{ades_f,r} - \min_{r' \in R_f} e_{ades_f,r'}) \right) + ca_b \cdot e_{ades_f,r} \quad \forall f \in F, r \in R_f, m \in M_f \tag{6.19}$$

AU Objective function: the total cost an airline has to endure for operating a flight f using route r and departing at time m is the sum of operational costs and en-route charges. Minimizing total cost for operating flights is the objective function proper to each airspace user.

$$\min_x \quad TC_{f,r}^{(m)} \cdot x_{f,r}^{(m)} = (OC_{f,r}^{(m)} + RC_{f,r}^{(m)}) \cdot x_{f,r}^{(m)} \qquad \forall f \in F, r \in R_f, m \in M_f \tag{6.20}$$

Route uniqueness constraints: for every flight, exactly one route r and one departure time m are to be chosen.

$$\sum_{r\in R_f, m\in M_f} x_{f,r}^{(m)} = 1 \qquad \forall f \in F \tag{6.21}$$

6.7.5 Model formulation: bi-level cPLP

The CP problem can be combined with the AU problem into a bi-level formulation representing a Stackelberg game between the two agents, with the CP acting as leader and the AUs as followers. In such a configuration, the CP is able to anticipate the followers' reaction (in terms of route choice) to his/her pricing strategies and can therefore choose a set of rates that will optimize his objective by anticipating the associated followers' optimal route choice.
The formulation of the BOP is the following:

$$\min_x \left(\sum_{\substack{f\in F,\\ r\in R_f,\\ m\in M_f}} GS_{f,r}^{(m)} \cdot x_{f,r}^{(m)} + K_1 \cdot |\epsilon| + K_2 \cdot \sum_{s\in S, h\in H} \alpha_s^{(h)} + K_3 \cdot \sum_{a\in A, h\in H} \left(\alpha_{a,dep}^{(h)} + \alpha_{a,arr}^{(h)} + \alpha_{a,gl}^{(h)} \right) \right) \tag{6.22}$$

$$\text{s.t.} \sum_{\substack{f\in F,\\ r\in R_f,\\ m\in M_f}} RC_{f,r}^{(m)} \cdot x_{f,r}^{(m)} - \sum_{\substack{n\in N, f\in F,\\ r\in R_f, m\in M_f\\ s\in (S_n\cap S_r)}} U_n \cdot D_{s,r} \cdot W_{b_f} \cdot x_{f,r}^{(m)} = \epsilon \tag{6.23}$$

$$\min_x \sum_{\substack{f\in F,\\ r\in R_f,\\ m\in M_f}} (OC_{f,r}^{(m)} + RC_{f,r}^{(m)}) \cdot x_{f,r}^{(m)} \tag{6.24}$$

$$\sum_{f\in F, r\in R_f, h\in H, m\in M_f | m\in T_h} x_{f,r}^{(m)} - Q_s^{(h)} \le \alpha_s^{(h)} \qquad \forall s \in S, h \in H \tag{6.25}$$

$$\sum_{\substack{f\in F | adep_f = a, r\in R_f,\\ h\in H, m\in M_f | m\in T_h,}} x_{f,r}^{(m)} - Q_{a,dep}^{(h)} \le \alpha_{a,dep}^{(h)} \qquad \forall a \in A, h \in H \tag{6.26}$$

$$\sum_{\substack{f\in F | ades_f = a, r\in R_f, h\in H,\\ m\in M_f | (m+e_{a,r})\in T_h; e_{a,r}\neq 0}} x_{f,r}^{(m)} - Q_{a,arr}^{(h)} \le \alpha_{a,arr}^{(h)} \qquad \forall a \in A, h \in H \tag{6.27}$$

$$\sum_{\substack{f\in F | adep_f = a \vee ades_f = a, r\in R_f,\\ h\in H, m\in M_f | (m+e_{a,r})\in T_h,}} x_{f,r}^{(m)} - Q_{a,gl}^{(h)} \le \alpha_{a,gl}^{(h)} \qquad \forall a \in A, h \in H \tag{6.28}$$

$$\sum_{r \in R_f, m \in M_f} x_{f,r}^{(m)} = 1 \qquad \forall f \in F \tag{6.29}$$

$$x_{f,r}^{(m)} \in \{0,1\}, RC_{f,r}^{(m)} \geq 0 \qquad \forall f \in F, r \in R_f, m \in M_f \tag{6.30}$$

$$\alpha_s^{(h)} \geq 0 \qquad \forall s \in S, h \in H \tag{6.31}$$

$$\alpha_{a,dep}^{(h)}, \alpha_{a,arr}^{(h)}, \alpha_{a,gl}^{(h)} \geq 0 \qquad \forall a \in A, h \in H \tag{6.32}$$

6.7.6 Pricing schemes for cPLP

According to the way the en-route charges variables $RC_{f,r}^{(m)}$ are defined, different pricing schemes can be derived. This section illustrates some relevant examples.

6.7.6.1 Trajectory-based pricing

In a trajectory-based pricing scheme rates are set on a 4-dimensional route basis, meaning that a price is assigned to the entire route and the modulation depends on the time chosen for departure.

In terms of modeling, each $RC_{f,r}^{(m)}$ variable is independent of the others and therefore no further constraint is added to the model.

This scheme is similar to the one applied to a variant of the Network Pricing Problem called *NPP with connected toll arcs* (Heilporn, 2008; Violin, 2014), as already explained in Chapter 5. It concerns a highway network where all paths share the same cost structure, a fixed part for the origin-to-highway-entry-point and highway-exit-to-destination-point portions of the trip and a variable cost for the highway segment in between (highway-entry-point-to-highway-exit-point). A different toll has to be assigned to each highway segment and is charged to all users who use it. Different highway segments have different tolls. This NPP variant was proved to be NP-Hard in Heilporn (2008).

6.7.6.2 Segment-based pricing

In a segment-based pricing scheme different rates are set for each airspace sector. En-route charges for a route are therefore calculated as the sum of the charges associated with the segment flown in each airspace sector of that route. Modulation depends on the entry time in each sector, thus, indirectly, on the time chosen for departure.

Let us therefore introduce the $P_s^{(h)}$ rates as the unit rates applied on sector s during hour h. Resulting en-route charges for a flight f flying on route r and departing at time m are

defined as follows:

$$RC_{f,r}^{(m)} = \sum_{s \in S_r} P_s^{(h|(m+e_{s,r}) \in T_h)} \cdot D_{s,r} \cdot W_{b_f} \tag{6.33}$$

This scheme is similar to the one applied to the general arc-based formulation of the Network Pricing Problem, where the arcs of the network (or a subset of them) have each a different fixed cost plus a variable toll, that is charged to all users that use it. This NPP variant was proved to be NP-Hard in Labbé et al. (1998).

6.7.6.3 ANSP-based peak-load pricing

In an ANSP-based pricing scheme, rates are applied at ANSP level, meaning that all sectors included in the airspace of a certain ANSP share the same rates. En-route charge modulation is accomplished by imposing a higher rate when and where traffic peaks are expected (*peak rate*) and a lower rate for less trafficked periods and areas (*off peak rate*). A couple of peak and off peak rates is therefore defined for each ANSP and their values are assigned to each sector/hour pair, as follows:

$$P_s^{(h)} = \begin{cases} Pp_n & \text{if } h \text{ is peak time for sector } s \\ Po_n & \text{otherwise} \end{cases} \qquad \forall n \in N, s \in Sn_n, h \in H \tag{6.34}$$

where P_{p_n} and P_{o_n} represent, respectively, the peak and off-peak rate for ANSP n. Resulting en-route charges for a flight f flying on route r and departing at time m are calculated as in the general segment-based pricing scheme (Eq. 6.33).

A variant of the ANSP based peak-load pricing scheme is obtained if the peak and off-peak rates are parametrized, meaning that one is defined proportionally to the other. In this case the parametrized set of tariffs is defined as follows:

$$P_s^{(h)} = \begin{cases} Po_n & \text{if } h \text{ is off-peak time for sector } s \\ \delta_n \cdot Po_n & \text{if } h \text{ is peak time for sector } s \end{cases} \qquad \forall n \in N, s \in Sn_n, h \in H, \delta_n \geq 1 \tag{6.35}$$

where δ_n is a term greater than 1 denoting the proportionality ratio between the peak and off-peak rates.

Theoretically, either Po_n or δ_n could be decision variables of the problem. The fixed δ case is suitable for keeping the difference between peak and off peak rates as small as possible. The fixed Po_n case is suitable for evaluating elasticity in terms of routing alternatives within a specific ANSP territory. By setting the off peak as equal to the historical unit

rate ($Po_n = U_n$) it is possible for example to find the minimum increment of the unit rate that could reduce the amount of demand-capacity imbalance within a state. This scheme however appears to be more realistic with regard to possible implementation, therefore the fixed δ scenario will not be discussed any further in the remainder of this work.

ANSP-based peak-load pricing schemes have the desirable feature of being simple and transparent, being the most similar to the schemes already applied successfully in several other industries. Therefore only this type of scheme will be further analyzed in the remainder of the work, in some variants:

- *cPLP with fixed $Po_n = Pp_n = U_n$ (cPLP-one)*
  A one-rate per ANSP scheme obtained by adding Eq. 6.33 and 6.35 to the model, with variables $\delta_n = 0$ and a constant values of Po_n set equal to the historical unit rate U_n. The cPLP-one will be used as *baseline* because it is the reference solution able to correctly represent in the model the pricing situation.

- *cPLP with variable Po_n and Pp_n (cPLP-two)*
  A two-modulated-rates per ANSP scheme obtained by adding Eq. 6.33 and 6.34 to the model, where both the peak and off-peak rates can change freely.

- *cPLP with variable Po_n (cPLP-p)*
  In the case where δ_n is fixed, only one pricing variable needs to be set per ANSP, that is, the off-peak tariff Po_n.

- *cPLP with variable δ_n (cPLP-δ)*
  In the case where the pricing variable Po_n is fixed and given, the decision variables for the problem is the proportional increment to the tariff δ_n. In the case where the base tariff Po_n is set as equal to the current unit rate for each ANSP, U_n, the cost recovery constraint REF is trivially verified for each value of $\delta_n \geq 1$. Setting $Po_n = U_n$ is a realistic assumption since the unit rate represents the average cost for air navigation service unit; it is reasonable to assume that a service unit under demand-capacity imbalance has higher-than-average provision cost.

6.8 MOGASI: Framework application to cPLP

As already explained in detail in Chapter 4, the optimization framework described in this work is applied to bi-level problems with a nested approach (see Section 4.3.3). As the name suggests, these approaches rely on two optimization algorithms, one executed within the

other. According to the complexity of the optimization task, it is important to decide to use either an evolutionary algorithm at both levels or an evolutionary algorithm at the leader level and a exact optimization algorithm at the follower level. The latter has been chosen for the NPP application described in Chapter 5.

After an extensive review of the cPLP problem formulated as a bi-level problem, described in Section 6.7.5, it was decided to adopt a nested approach (see Chapter 4.3.3) based on an application of the framework GA for the leader level and a custom exact algorithm for the follower level, described in Section 6.9. In particular, the pricing scheme described in Section 6.7.6.3, which is a parametrized adaptation of cPLP-two, makes it possible to assign to the GA only the task of determining the $P_s^{(h)}$ rates (Eq. 6.34). Hence, the GA requires the assignment of Po_n and Pp_n for each ANSP as decision variables. These concepts are further discussed in Section 6.9.

Intuitively a simple iteration loop can be defined. The leader level, governed by the GA, assigns its decision variables (once their values are fixed) to the follower level. The exact solver at the follower level assigns of the most advantageous route with the lowest cost to each flight. These route choices will allow the leader to evaluate the quantitative indexes for the assignment of a fitness value to each solution, and thus for determining whether this solution is eligible for guiding the creation of the next generation of solutions.

