Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco

Electrical Engineering Department
Computer Science Section

Alternative Techniques to Handle Constraints in Evolutionary Optimization

By:
Efrén Mezura Montes

in partial fulfillment of the
requirements for the degree of:
Doctor of Science

Specialization in:
Electrical Engineering

Option:
Computer Science

Adviser:
Dr. Carlos A. Coello Coello

México City, México. December 7th, 2004.

Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Ingeniería Eléctrica
Sección de Computación

Técnicas Alternativas para el Manejo de Restricciones en Optimización Evolutiva

Tesis que presenta:
Efrén Mezura Montes

Como requerimiento parcial
para obtener el grado de:
Doctor en Ciencias

En la especialidad de:
Ingeniería Eléctrica

Opción:
Computación

Director de tesis:
Dr. Carlos A. Coello Coello

México D.F., México. Diciembre 7 de 2004.

To my beloved wife **Margarita**.

To my parents **Efraín** and **Guadalupe**.

To my grandma **Anselma**.

To my brother **José Miguel** and my sister **Irazema**.

Acknowledgments

I like to express my gratitude to my adviser Dr. Carlos A. Coello Coello for his time and encouragement dedicated to this dissertation. Thank you so much.

I also want to thank the reviewers of this document for their comments and suggestions which helped to improve its quality. Thank you to Dr. Gerardo de la Fraga, Dr. Andrés Gómez de Silva Garza, Dr. Arturo Hernández Aguirre, Dr. Zbigniew Michalewicz and Dr. Francisco Rodríguez Henríquez.

I want to thank Dr. Manuel Martinez Morales for his support on the statistics part.

I want to thank my dear friends Gregorio, Nareli, Ricardo, Erika, Daniel, Eduardo, Mario and Adriana for sharing this time of my life.

Finally I want to thank Sofia, Flor Anabel and Felipa for their support during my studies at CINVESTAV-IPN.

This research work was derived from NSF-CONACYT's project "Artificial