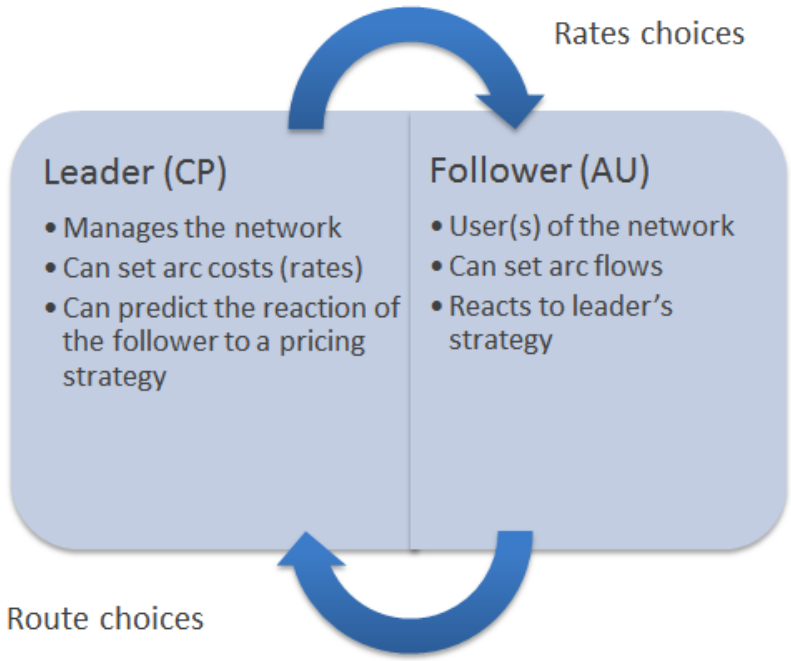


Figure 6.5: Example of an iterative decision loop for the cPLP optimization

The following Section contains the details of the cPLP problem tackled in this work, the necessary simplifications of the real data that lead to the creation of the problem instance, solution evaluation criteria and decisions taken to configure the optimization problem. At the end, the experimental data and the post-processing analysis thereof are presented.

6.8.1 Instance Design

The concepts developed in the framework of the SATURN project, i.e. bi-level formulation of the problem, existence of a centralized planner instead of the current decentralized charge-setting system and modulation of charges according to the peak-load pricing and rewarding predictability schemes, were decided to be tested on real air traffic data collected on 12th September 2014. This was the fourth busiest day of the year, selected because it was not disrupted by unusual event. Data on air traffic and network structures were sourced from Eurocontrol's Demand Data Repository DDR2 (Eurocontrol, 2014a).

The data covers the entire European airspace and includes different elements required for running the model, in particular: eligible flights, network configuration and resource capacities (sectors and airports), aircraft types with their flight and fuel costs, airline types and charge unit rates set by ANSPs.

6.8.1.1 Eligible Flights

All IFR scheduled passenger flights that departed from or arrived in the European airspace were taken into account excluding any military, overflights, helicopters, flights departing and arriving at the same airports, and flights with flight plans containing indicators "ZZZZ" or "AFIL"[3]. These data were sourced from DDR2 M1, the last filed flight plans.

6.8.1.2 Network Configuration and Resource Capacities

The network configuration changes during the day, mainly but only exclusively according to the planned traffic flows. These changes are represented as opening and closing times of sectors and as such need to be accounted for. In certain airspace areas the configuration may even change several times a day and some configurations may be in place for a limited amount of time. These changes cannot be ignored, so the dynamic sectorization in force on 12th September 2014 has been applied. DDR data also contain information on the declared capacity of sectors during their opening times.

[3]The AFIL indicator means that the flight plan has been transmitted to the appropriate air traffic service unit during flight, whereas ZZZZ is used if no official aircraft type designator has been assigned by ICAO.

Airports constitute another set of resources for which the capacity data are needed and are obtained from the DDR. However, some of the airport capacities were corrected because on closer inspection the declared capacities were much lower than the actual number of operations handled in a period of time. Capacities required corrections at the following airports: London Gatwick (EGKK); Nice Cote D'Azur (LFMN); Dusseldorf (EDDL); Istanbul Ataturk (LTBA) and Bergamo Orio al Serio (LIME).

6.8.1.3 Route Choices

A route is given as a combination of 3D trajectories (set of crossed sectors) and departure times. For each origin/destination (O/D) pair and aircraft type combination a set of 3D routes is defined. The routes per O/D/Aircraft triplet are determined through a clustering of historical flight data from the two weeks preceding 12th September 2014. Only routes differing significantly from one another in terms of geographical distance (specifically, more than 20 Km in the points where the distance between the two routes is maximal measured in 3-dimensional space) are taken in consideration. This reduces the number of viable routes per Origin-Destination-aircraft type triplet to an average of 3.4 routes per triplet. Allowed departure times range between 30 minutes before and 30 minutes after the time originally requested for each flight.

6.8.1.4 Flight and Fuel Costs

The main operational costs of AUs include fuel, crew, fleet (i.e. depreciation, rentals and leases) and maintenance costs. The reference values for European delay costs incurred by AUs are reported in Cook and Tanner (2011), updated and extended by Cook and Tanner (2015). The cost of delay is calculated separately for strategic delays (those accounted for in advance) and tactical delays (those incurred on the day of operations), as expressed in Section 6.7.4. Costs reference values for the most commonly used aircrafts in Europe are assigned under three cost scenarios: 'low', 'base' and 'high' cost scenarios. The base cost scenario is to the greatest possible extent designed to reflect the typical case. All cost calculations are performed for 15 reference aircraft types, which however make up for more than 90% of European air traffic. They are the following: B733, B734, B735, B738, B752, A319, A320, A321, A332, AT43, AT72, B744, B763, DH8D and E190.

Maintenance relates to factors such as the mechanical attrition of aircraft waiting at gates or airport accepting longer re-routes to obtain a better tactical departure slot.

Fleet costs refer to the full cost of fleet financing, such as depreciation, rentals and leases of flight equipment.

Table 6.4: Example of aircraft clustering results

Reference Aircraft (ICAO designator)	**Reference MTOW (source: NEST)**	**Other aircraft in the same cluster: ICAO designator**
A319	68.98	B737
A320	74.48	MD83; MD88; MD90
A321	86.47	B722; TU22
A332	229.51	B773; B788; MD11; A333; A342; A343
AT43	16.83	AT44 ; AT45; B25; DH8A; IL28
AT72	22.15	AT75 ; E135 ; E145; AN26
B733	61.6	B736 ; MD87 ; AN12
B734	65.63	A318 ; MD81
B735	56.55	DC92 ; B732
B738	76.47	B721 ; B739
B744	392.09	B773 ; A346 ; B741; A388; A345
B752	111.17	B720 ; B753; B701
B763	181.81	B764; B787; A310
DH8D	29.11	E170; GLF3
E190	49.07	T134; RJ85;

Crew costs, i.e. typical pilot and flight attendant salaries, were calculated in 2008 for various European airlines, then updated using the 2010 data. The high cost scenario is based on overtime rates. The base cost scenario is based on typical time-based costs. The crew costs commonly apply to ground and airborne phases.

6.8.1.5 Aircraft Types

All aircrafts used for passenger services have been grouped into clusters using the 15 reference aircraft types from Cook and Tanner (2015) study as cluster centroids (see Table 6.4). This clustering was necessary to facilitate the assessment of operational strategic costs for each flight. The report states that the square root of the maximum takeoff weight (MTOW) of the aircraft can be taken as a sound proxy for two aircraft having similar size and operational costs.

Furthermore, an additional aircraft cluster (Other) has been added for all the aircrafts with MTOW<10t, which constitute a fair portion of flights on the chosen day (about 7%). A decision was made to keep these aircrafts and associated flights in the sample to avoid adjustments of airspace sectorization since the removal of these flights can result in a low traffic load for the given sectorization.

Table 6.5: Cost scenario assigned to flights

Cost scenario	Flight categorisation	Proportion of flights
Low	All low-cost carrier flights	30%
High	Full-service flights into hub airports Regional flights into hub airports	20%
Base	All other flights	50%

The fuel costs are based on 2014 costs reported by Cook and Tanner (2015) for 'low' (0.7), 'base' (0.8) and 'high' (0.9) cost scenarios, expressed in euros per kg. Although the cost of carbon emissions (to airlines) has been considered, this has been disregarded due to its minor impact on fuel cost.

6.8.1.6 Airline Types

Airlines, or AUs, are divided into four types: full-service, low-cost, charter and regional. Airlines providing services covering more than one category are assigned to the most appropriate type. All uncategorized airlines are assigned to the regional category since this has a fairly cost-neutral effect on the model. Other non-commercial IFR passenger operators, such as all-cargo, military transport and private/business aviation, are flagged for exclusion, being out of scope for strategic traffic management. Based on this subdivision, flights can be grouped into three different flight operational cost profiles, as shown in Table 6.5.
e.g. the high cost scenario has been used for full-service and regional flights arriving at hub airports (i.e. inbound flights only). 14 ECAC hub airports were selected using ACI EUROPE's Group 1 definition, that is airports with over 25 million passengers in 2014 (ACI EUROPE, 2015).

6.8.1.7 Assessment Indicators

The described cPLP mechanism distributes traffic both in time (shifts in departure and/or arrival times) and space (alternative routes) to balance demand capacity. The resulting traffic pattern avoids bottlenecks, but also impacts other important phenomena of the system. For this reason a global assessment can be performed to take other indicators into account and look into the resulting trade-offs. The indicators can be calculated for each scenario and scenario variants to enable the comparison of their impacts. These possible main indicators are as follows:

- *Horizontal en-route flight efficiency*: The horizontal en-route efficiency used in this

work is the difference between the origin-destination en-route distance of assigned routes (L) and the great circle distance (G) between the origin and destination, expressed as a percentage of the great circle distance ($(L-G)/G$, %).

- *Sector capacity utilization*: This indicator shows for each open sector the capacity utilization, measured as the number of sector entries over the declared capacity (for each hour).

- *ANSP revenues*. This indicator measures the ANSP revenues collected by each ANSP in each scenario. The indicator sums all the charges each flight needs to pay to pass through the ANSP's airspace.

- *Charges per flight:* This indicator measures the charges imposed on the flights and used to cover the costs of the provision of air navigation services. Considering the mechanisms developed in this work it is sum of charges per flight, which include route charges, modulation of charges or incentives, as applied by the model/mechanism.

- *Distribution of charges across AUs*. The aim of this indicator is to assess the equity that different mechanisms and models produce. In practice, it is the aggregation of the previous indicator across AUs.

- *Flight operation costs*. Based on the cost data from Cook and Tanner (2015), the cost of operation of flights is calculated considering the assigned routes and strategic shifts.

- *Departure shift*. Absolute difference between the requested and assigned departure time.

- *Arrival shift*. Absolute Difference between the arrival time obtained by departing at requested time using the shortest route and the assigned arrival time.

6.8.1.8 En-route Unit Rates

Unit Rates as set by each national ANSP for September 2014 have been applied, as described in Section 6.6. The European airspace considered in this work comprises 41 States which are members of the Eurocontrol's CRCO system plus Estonia and Ukraine for geographic reasons.

Table 6.6: Unit rates in Europe in the month of September of 2013-2015

		Unit rate (€)		
Zone		**2015**	**2014**	**2013**
AZ	Portugal Santa Maria	10.43	10.60	8.96
EB	Belgium-Luxembourg	70.79	72.19	67.99
ED	Germany	90.26	77.47	76.65
EF	Finland	56.34	52.21	49.79
EG	United Kingdom	102.34	88.54	84.29
EH	Netherlands	66.68	66.62	65.53
EI	Ireland	29.71	30.77	28.35
EK	Denmark	63.25	71.43	73.58
EN	Norway	46.53	52.24	54.31
EP	Poland	34.41	35.33	35.66
ES	Sweden	64.09	69.71	76.31
EV	Latvia	27.69	28.59	29.01
EY	Lithuania	46.93	45.92	46.46
GC	Spain Canarias	58.47	58.51	58.51
LA	Albania	45.76	45.58	44.61
LB	Bulgaria	31.00	37.53	36.49
LC	Cyprus	37.02	38.56	37.72
LD	Croatia	46.57	43.11	41.49
LE	Spain Continental	71.80	71.84	71.84
LF	France	70.11	65.92	64.76
LG	Greece	38.49	34.68	33.89
LH	Hungary	36.08	40.14	41.00
LI	Italy	78.91	78.98	78.98
LJ	Slovenia	68.47	67.61	66.74
LK	Czech Republic	43.97	43.26	45.19
LM	Malta	22.44	27.76	31.65
LO	Austria	73.45	73.54	70.21
LP	Portugal Lisbon	37.24	38.89	34.65
LQ	Bosnia Herzegovina	38.03	51.46	45.52
LR	Romania	37.36	38.34	39.15
LS	Switzerland	110.42	100.41	96.68
LT	Turkey	28.07	32.12	31.14
LU	Moldova	43.34	38.05	40.90
LW	FYROM	55.35	60.10	60.09
LY	Serbia-Montenegro-KFOR	41.86	46.83	45.44
LZ	Slovak Republic	55.10	61.08	60.92
UD	Armenia	38.99	34.24	29.67
UG	Georgia	21.83	24.85	24.85

6.9 Optimization Problem Setup

Analyzing the original objective formulation expressed in Section 6.7.5, it is possible to see that there are three different components to be optimized in the problem target. These components are balanced by introducing the weights K_1, K_2, K_3 which steer the exploration through the space of the three objectives: minimization of the total shift for all flights, absolute value of ϵ discussed in Section 6.7.3 and violation of the capacity of sectors/airports. The heuristic strategy presented in this work enables the retainment of all objectives and thus a reformulation of the problem as single-objective is unnecessary. The reason for this is that GAs are general considered to be suitable for multi-objective problems and for finding reasonably good trade-off solutions (i.e., Pareto solutions, referring to the formulation used so far, equivalent to trying all possible combinations of the $K1$, $K2$ and $K3$ parameters). Owing to the stochastic element contained in GAs, MOGASI is able to avoid the creation of a number of almost identical sub-optimal individuals. It is a heuristic algorithm, so there is no guarantee that the final solution will be the optimal one, but it is able to explore the solution search space and eventually converge to a solution in a reasonable amount of time even in case of a very large search space. As explained in Chapter 2, solving a multi-objective optimization problem reformulated as single-objective could radically change the shape of the objective landscape and doing so it could undermine for certain weight values the possibility to explore potentially interesting areas.

In order to let the algorithm with the largest possible maneuvering space and at the same time maintain a reasonable selection pressure, it was necessary to properly select the optimization objectives. Considering the seven assessment indicators presented in Section 6.8.1.7, preliminary optimization runs were carried out on smaller instances to identify the most appropriate objectives. Even though at the beginning the capacity violation was always used as minimization objective, the experiments showed that MOGASI had difficulties to converge most probably due to the high number of equivalent solutions in terms of capacity violation. The problem was therefore configured as bi-objective as follows:

- Minimization of the global shift for all flights ($GS_{f,r}^{(m)}$).
- Minimization of the maximum revenue neutrality violation $|\epsilon_n|$ (Eq. 6.9).

The solving procedure applies the GA to the peak and off peak rates: for a candidate assignment of rates, the corresponding optimal routing choices for the flights are calculated, and these latter are used to compute the corresponding objective value and therefore the fitness of that particular set of tariffs. The pricing variables are defined as:

$$P_s^{(h)} = \begin{cases} Po_n = U_n + \lambda_n & \text{if } h \text{ is off-peak time for sector } s \\ Pp_n = (1+\delta_n) \cdot Po_n & \text{if } h \text{ is peak time for sector } s \end{cases} \tag{6.36}$$

$$\forall n \in N, s \in Sn_n,\ h \in H, Po_n \geq 0, \delta \geq 0, \lambda_n \in [-U_n, 50]$$

where λ_n and δ_n are respectively off-peak variation of the historical unit rate and the increase percentage for the peak rates with respect to the off-peak rates. The margin of increment/decrement of the off-peak rates with respect to unit rates is 50, but their values cannot be negative. The actual off-peak rate is computed as the unit rate plus/minus this variation (as found by the algorithm). The peak charges are expressed as the off-peak rate multiplied by the sum of the off-peak variation plus 1. Furthermore, to drive the search through the solution space towards feasible regions, i.e., fulfilling all constraints (see Section 6.7.3), an additional constraint is set on the revenue neutrality violation term ϵ_n stating that the maximum revenue neutrality violation over all the ANSPs should not exceed 10%:

$$\sum_{n \in N} |\epsilon_n| \leq 0.1 \tag{6.37}$$

Even though this additional constraint may be seen as of little relevance, it has in fact hindered the natural tendency of the MO approaches to extend the Pareto exploration to its very extremes. Furthermore, there is the practical issue of ANSP budget management, which makes high violations of absolute revenue values unacceptable in practice.

Finally, the maximum allowed average capacity breach is also bound not to exceed 35% of the total sectors/airports capacity:

$$\frac{\sum_{s \in S, h \in H} \frac{\alpha_s^{(h)}}{Q_{ss}^{(h)}} + \sum_{a \in A, h \in H} \left(\frac{\alpha_{a,dep}^{(h)}}{Q_{a,dep}^{(h)}} + \frac{\alpha_{a,arr}^{(h)}}{Q_{a,arr}^{(h)}} + \frac{\alpha_{a,gl}^{(h)}}{Q_{a,gl}^{(h)}} \right)}{\sum_{s \in S, h \in H} \alpha_s^{(h)} + \sum_{a \in A, h \in H} \left(\alpha_{a,dep}^{(h)} + \alpha_{a,arr}^{(h)} + \alpha_{a,gl}^{(h)} \right)} \leq 0.35 \tag{6.38}$$

At each iteration the GA generates a new population of individuals and for each of them in the evaluation phase the sets of Po_n and Pp_n are computed and used as inputs for the follower level (see Section 6.7.4).

6.9.1 Custom Solver Iteration

As shown in Figure 6.6, peak and off-peak rates for each ANSP are used as decision variables to run the custom solver written specifically to efficiently perform the minimum cost assignment. This minimum cost assignment is used to compute the following optimization output values, listed below, based on the analyzed assessment indicators explained in Section 6.8.1.7.

- *Total Shift* for all flights, defined as the sum of the difference (absolute) between the actual and the scheduled departure time.

- *Maximum Revenue Neutrality Violation*, defined as the absolute value of the largest difference between the charges computed based on the modulated rates and the charges computed based on the baseline unit rates as in force in September 2014. This is an objective of the optimization which needs to be minimized. In any case, the maximum revenue neutrality violation over all ANSPs must be lower than 10% (formulated as a constraint).

- *Number of Capacity Violations*, defined as the number of sector/airport-hour pairs that violate their nominal capacity levels.

- *Average Capacity Violation*, defined as the cumulative capacity violation, sum (over all sector/airport-hour pairs) of the number of flights that violate the nominal capacity, divided by the violated capacity.

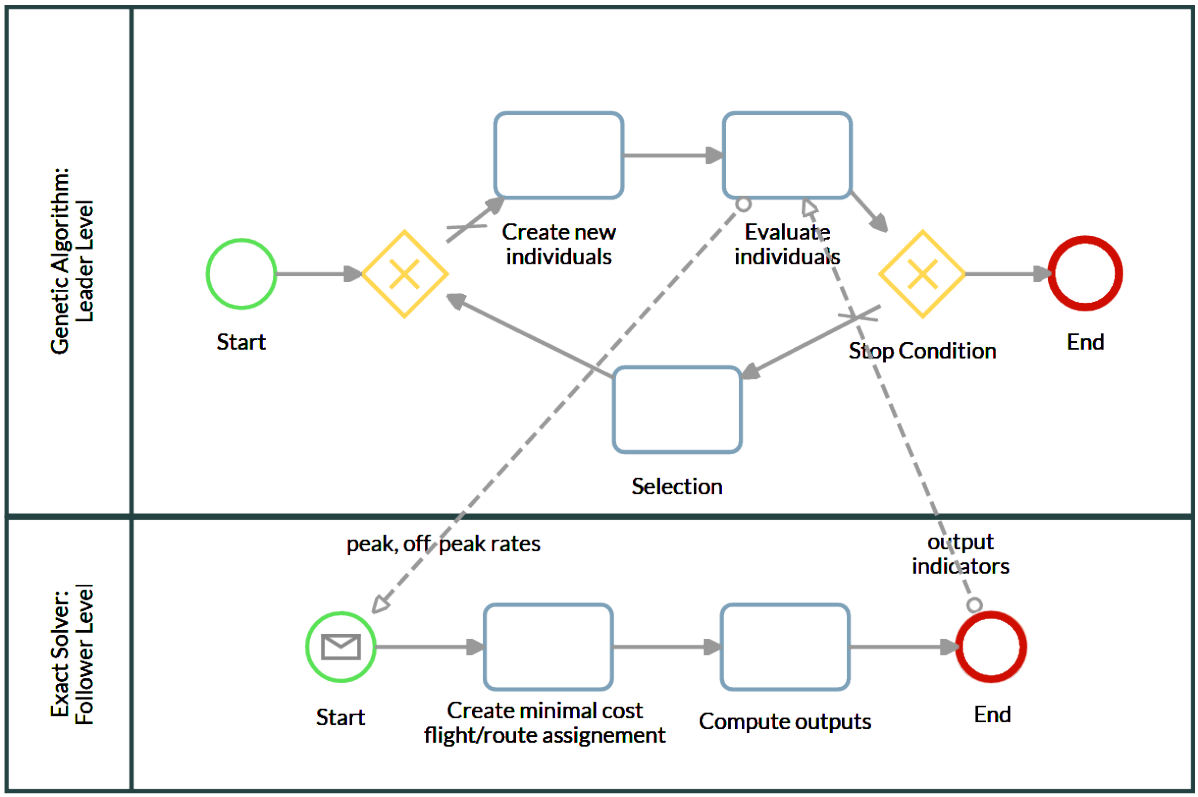


Figure 6.6: MOGASI nested approach for solving cPLP bi-level optimization problem

Some of the listed indicators are not used directly in the optimization problem formulation of the leader level, but have been deemed as useful for the post processing phase. Their computation, however, has not burdened the execution time of the follower's custom solver.

6.9.2 Baseline Creation

The historical traffic data used in this work are unfortunately unsuitable as baseline because they are composed of last filed flight plans which are made a few hours before the flight (i.e. in the tactical phase). This means that they are not known in the strategic phase so the routing decisions are make afterwards. For this reason a suitable baseline scenario has been created adopting the cPLP-one as anticipated in Section 6.7.6.3. In the cPLP model the baseline scenario is obtained by fixing all the charges at their respective historical unit rates and assigning each flight to its lowest cost trajectory (considering both operational costs and route charges). In other words, no modulation of the unit rates is performed at all. This solution of the model will be used as reference solution for comparison purposes and called *Baseline solution*.

6.10 Experimental Results

The run presented in this section was performed in a custom Java-based application on an desktop computer Intel(R) Core(TM) i7-4770 with 3.40 GHz CPU and 16 GB RAM. The MOGASI parameters used in this application are summarized in Table 6.7. In this case a proper tuning was not possible because of the long run time required for each instance and the unavailability of a high number of different instances that would be required for the tuning. In about 36 hours MOGASI generated approximately 100,000 solutions consisting in different combinations of peak and off-peak rates. 70,000 solutions were feasible, meaning that they respected all constraints.

6.10.1 General Analysis

The optimization correctly produced a Pareto front offering many trade-off solutions in terms of the two objectives satisfying imposed constraints, as set in Section 6.9. Nevertheless, after a rapid assessment it was decided to extend the post-processing result analysis to include all four indicators computed by the follower custom solver, as explained in Section 6.9.1. The reason is that this is the first time that an instance of such dimensions is solved, so a complete examination of the optimization indirect effects was deemed relevant.

Table 6.7: cPLP problem: MOGASI parameters

Description	Value
Population Size	80
Num.Generations	1250
Max Evaluations	100,000
Operators prob.	uniform
Auto scaled op.prob.	true
Initialization	single point
Replacement prob.	0.4
Ind. prob. distribution	cumulative
Q ind. distribution	0.1

In order to sift through the entire set of feasible solutions a general analysis has been conducted to further reduce their number and identify the most interesting ones. The Parallel Coordinates is a practical tool for obtaining a clearer picture of the impact that different solutions have on the chosen indicators. The tool is used for visualizing and analyzing high-dimensional and multi-variate data in predefined adjustable value ranges, spotting patterns in the objectives' behavior and correlations between them, and filtering out "bad" data. Dimensions (objectives) are represented by equally spaced parallel vertical lines, whereas each solution is represented by a colored polyline connecting vertices on the vertical axes.

The Figure 6.7 shows a Parallel Coordinates chart in four views, each with a different highlighted indicator (visible from the legend on the right of each view). It shows all 70,000 feasible solutions found by the MOGASI execution. The lines, i.e. single solutions, are colored according to the value in the highlighted indicator. There is a clear negative relation between Total Shift and the number of Violated Capacity: the less the flights are shifted, the higher the number of capacity violations. In fact the blue lines for Total Shift (low-value solutions) are all concentrated in the upper half of the Violated Capacity vertical line. Similarly, high-value solutions for Violated Capacity (red lines in the bottom-right view) are mainly concentrated in the low-value region of Total Shift. Another interesting fact that can be observed from these charts is the distribution of solutions with the Violated Capacity indicator highlighted, which is much more diverse than in any other indicator (color-wise). This means that to reach the two objectives the air traffic was spatially re-distributed in such way that the number of sectors/airports in which a violation occurs might have indeed increased, but the actual violation (quantified) has decreased, which can be observed in the average Capacity Violation over all sectors/airports. In fact, most solutions are "located" in the lower half of the vertical line. The same consideration, i.e. the existence of a correlation between the average Capacity Violation and the number of Violated Capacity, can also be made by analyzing the

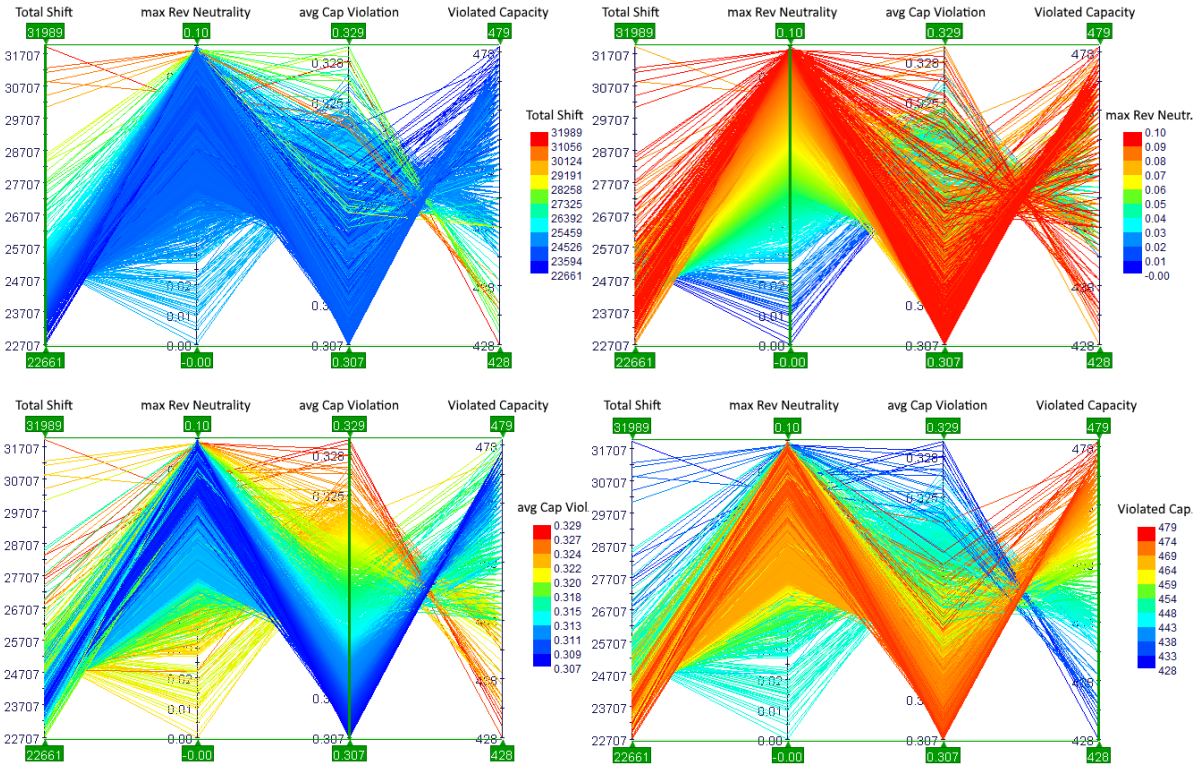


Figure 6.7: cPLP - Analysis of feasible solutions on the Parallel Coordinates chart

bottom-left view with the average capacity violation as highlighted indicator.

As far as the maximum Revenue Neutrality violation is concerned, all solutions plotted on the charts respect the upper limit of 10% even though the majority are in the upper half of its bounds, included between 5-10%. This is a clear example of trade-off because it was not possible to bring this value any closer to 0, like in the baseline solution, without compromising other indicator and objective values.

Moreover, the max Revenue Neutrality violation is well distributed over the entire permitted range. These fluctuations indicate that this is the key indicator, the manipulation of which enables the minimization of the Total Shift, as well as a drastic reduction of the average Capacity Violations. This can be easily observed in the Figure 6.7 on the top-right view in the form of the red stack of designs which has approximately the same width in all but the last indicator. The higher the Revenue Neutrality violation, the more the maneuvering space for the adjustment of other indicators. Naturally, if the revenue neutrality violation were fixed, for example to 0 as in the baseline solution, the flight route flexibility would be reduced close to 0. This in fact is the clear indicator that the proposed pricing system can successfully achieve a good redistribution of flights thanks to the margin created by the

Table 6.8: cPLP - Baseline and selected best solutions

	Total Shift	**Max Rev Neutrality**	**Violated Capacity**	**avg Cap Violation**
Baseline	25,201	0	447	0.32285
Solution (S1)	22,707	0.083709	477	0.31964
Solution (S2)	23,286	0.098726	465	0.31280
Solution (S3)	24,644	0.083839	448	0.32079
Solution (S4)	27,617	0.08691	430	0.32880

possibility to violate the revenue neutrality within the imposed constraints.

The analysis highlighted a number of equally valid alternatives, but to better assess the impact of this new proposed pricing scheme, four solutions have been chosen from very different regions of the Pareto for analysis purposes. They are reported in the Table 6.8.

6.10.2 Pareto Inspection

The Scatter chart presented in Figure 6.8 shows a collection of selected Pareto solutions (green squares) in terms of two indicators to be minimized: the objective Total Shift on X axis and the constraint Cumulative Capacity Violation on Y axis. MOGASI identified a number equally good alternatives for both indicators. The baseline solution, computed using the historical unit rates, is marked in the scatter chart as a red square and labeled *S0*.

The Parallel Coordinates chart on Figure 6.9 shows the four selected solutions plus the baseline, represented with the red polyline. Referring to the notation of Table 6.8 the selected solutions are represented as: S1 – dark blue, S2 – green, S3 – light blue, S4 – purple.

It is clear than none of the solutions has desired values in all objectives, but they rather represent equally valid trade-offs between opposing requirements, achieved by simply re-distributing air traffic in different ways. The baseline solution exhibits medium values for all plotted indicators, whereas the Revenue Neutrality is by definition perfectly matched for all ANSPs, i.e. equal to 0. This solution, however, cannot be considered as the optimal solution because the goal is to balance all objectives while keeping the number of Violated Capacity and the maximum Revenue Neutrality as low as possible, and in any case under a pre-defined threshold.

Owing to the dedicated constraint, all solutions keep the maximum Revenue Neutrality under 10%, with the S2 solution barely under this threshold. None, however, manages to go lower than approximately 8% (with S1 being the lowest), but this is a necessary trade-off in order to keep the other values on the Pareto front. S1 and S4 solutions are on the opposite extremes of the Pareto, so it may be expected that they are representative of two different reactions of the model to avoid congestion. S1 keeps the Total Shift very low, which in turn

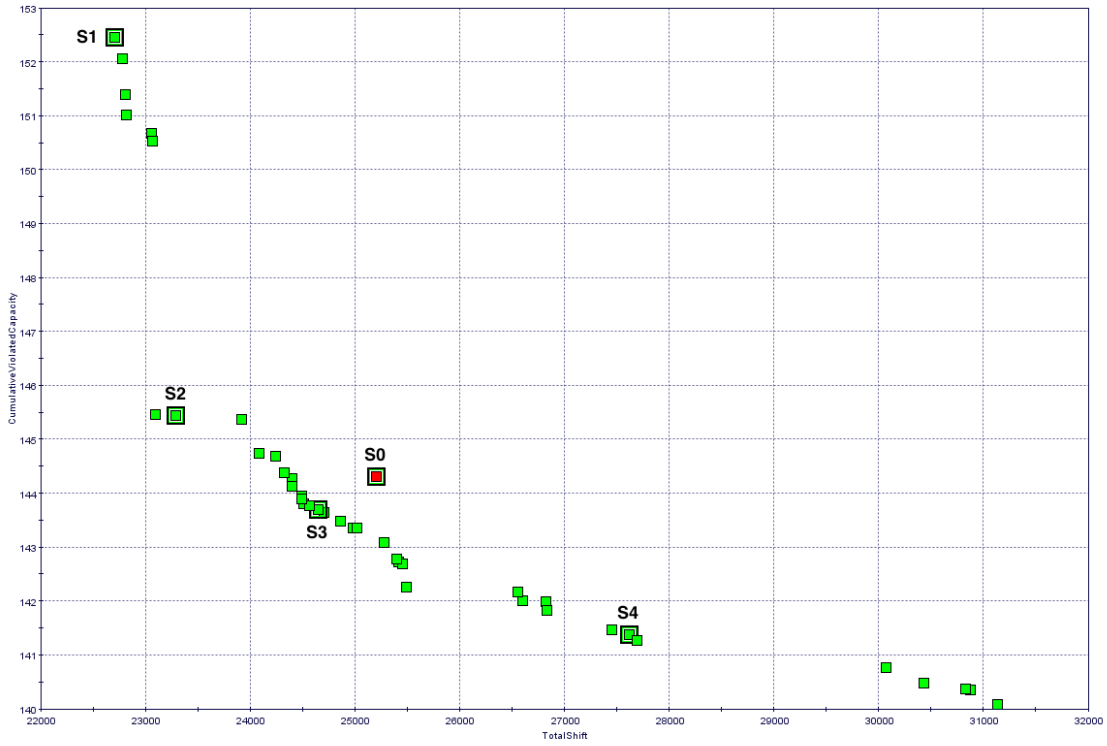


Figure 6.8: cPLP - Alternative Pareto for Cumulative Capacity Violation vs Total Shift

results in a higher number of Violated Capacity. This indicates that the flights have been spatially redistributed route-wise to respect as much as possible their departure and arrival times and keep the average Capacity Violation low. S4, on the contrary, uses aircraft shifting as means to lower the number of capacity violations to a minimum at the expense of the average Capacity Violation: this means that the number of sectors/airports in which the violations occur is low, but the amount of violation they produce is high. It is interesting to note how these entirely opposing reaction produce virtually the same financial effect in terms of maximum Revenue Neutrality Violation on the ANSPs, which is almost the same. Moreover, S2 has the lowest average Capacity Violation, indicating that the flights have been spatially well distributed, which is confirmed by a very low Total Shift and high number of Violated Capacity. Since this solution has the highest value of the maximum Revenue Neutrality objective on the Pareto Front, among those presented, it emphasizes the key role of the feasibility threshold of this objective in the maneuverability of the model to avoid congestion. In other words, if this threshold were higher it is probable that further solutions would be obtained with better values in all other objectives. With this in mind, a definition of an acceptable threshold by the European Union would enable the transformation of this

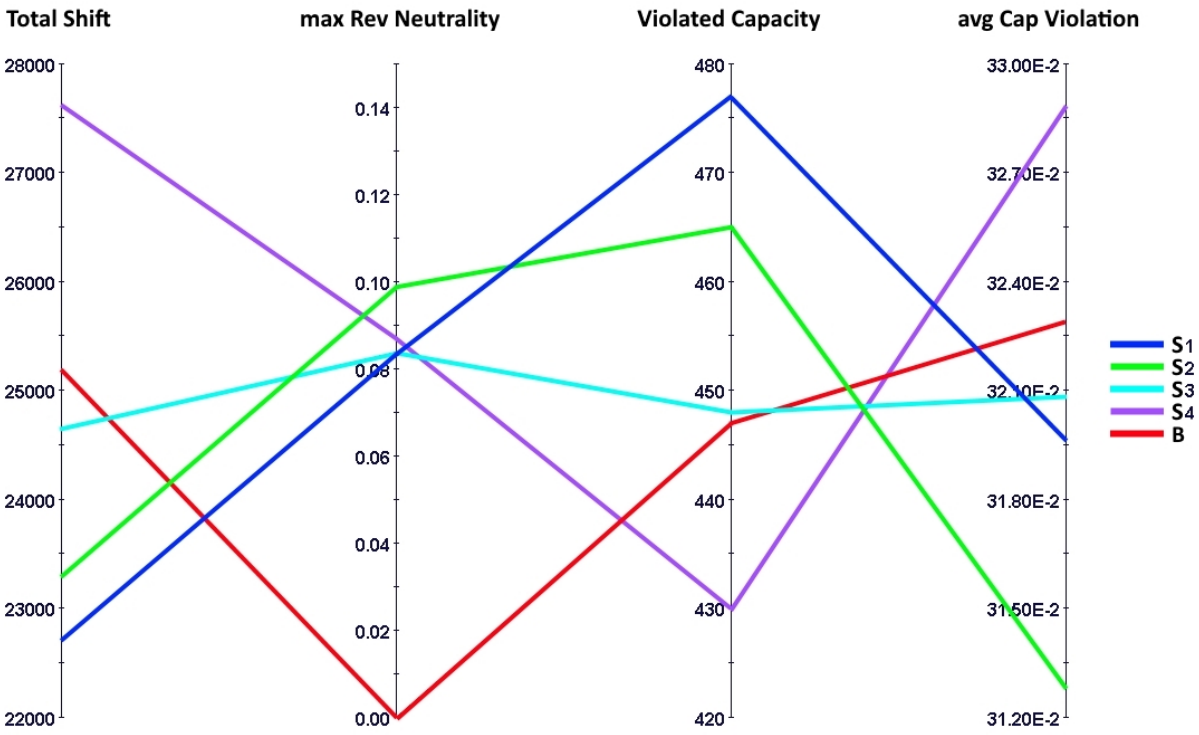


Figure 6.9: Trade-offs among four Pareto solutions and the baseline solution

objective into a constraint leaving the possibility to choose an other assessment indicator for the GA bi-objective formulation. Finally, S3 is the most balanced solution because, considering the ranges for all indicators resulting from these four solutions, its values are in the middle. With respect to baseline, the number of Violated Capacity of S3 is very similar, but both the Total Shift and average Capacity Violation are lower at the expense of the max Revenue Neutrality which is at similar levels as other selected solutions. This solution might be considered as the less disruptive.

6.10.3 Achieved Rate Modulation

In this section a further analysis on the peak and off-peak rates that make the favorable traffic redistributions possible is presented. Figure 6.10 shows peak and off-peak rates for the four selected solutions only for cases which deviate for more than 4 euros from the September 2014 unit rates. Imposed by SATURN project, this analysis is commented in an anonymous form in which the actual ANSP have been replaced by generic names such as Country1 ($C1$), Country2 ($C2$) and so forth. The original unit rate is represented as value 0, for the real values refer to Table 6.6. For some states (such as $C1$, $C3$ and $C8$) both peak and off-peak

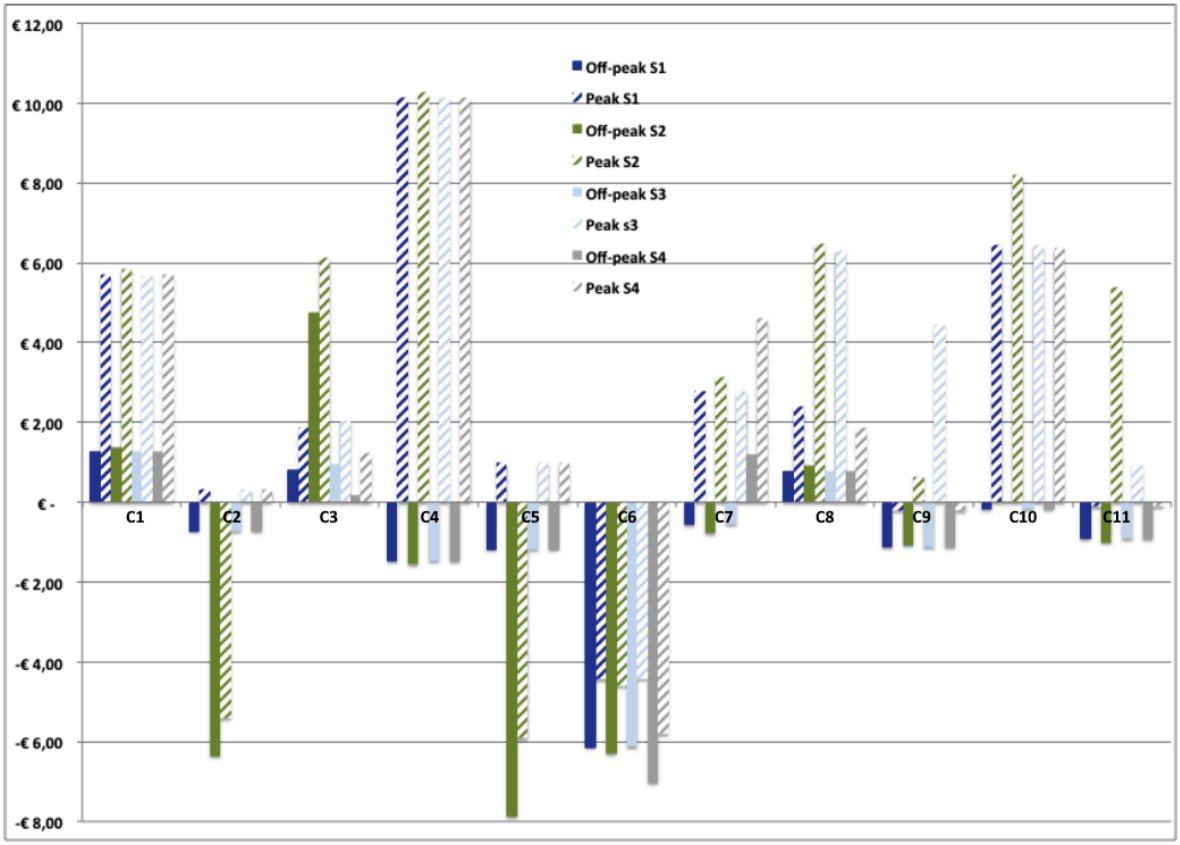


Figure 6.10: Peak and off-peak rates for a selected ANSPs subset

rates are higher than historical rates in all solutions. This can be interpreted in different ways: for example, it may mean that these countries have a rather congested airspace and the increase of rates is meant to discourage AUs from using routes which pass through these countries. Another explanation could be that the rate modulation in neighboring countries has resulted in less traffic passing through $C1$, $C3$ and $C8$. This means that these particular ANSPs are forced to increase their charges to maintain revenue neutrality with less flights.

Another interesting case is the $C4$ where the off-peak rates are always lower than the unit rate and the peak rates are always higher than the unit rate. This can indicate that there is much maneuvering space for the modulation of charges within the limits imposed by the European ATM regulations, which can lead to reduced congestion without deteriorating flight efficiency and flight operational costs.

In other cases, such as for $C6$, the opposite occurs, i.e. all charges are lower than the unit rate, which means that this ANSP may use them as a way to attract more airspace users or a way to re-distribute air traffic from a very congested neighbor (such as $C12$, which

historically has one of the highest capacity violations). In fact, $C12$ has some of the greatest differences between the unit rate and the computed peak and off-peak rates, even up to 30-50 euros, as can be observed from the Figure 6.11. However, the difference between the peak and the off-peak rate for each solution is very small. This may mean, for example, that in spite of the charge modulation the $C12$ air space has remained congested, so the peak rate has been adjusted (solution S4) to avoid negative consequences of an overly high peak rate for the revenue neutrality. Another possible explanation, for instance, is that the off-peak rate has been increased to the levels of the peak rate to decongest the air space and encourage AUs to choose different routes. An analogous logic can be applied in case of the solution S1, which is the opposite of S4. A more detailed analysis considering the entire European context are required to explain these differences among the selected solutions and it may represent the object of a future work to gain a clear and comprehensive insight into the European air traffic pricing mechanisms and the balancing of different tariffs over different ANSPs.

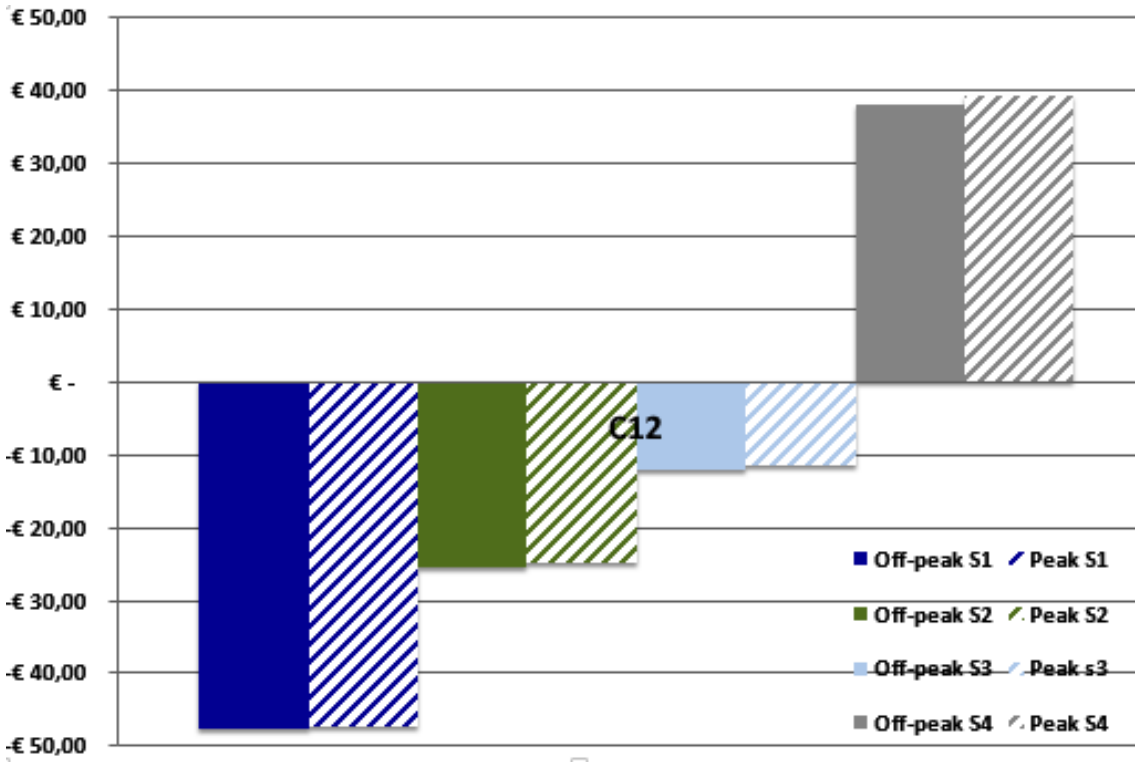


Figure 6.11: Peak and off-peak rates for Country12 (C12)

Table 6.9: Number of flights using different routes between two solutions

	Baseline	**S1**	**S2**	**S3**	**S4**
Baseline	0	858	550	258	541
S1	-	0	525	642	1236
S2	-	-	0	320	936
S3	-	-	-	0	644
S4	-	-	-	-	0

Table 6.10: Number of flights with differences in the number of shifted flights

	Baseline	**S1**	**S2**	**S3**	**S4**
Baseline	0	1000	685	308	643
S1	-	0	629	744	1420
S2	-	-	0	412	1114
S3	-	-	-	0	740
S4	-	-	-	-	0

6.10.4 Achieved Traffic Redistribution

Based on the trajectories assigned to each flight that were used for calculating the baseline solution and the optimization results, further calculations have been performed for comparison purposes. Table 6.9 shows the number of routes that differ between each pair of solutions, including the baseline (spatial redistribution). The largest difference (1,236 routes) can be observed between solutions S1 and S4, which, as said previously, lie in entirely opposite regions of the Pareto front. S1 solution is also the solution that differs mostly from the baseline. The solution that differs least (258 routes) from baseline is S3, which shows least difference also in other aspects (see Parallel Coordinates chart on Figure 6.9). The optimized solutions with the lowest number of route differences are S2 and S3 (320). If the absolute values are observed, the solution S1 undertakes the smallest number of routes, whereas S4 is predictably the largest.

The situation is somewhat similar as regards the differences in the number of shifted flights (temporal redistribution) shows in Table 6.10. Shifted flights in this context is intended as flights with a changed departure and/or arrival time based on the changed flight trajectory. Again, the largest difference is between solutions S1 and S4 (1,420), with S1 differing from baseline by 1,000 shifted flights. S3 is the solution most similar to baseline.

The Figure 6.12 shows a visual example of a reassigned route to a flight due to the modulation of the rates, i.e. the spatial redistribution for the Rome Fiumicino (LIRF) –

Helsinki (EFHK) flight. The red route is the baseline solution whereas the light blue line represents the route chosen under solution S3. This is a simple example of a spatial and temporal traffic redistribution achieved as a result of the route modulation mechanism.

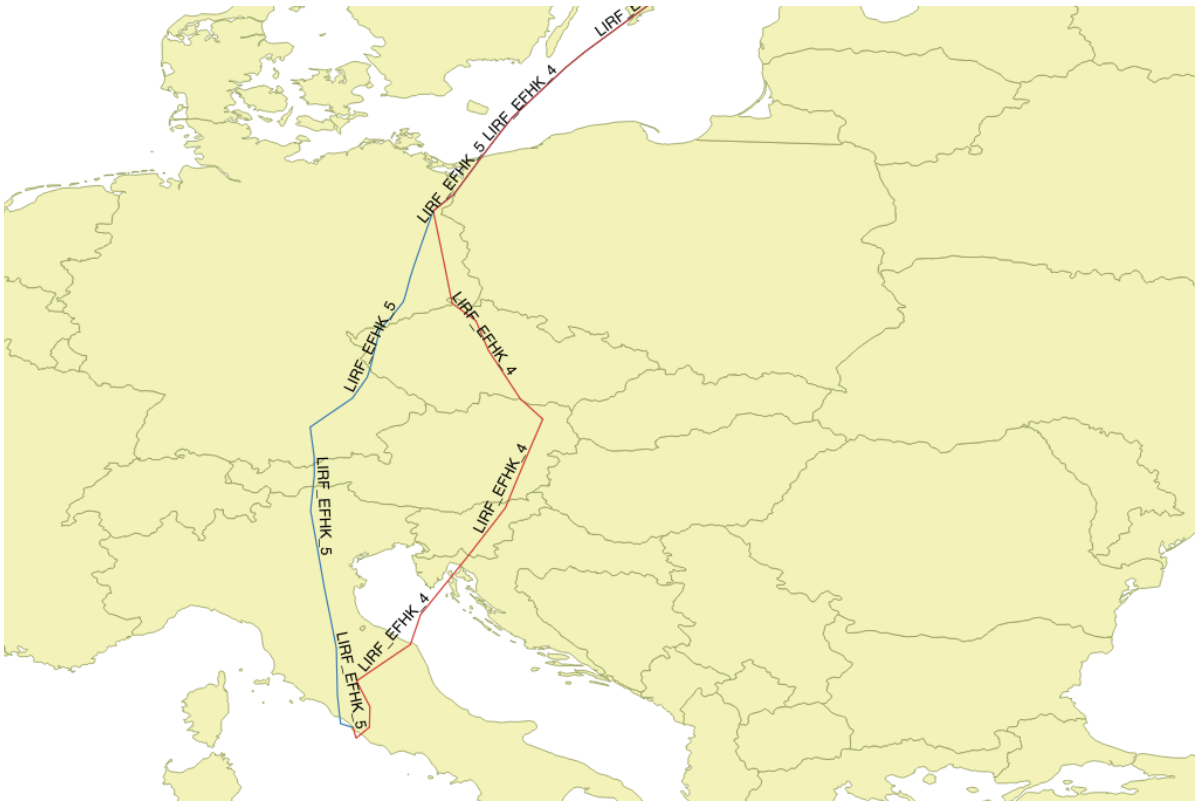


Figure 6.12: Example of the effect of charge modulation on LIRF-EFHK

6.10.5 cPLP application outcome

The case study presented in this chapter was motivated after having obtained satisfying results in terms of data quality and low computational costs of the application of MOGASI to large-scale bi-level optimization problems (see Chapter 5). For this reason this methodology was extended to the problem of European air traffic congestion in the framework of the SATURN project.

The chosen day of operations was 12th September 2014, which was one of the busiest days of the year, but it was not disrupted by any unusual or unforeseeable events. This day's data also did not show a particularly high levels of congestion in the network, with the exception of very few sectors and time slots. This means that for most of the day and for most of the airspace the demand was below the nominal capacity.

The application results show that the modulation of en-route charges indeed redistributes the European air traffic. The implemented bi-level approach identified a wide range of possible solutions (see Section 6.10.1). Trade-offs between the total shift, satisfaction of capacities and revenue neutrality constraints have been necessary (see analysis in 6.10.2). MOGASI managed to find solutions quite different from the baseline scenario since a modulation of en-route charges enables the reduction of the sector load and the total shift without increasing the AUs' operational costs (see Section 6.10.3).

Future work will be the application of the PLP in a decentralised manner, thus having ANSPs set the modulations instead of the CP. This will enable to try the MOGASI framework in the context of the co-evolutionary approach for multi-level optimization.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The main contribution of this thesis is the application of metaheuristic algorithm, taken from the field of Evolutionary Algorithms (EAs), to specific problems, and the adaptation of its behavior to the characteristics of the problem.

This idea was coded in the algorithm named Multi-Objective Genetic Algorithm for Structured Inputs (MOGASI). Among other characteristics, particular attention was given to the study of specific methods for handling information extracted from linear constraints. A further analysis focused on the issue of how the heuristics belonging to the family of Genetic Algorithms (GAs) could tackle problems with equality constraints, which are known to be difficult for them. The main idea came from the linear programming logic and consisted in the inclusion of information on variables and constraints in the GA. It created the possibility of using other mechanisms, such as the pre-processing of problems, in the algorithm. This is possible because the functions of the solution creation, i.e. the GA operators, are able to use the collected additional information. The issue of equalities, on the other hand, was solved by introducing a strategy for the reduction of the variables. It enabled the creation of a smaller problem search space that was entirely feasible only for the eliminated constraints. Update and pre-processing actions resulted in a considerable reduction in search space, as discussed in Chapter 3. MOGASI was validated on a large number of well-known benchmark problems of different kinds. The goal was to verify the performance of MOGASI by comparing it with other standard GAs known in literature.

In future work, heavy testing of the specialized components of MOGASI will be performed using selected problems, e.g. problems with linear equality constraints. The number of those

components will be expanded in order to include a wider variety of structures that the algorithm can handle. Furthermore, the performance of this approach will be studied in detail on mixed-integer and combinatorial problems.

After the algorithm validation through extensive benchmarking, a study of a class of NP-hard problems, the Bi-level Programming Problems (BPPs), was conducted. Within this class, the focus was on Pricing Problems. These problems are well-known in literature and reflect the real-world problems in which an entity (e.g. a company) is confronted with a best-price strategy issue. Entities want to determine the price of a set of products in order to maximize revenue. The reaction of potential customers has to be taken into account: if prices are too high, they may decide not to buy the products, whereas if prices are too low, the company loses revenue. The economic principle of the Stackelberg equilibrium reflects this situation, where a balance within a hierarchy involving two players is sought which requires specialized strategies to tackle this problem using metaheuristics.

Following an overview of the most promising strategies for the application of heuristics to BPPs proposed in literature, the nested solution strategy was chosen. It consists in a hierarchical optimization where the levels are solved sequentially by improving solutions on each level to obtain a good overall solution on both levels. It was necessary to develop an algorithm dedicated for the inner level which would be able to sequentially tackle the problems of the targeted applications and thus enable the use of the nested approach.

The first application focused on a specific pricing problem called "Network Pricing Problem" with connected toll arcs (Highway Problem). In this problem, the toll arcs were connected so as to constitute a single path with pairs of entry and exit nodes and a toll associated to the sub-path delimited by each of these pairs. Given that the problem was rather difficult and the existing exact approaches were not able to solve large instances within a reasonable amount of time, it was considered to be an appropriate application field.

A large number of instances of different sizes and with different characteristics were generated for benchmarking purposes. The idea was to identify a promising area, or at least a niche, in which a nested metaheuristic such as MOGASI can compete against or even surpass the performance of traditional linear solvers.

A specific promising area was found for the MOGASI approach with respect to the exact solver. In fact, the results presented in this thesis show that MOGASI was outperformed by the exact solver in small size instances but the situation was inverted in favor of MOGASI with the increase of the instance size. In particular, for one of the largest instance sizes it was possible to state that the application of the MOGASI nested approach yielded results of the same quality levels as those obtained with the exact solver in at least one order of

magnitude of time less. However, due to the stochastic nature of the GA, this performance cannot always be guaranteed. Considering that the average execution time of the case with the longest run duration was approximately a fifth of the time required by the exact solver, it was deemed more than acceptable to perform repeated runs in search for the optimum. Future work will focus on the dynamics connected to large scale instances to delimit more precisely the promising application area for MOGASI in case of such instances. A further intention is to develop new specialized MOGASI modules for integer problems given that the current implementation does not have any specialized strategy for handling them, despite the results achieved in this application.

The second application focused on Peak-load Pricing (PLP) which is a pricing mechanism commonly used in transport and utilities. PLP was evaluated in the framework of the SATURN European project as the most appropriate strategy for reducing network congestion in European Air Traffic Management (ATM). In particular, a centralized approach to PLP (cPLP) where a Central Planner (CP) sets en-route charges on the network is presented. Such charges should guarantee that Air Navigation Service Providers (ANSPs) are able to recover their operational costs while inducing the Airspace Users (AUs) to route their traffic in a configuration that the network is able to sustain. The interaction between CP and AUs was modeled as a Stackelberg game and formulated by means of bi-level linear programming. In this case, a specific algorithm for the solution of the lower-level problem was also developed and the MOGASI optimization framework was extended to tackle the cPLP. A large scale instance was created based on real air traffic data over the entire European airspace collected on 12th September 2014. This was one of the busiest day of the year and it was selected because it was not disrupted by any unusual events. Results has shown the effective functioning of the cPLP strategy since it was capable of redistributing the air traffic and mitigating congestion over the entire European air space preserving the existing revenue neutrality with a certain tolerance. Moreover, they showed that significant improvements could be achieved in traffic distribution in terms of both delay and sector load through this simple en-route charge modulation scheme. Finally, this case showed a successful application of MOGASI to a bi-level problem with multiple objectives at the upper-level.

Future work will look into different aspects of cPLP applications, in particular, a detailed analysis of revenue distribution and airspace use across ANSPs. The reason for this is that, despite the positive results, the structure of the found Pareto front is deemed fragile and extremely sensitive to differences in traffic volumes that each ANSP is responsible for. This shortcoming could be balanced by the introduction of more elaborate definitions of objective functions, in particular, the revenue function that should be redefined based on

different ANSP categories.

In the framework of the SATURN European project, the fact that the ANSPs set the modulations, and not the Central Planner, is of considerable relevance. The preliminary studies showed that such a structure would not require a nested approach, but rather a co-evolutionary approach. It is a framework extension that could open up new research applications for MOGASI.

Future work will also consider the multi-objective bi-level optimization in forms that are not explored in this thesis: problems with multiple objectives also in the lower-level. This branch of study has recently received increased attention and it looks promising for the future application of GAs.

BIBLIOGRAPHY

H. A. Abbass and K. Deb. Searching under multi-evolutionary pressures. In *Evolutionary Multi-criterion Optimization*, volume 2632, pages 391–404. Springer, 2003.

U. Aickelin. An Indirect Genetic Algorithm for Set Covering Problems. *The Journal of the Operational Research Society*, 53(10):1118–1126, 2002.

G. Andreatta and A. Odoni. Preliminary investigation on market-based demand management strategies for European airports and air routes. Report for EUROCONTROL Experimental Centre. 2001.

J. S. Angelo, E. Krempser, and H. J. C. Barbosa. Differential Evolution for Bilevel Programming. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 470–477, 2013.

T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

D. Beasley, D. R. Bull, and R. R. Martin. An overview of genetic algorithms : Part 1, fundamentals. *University Computing*, 2(15):1–16, 1993.

D. Bertsimas, G. Lulli, and A. Odoni. An Integer Optimization Approach to Large-Scale Air Traffic Flow Management. *Operations Research*, 59(1):211–227, 2011.

M. Bhattacharya. An informed operator approach to tackle diversity constraints in evolutionary search. In *International Conference on Information Technology: Coding and Computing*, volume 2, pages 326–330, 2004.

T. Bolić, L. Castelli, and D. Rigonat. Peak-load pricing for the European Air Traffic Management system using modulation of en-route charges. *European Journal of Transport and Infrastructure Research*, 17(1):136–152, 2017.

S. Borenstein, M. Jaske, and A. Rosenfeld. Dynamic Pricing, Advanced Metering, and Demand Response in Electricity Markets. Technical report, Center for the Study of Energy Markets, 2002.

M. Bouhtou, S. Van Hoesel, A. F. Van der Kraaij, and J. L. Lutton. Tariff optimization in networks. *INFORMS Journal on Computing*, 19(3):458–469, 2007.

I. Boussaïd, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

J. Bracken and J. T. McGill. Mathematical Programs with Optimization Problems in the Constraints. *Operations Research*, 21(1):37–44, 1973.

L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard. A Bilevel Model and Solution Algorithm for a Freight Tariff-Setting Problem. *Transportation Science*, 34(3):289–302, 2000.

L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard. A bilevel model for toll optimization on a multicommodity transportation network. *Transportation Science*, 35(4):345–358, 2001.

L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard. Joint Design and Pricing on a Network. *Operations Research*, 56(5):1104–1115, 2008.

L. T. Bui, H. Abbass, and J. Branke. Multiobjective optimization for dynamic environments. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 3, pages 2349–2356. IEEE, 2005.

H. I. Calvete, C. Galé, and P. M. Mateo. A new approach for solving linear bilevel problems using genetic algorithms. *European Journal of Operational Research*, 188(1):14–28, 2008.

W. Candler. A linear bilevel programming algorithm: A comment. *Computers & Operations Research*, 15(3):297–298, 1988.

W. Candler and R. Norton. Multi-level Programming. *Development Research Department discussion paper no. DRD 20*, pages 1–53, 1977.

E. Cantú-Paz. A Survey of Parallel Genetic Algorithms. *Calcul. Paralleles Reseaux Syst. Repart.*, 10(2):141–171, 1998.

J. Cardinal, E. D. Demaine, S. Fiorini, G. Joret, I. Newman, and O. Weimann. The Stackelberg minimum spanning tree game on planar and bounded-treewidth graphs. *Journal of Combinatorial Optimization*, 25(1):19–46, 2013.

B. P. Carlson and D. F. Hougen. Phenotype feedback genetic algorithm operators for heuristic encoding of snakes within hypercubes. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 791–798. ACM Press, 2010.

L. Castelli, M. Labbé, and A. Violin. A Network Pricing Formulation for the revenue maximization of European Air Navigation Service Providers. *Transportation Research Part C: Emerging Technologies*, 33:214–226, 2013.

P. A. Clark and A. W. Westerberg. A note on the optimality conditions for the bilevel programming problem. *Naval Research Logistics (NRL)*, 35(5):413–418, 1988.

B. Colson, P. Marcotte, and G. Savard. Bilevel programming: A survey. *4OR*, 3(2):87–107, 2005.

A. J. Cook. *European air traffic management: principles, practice, and research*. Ashgate Publishing, Ltd., 2007.

A. J. Cook and G. Tanner. European airline delay cost reference values. Technical report, University of Westminster, 2011.

A. J. Cook and G. Tanner. European airline delay cost reference values, 2015 update. Technical report, University of Westminster, 2015.

S. Costanzo, L. Castelli, and A. Turco. MOGASI: A multi-objective genetic algorithm for efficiently handling constraints and diversified decision variables. In *Engineering Optimization 2014*, pages 99–104. CRC Press/Balkema, 2014.

R. Courant. Variational Methods for the Solution of Problems of Equilibrium and Vibrations. *Bulletin of the American Mathematical Society*, 49(1):1 – 23, 1943.

Y. Davidor. Epistasis Variance: A Viewpoint on GA-Hardness. In *Foundations of Genetic Algorithms*, volume 1, pages 23–35. Morgan Kaufmann, 1991.

K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

K. A. De Jong and W. M. Spears. Using genetic algorithms to solve NP-complete problems. In *Proceedings of the 3rd international conference on Genetic algorithms*, number 7, pages 124–132, 1989.

P. L. De Matos. Yield management for privatised air traffic control? *Journal of the Operational Research Society*, 52(8):888–895, 2001.

A. De Palma and R. Lindsey. Traffic congestion pricing methodologies and technologies. *Transportation Research Part C: Emerging Technologies*, 19(6):1377–1399, 2011.

K. Deb. An introduction to genetic algorithms. *Sadhana*, 24(4-5):293–315, 1999.

K. Deb and D. E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 42–50, 1989.

K. Deb and J. Sachin. Running performance Metrics for evolutionary multi-objective optimization. *Kangal Report 2002004*, pages 13–20, 2002.

K. Deb and A. Sinha. An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary-local-search algorithm. *Evolutionary computation*, 18(3):403–49, 2010.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

L. Delgado. European route choice determinants. In *Proceedings of the 11th USA/Europe Air Traffic Management Research and Development Seminar*, 2015.

A. Della Cioppa, C. De Stefano, and A. Marcelli. On the role of population size and niche radius in fitness sharing. *IEEE Transactions on Evolutionary Computation*, 8(6):580–592, 2004.

A. Della Cioppa, C. De Stefano, and A. Marcelli. Where are the niches? Dynamic fitness sharing. *IEEE Transactions on Evolutionary Computation*, 11(4):453–465, 2007.

S. Dempe. *Foundations of bilevel programming*, volume 61. Kluwer Academic Publishers, 2002.

S. Dempe and J. Dutta. Is bilevel programming a special case of a mathematical program with complementarity constraints? *Mathematical Programming*, 131(1):37–48, 2012.

S. Dempe, V. Kalashnikov, G. A. Pérez-Valdés, and N. Kalashnykova. *Bilevel Programming Problems*. 2015.

K. Deschinkel, J. L. Farges, and D. Delahaye. Optimizing and assigning price levels for air traffic management. *Transportation Research Part E: Logistics and Transportation Review*, 38(3):221–237, 2002.

S. Dewez, M. Labbé, and P. Marcotte. New formulations and valid inequalities for a bilevel pricing problem. *Operations Research Letters*, 36(2):141–149, 2008.

A. E. Eiben and C. A. Schippers. On Evolutionary Exploration and Exploitation. *Fundamenta Informaticae*, 35(1):35–50, 1998.

A. E. Eiben and J. E. Smith. What is an Evolutionary Algorithm? *Introduction to Evolutionary Computing*, pages 15–35, 2003.

A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Berlin Heidelberg, 2015.

L. J. Eshelman. The CHC Adaptive Search Algorithm: How to Safe Search When Engaging in Nontraditional Genetic Recombination. *Foundations of genetic algorithms*, pages 265–283, 1991.

Eurocontrol. Conditions of application of the route charges system and conditions of payment, Central Route Charges Office. Technical report, 2011.

Eurocontrol. DDR2 Reference Manual v.1.0.1. Technical report, 2014a.

Eurocontrol. European Network Operations Plan 2014-2018/19. Technical report, 2014b.

Eurocontrol. Principles for establishing the cost-base for en route charges and the calculation of the unit rates. Technical report, 2015a.

Eurocontrol. Performance Review Report: An Assessment of Air Traffic Management in Europe during the Calendar Year 2014. Technical report, 2015b.

Eurocontrol. Standard Inputs for EUROCONTROL Cost-Benefit Analyses. Technical report, 2015c.

Eurocontrol. ATM Cost-Effectiveness (ACE) 2013 Benchmarking Report with 2014-2018 outlook Prepared by the Performance Review Unit (PRU) with the ACE Working Group. Technical report, 2015d.

European Commission. Commission Regulation (EC) No 1794/2006 of 6 December 2006, laying down a common charging scheme for air navigation services. Technical report, 2006.

M. Falkner, M. Devetsikiotis, and I. Lambadaris. An overview of pricing concepts for broadband IP networks. *IEEE Communications Surveys Tutorials*, 3(2):2–13, 2000.

W. Fan and X. Jiang. Tradable mobility permits in roadway capacity allocation: Review and appraisal. *Transport Policy*, 30:132–142, 2013.

R. Farmani and J. A. Wright. Self-Adaptive Fitness Formulation for Constrained Optimization. *IEEE Transactions on Evolutionary Computation*, 7(5):445–455, 2003.

J. Fliege and L. N. Vicente. Multicriteria approach to bilevel optimization. *Journal of Optimization Theory and Applications*, 131(2):209–225, 2006.

C. M. Fonseca and P. J. Fleming. Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms,ÄìPart I: A Unified Formulation. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 28(1):26–37, 1998.

S. Forrest and M. Mitchell. Relative building-block fitness and the building-block hypothesis, 1993.

T. Friedrich, N. Hebbinghaus, and F. Neumann. Rigorous analyses of simple diversity mechanisms. *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, page 1219, 2007.

I. I. Garibay and A. S. Wu. Cross-fertilization between proteomics and computational synthesis. In *AAAI Spring Symposium Series*, pages 67–74, 2003.

M. S. Gibbs, G. C. Dandy, and H. R. Maier. A genetic algorithm calibration method based on convergence due to genetic drift. *Information Sciences*, 178(14):2857–2869, 2008.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986.

D. E. Goldberg. Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. *Complex Systems*, 3:129–152, 1989a.

D. E. Goldberg. Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis. *Complex Systems*, 3:153–171, 1989b.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., 1989c.

D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69–93, 1991.

D. E. Goldberg and M. Rudnick. Genetic Algorithms and the Variance of Fitness, 1991.

D. E. Goldberg, K. Deb, and J. H. Clark. Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems*, 6(4):333–362, 1991.

S. Gotshall and B. Rylander. Optimal population size and the genetic algorithm. *Proceedings On Genetic And Evolutionary Computation Conference*, pages 1–5, 2000.

J. J. Grefenstette. Deception Considered Harmful. *Foundations of Genetic Algorithms 2*, pages 75–91, 1992.

P. Hansen, B. Jaumard, and G. Savard. A new branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 5(13):1194–1217, 1992.

G. Heilporn. *Network pricing problems: complexity, polyhedral study and solution approaches*. PhD thesis, Université Libre de Bruxelles and Université de Montreal, 2008.

G. Heilporn, M. Labbé, P. Marcotte, and G. Savard. A parallel between two classes of pricing problems in transportation and marketing. *Journal of Revenue and Pricing Management*, 9(1):110–125, 2010a.

G. Heilporn, M. Labbé, P. Marcotte, and G. Savard. A polyhedral study of the network pricing problem with connected toll arcs. *Networks*, 55(3):234–246, 2010b.

G. Heilporn, M. Labbé, P. Marcotte, and G. Savard. Valid inequalities and branch-and-cut for the clique pricing problem. *Discrete Optimization*, 8(3):393–410, 2011.

J. Holland. *Adaption in Natural and Artificial Systems*, volume 11. University of Michigan Press, 1975.

J. Holland. *Adaptation in natural and artificial systems*. M.I.T.P., 1992.

A. Homaifar, C. X. Qi, and S. H. Lai. Constrained Optimization Via Genetic Algorithms. *SIMULATION*, 62(4):242–253, 1994.

H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *IEEE Congress on Evolutionary Computation*, pages 2419–2426, 2008.

R. G. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, 32:146–164, 1985.

J. Joines and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pages 579–584, 1994.

G. Joret. Stackelberg network pricing is hard to approximate. *Networks*, 57(2):117–120, 2011.

R. Jovanović, V. Tošić, M. Čangalović, and M. Stanojević. Anticipatory modulation of air navigation charges to balance the use of airspace network capacities. *Transportation Research Part A: Policy and Practice*, 61:84–99, 2014.

V. Kalashnikov, S. Dempe, G. A. Pérez-Valdés, N. I. Kalashnykova, and J. F. Camacho-Vallejo. Bilevel programming and applications. *Mathematical Problems in Engineering*, pages 1–16, 2015.

V. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *Evolutionary Multi-Criterion Optimization*, pages 376–390. Springer, 2003.

J. D. Knowles, R. A. Watson, and D. W. Corne. Reducing local optima in single-objective problems by multi-objectivization. In *Evolutionary multi-criterion optimization*, pages 269–283. Springer, 2001.

A. Koh. Solving transportation bi-level programs with differential evolution. In *2007 IEEE Congress on Evolutionary Computation, CEC 2007*, pages 2243–2250, 2007.

M. Labbé and A. Violin. Bilevel Programming and Price Setting Problems. *4OR quarterly journal of the Belgian, French and Italian Operations Research Societies*, 11(1):1–30, 2013.

M. Labbé, P. Marcotte, and G. Savard. A Bilevel Model of Taxation and Its Application to Optimal Highway Pricing. *Management Science*, 44(12):1608–1622, 1998.

T. Language and T. Computing. MATLAB The Language of Technical Computing. *Components*, 3(7):750, 2004.

K.-H. Lee, K.-S. Leung, C.-W. Ho, and Y.-Y. Wong. A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 7(8):506–515, 2003.

F. Legillon, A. Liefooghe, and E. G. Talbi. CoBRA: A cooperative coevolutionary algorithm for bi-level optimization. In *IEEE Congress on Evolutionary Computation*, 2012.

H. Li and Y. Wang. A Hybrid Genetic Algorithm for Solving Nonlinear Bilevel Programming Problems Based on the Simplex Method. In *3rd International Conference on Natural Computation*, volume 4, pages 91–95, 2007.

T. Li, C. Wei, W. Pei, D. E. Goldberg, and J. Richardson. PSO with sharing for multimodal function optimization. *Proceedings of the International Conference on Neural Networks and Signal Processing*, 1:450–453, 2003.

X. Li, P. Tian, and X. Min. *A Hierarchical Particle Swarm Optimization for Solving Bilevel Programming Problems*, pages 1169–1178. Springer Berlin Heidelberg, 2006.

G. E. Liepins and M. D. Voset. Representational issues in genetic optimization. *Journal of Experimental & Theoretical Artificial Intelligence*, 2(2):101–115, 1990.

R. G. Lipsey and K. Lancaster. The General Theory of Second Best. *The Review of Economic Studies*, 24(1):11–32, 1956.

F. G. Lobo. Idealized Dynamic Population Sizing for Uniformly Scaled Problems. *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference*, pages 917–924, 2011.

F. G. Lobo and D. E. Goldberg. The parameter-less genetic algorithm in practice. *Information Sciences*, 167(1):217–232, 2004.

M. López-Ibáñez, L. Paquete, and T. Stützle. Exploratory analysis of stochastic local search algorithms in biobjective optimization. Technical Report TR/IRIDIA/2009-015, 2009.

M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace Package : Iterated Race for Automatic Algorithm Configuration. Technical report, Université Libre de Bruxelles, 2011.

S. W. Mahfoud. Crowding and preselection revisited. *Parallel Problem Solving From Nature II*, pages 27–36, 1992.

S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, 1995.

P. Marcotte and M. Patriksson. Traffic equilibrium. *Handbooks in Operations Research and Management Science*, 14:623–713, 2007.

P. Marcotte, A. Mercier, G. Savard, and V. Verter. Toll Policies for Mitigating Hazardous Materials Transport Risk. *Transportation Science*, 43(2):228–243, 2009.

R. Mathieu, L. Pittard, and G. Anandalingam. Genetic algorithm based approach to bi-level linear programming. *RAIRO - Operations Research - Recherche Opérationnelle*, 28(1):1–21, 1994.

J. Maturana and F. Saubion. A compass to guide genetic algorithms. In *Lecture Notes in Computer Science*, volume 5199, pages 256–265, 2008.

M. L. Mauldin. Maintaining Diversity in Genetic Search. In *Proceedings of the Fourth AAAI Conference on Artificial Intelligence*, pages 247–250, 1984.

C. Mészáros and U. H. Suhl. Advanced preprocessing techniques for linear and quadratic programming. *OR Spectrum*, 25(4):575–595, 2003.

E. Mezura-Montes and C. A. C. Coello. Constrained optimization via multiobjective evolutionary algorithms. In *Multiobjective problem solving from nature*, pages 53–75. Springer, 2008.

Z. Michalewicz. Genetic Algorithms, Numerical Optimization, and Constraints. In *Proceedings of the 6th international conference on genetic algorithms*, volume 195, pages 151–158, 1995.

Z. Michalewicz and N. Attia. Evolutionary optimization of constrained problems. *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 98–108, 1994.

Z. Michalewicz and D. B. Fogel. *How to solve it: modern heuristics*. Springer, 2000.

Z. Michalewicz and C. Z. Janikow. GENOCOP: a genetic algorithm for numerical optimization problems with linear constraints. *Communications of the ACM*, 39(12):175, 1996.

Z. Michalewicz and G. Nazhiyath. Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *IEEE International Conference on Evolutionary Computation*, volume 2, pages 647–651, 1995.

Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(10):1–32, 1996.

K. Miettinen. *Nonlinear Multiobjective Optimization*. Springer US, 1998.

A. Migdalas. Bilevel programming in traffic planning: Models, methods and challenge. *Journal of Global Optimization*, 7(4):381–405, 1995.

B. Miller and M. Shaw. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In *IEEE International Conference on Evolutionary Computation*, pages 786–791, 1996.

M. Mitchell, S. Forrest, and J. H. Holland. The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance. In *Proceedings of the First European Conference on Artificial Life*, pages 245–254, 1991.

M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317(C):224–245, 2015.

J. F. Nash. Equilibrium points in n-Person games. *Proceedings of the National Academy of Sciences of the USA*, 36(1):48–49, 1950.

B. Naudts, D. Suys, and A. Verschoren. Epistasis as a Basic Concept in Formal Landscape Analysis. In *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 65–72, 1997.

A. E. Nix and M. D. Vose. Modeling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5(1):79–88, 1992.

Object Management Group. Business Process Model and Notation (BPMN) Version 2.0. *Business*, 50:170, 2011.

V. Oduguwa and R. Roy. Bi-level optimisation using genetic algorithm. In *Proceedings - 2002 IEEE International Conference on Artificial Intelligence Systems*, pages 322–327, 2002.

F. Oppacher and M. Wineberg. The Shifting Balance Genetic Algorithm: Improving the GA in a Dynamic Environment. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, pages 504–510. Morgan Kaufmann Publishers Inc., 1999.

D. Orvosh and L. Davis. Using a genetic algorithm to optimize problems with feasibility constraints. In *Proceedings of 1st IEEE World Congress on Computational Intelligence*, pages 548–553, 1994.

A. Osyczka and S. Kundu. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural optimization*, 10(2):94–99, 1995.

C. C. Palmer and A. Kershenbaum. An approach to a problem in network design using genetic algorithms. *Networks*, 26(3):151–163, 1995.

M. Pelikan, K. Sastry, and D. E. Goldberg. Scalability of the Bayesian optimization algorithm. *International Journal of Approximate Reasoning*, 31(3):221–258, 2002.

Performance Review Unit with ACE Working Group. ATM Cost-Effectiveness (ACE) Benchmarking Report with 2014-2018 outlook. Technical report, Tecnical Report for Performance Review Commission, 2014.

B. Peter. Railway infrastructure: pricing and investment, Internal Report. Technical report, Technical University of Berlin, 2003.

A. Pétrowski. A New Selection Operator Dedicated to Speciation. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, 1995.

A. Piszcz and T. Soule. Genetic Programming : Optimal Population Sizes for Varying Complexity Problems. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 953–954, 2006.

C. Poloni and V. Pediroda. GA coupled with computationally expensive simulations: tools to improve efficiency. In *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 267–288. John Wiley & Sons New York, 1997.

N. J. Radcliffe. Non-Linear Genetic Representations. In *Parallel Problem Solving From Nature II*, pages 259–268, 1992.

N. J. Radcliffe and P. D. Surry. For mae and the variance of fitness. *Foundations of the Genetic Algorithms*, 3:51–72, 1995.

M. Raffarin. Congestion in European airspace: A pricing solution? *Journal of Transport Economics and Policy*, 38(1):109–126, 2004.

A. Ravindran, K. M. Ragsdell, and G. V. Reklaitis. *Engineering Optimization: Methods and applications*. Wiley Online Library, 2006.

G. J. E. Rawlins. *Foundations of Genetic Algorithms*, volume 21. Elsevier, 1991.

I. Rechenberg. Cybernetic solution path of an experimental problem. *Journal of Theoretical Biology*, 215:441–448, 1965.

C. R. Reeves and C. C. Wright. Epistasis in Genetic Algorithms: An Experimental Design Perspective. In *Proc. of the 6th International Conference on Genetic Algorithms*, pages 217–224, 1995.

D. Rigonat. *Models and algorithms for an efficient market-driven management of European airspace demand*. PhD thesis, University of Trieste, 2016.

S. Roch, P. Marcotte, and G. Savard. Design and analysis of an approximation algorithm for Stackelberg network pricing. *Networks*, 46(1):57–67, 2005.

S. Ronald. Robust encodings in genetic algorithms: a survey of encoding issues. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, pages 43–48, 1997.

F. Rothlauf. *Representations for genetic and evolutionary algorithms*. Springer, 2006.

S. Ruuska and K. Miettinen. Constructing evolutionary algorithms for bilevel multiobjective optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–7, 2012.

Ryanair. Financial highlights - Annual report 2015. Technical report, Ryanair Ltd, 2015.

A. Santiago, H. J. F. Huacuja, B. Dorronsoro, J. E. Pecero, C. G. Santillan, J. J. G. Barbosa, and J. C. S. Monterrubio. A survey of decomposition methods for multi-objective optimization. In *Studies in Computational Intelligence*, volume 547, pages 453–465. 2014.

B. Sareni and L. Krähenbühl. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation*, 2(3):97–106, 1998.

N. N. Schraudolph and R. K. Belew. Dynamic Parameter Encoding for Genetic Algorithms. *Machine Learning*, 9(1):9–21, 1992.

X. Shi and H. S. Xia. Model and Interactive Algorithm of Bi-Level Multi-Objective Decision-Making with Multiple Interconnected Decision Makers. *Journal of Multi-Criteria Decision Analysis*, 10(1):27–34, 2001.

H. Shimodaira. DCGA: A diversity control oriented genetic algorithm. *Proceedings of the International Conference on Tools with Artificial Intelligence*, pages 367–374, 1997.

R. Shioda, L. Tunçel, and T. Myklebust. Maximum utility product pricing models and algorithms based on reservation price. *Computational Optimization and Applications*, 48 (2):157–198, 2011.

O. M. Shir, M. Emmerich, and T. Bäck. Adaptive niche radii and niche shapes approaches for niching with the CMA-ES. *Evolutionary computation*, 18(1):97–126, 2010.

A. Sinha, P. Malo, A. Frantsev, and K. Deb. Finding optimal strategies in a multi-period multi-leader-follower Stackelberg game using an evolutionary algorithm. *Computers and Operations Research*, 41(1):374–385, 2014.

M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer, 2008.

R. E. Smith, S. Forrest, and A. S. Perelson. Searching for Diverse, Cooperative Populations with Genetic Algorithms. *Evolutionary Computation*, 1(2):127–149, 1993.

I. M. Sobol. On the Systematic Search in a Hypercube. *SIAM Journal on Numerical Analysis*, 16(5):790–793, 1979.

H. V. Stackelberg. *The Theory of Market Economy*. Oxford University Press, 1952.

Steer Davies Gleave. Policy options for the modulation of charges in the Single European Sky. Technical report, Final report prepared for European Commission, 2015.

C. R. Stephens and H. Waelbroeck. Analysis of the Effective Degrees of Freedom in Genetic Algorithms. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, 57(3):31, 1996.

M. W. Strickberger. *Genetics*. Macmillan, 1968.

T. Takahama and S. Sakai. Constrained Optimization by ϵ Constrained Particle Swarm Optimizer with ϵ-level Control. In *Soft Computing as Transdisciplinary Science and Technology*, volume 29, pages 1019–1029. Springer, 2005.

E.-G. Talbi. *Metaheuristics for Bi-level Optimization*. Springer-Verlag Berlin Heidelberg, 1 edition, 2013.

D. Thierens. Adaptive mutation rate control schemes in genetic algorithms. In *Proceedings of the Congress on Evolutionary Computation*, pages 2–7, 2002.

M. Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time*. Springer Science, 2005.

A. Turco and C. Kavka. MFGA: a GA for Complex Real-World Optimization Problems. *International Journal of Innovative Computing and Applications*, 3(1):31–41, 2011.

R. K. Ursem. Diversity-guided evolutionary algorithms. In *Proceedings of the Parallel Problem Solving from Nature VII*, pages 462–471, 2002.

A. Van Ackere. The principal/agent paradigm: Its relevance to various functional fields. *European Journal of Operational Research*, 70(1):83–103, 1993.

S. Van Hoesel. An overview of Stackelberg pricing in networks. *European Journal of Operational Research*, 189(3):1393–1402, 2008.

D. A. Van Veldhuizen. Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Technical report, 1999.

D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis, 1998.

H. Vanhove and A. Verschoren. On epistasis. *Computers and Artificial Intelligence*, 14(3): 271–277, 1995.

L. N. Vicente and P. H. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global optimization*, 5(3):291–306, 1994.

A. Violin. *Mathematical programming approaches to pricing problems*. PhD thesis, Université Libre de Bruxelles and University of Trieste, 2014.

M. D. Vose and G. E. Liepins. Schema disruption. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 237–242, 1991.

G. P. Wagner. Adaptation and the modular design of organisms. In *Advances in artificial life*, pages 315–328. Springer, 1995.

G. Wang, Z. Wan, X. Wang, and Y. Lv. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Computers & Mathematics with Applications*, 56(10):2550–2555, 2008.

Y. Wang, Y.-C. Jiao, and H. Li. An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(2):221–232, 2005.

Y. Wang, H. Li, and C. Dang. A New Evolutionary Algorithm for a Class of Nonlinear Bilevel Programming Problems and Its Global Convergence. *INFORMS Journal on Computing*, 23 (4):618–629, 2010.

T. Weise, S. Niemczyk, H. Skubch, R. Reichle, and K. Geihs. A tunable model for multi-objective, epistatic, rugged, and neutral fitness landscapes. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 795–802, 2008.

U.-P. Wen and S.-T. Hsu. A note on a linear bilevel programming algorithm based on bicriteria programming. *Computers & Operations Research*, 16(1):79–83, 1989.

D. Whitley. The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 116–121, 1989.

D. Whitley, V. Gordon, K. Mathias, Y. Davidor, H.-P. Schwefel, and R. Männer. Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving From Nature III*, volume 866, pages 5–15. 1994.

L. D. Whitley. Fundamental Principles of Deception in Genetic Search. *Foundations of genetic algorithms*, pages 221–241, 1991.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

L. A. Wolsey. *Integer programming*, volume 42. Wiley New York, 1998.

X. Yin and N. Germay. *A Fast Genetic Algorithm with Sharing Scheme Using Cluster Analysis Methods in Multimodal Function Optimization*, pages 450–457. Springer Vienna, 1993.

Y. Yin. Genetic-Algorithms-Based Approach for Bilevel Programming Models. *Journal of Transportation Engineering*, 126(2):115–120, 2000.

A. Zinflou, C. Gagné, and M. Gravel. Genetic algorithm with hybrid integer linear programming crossover operators for the car-sequencing problem. *INFOR: Information Systems and Operational Research*, 48(1):23–37, 2010.

E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.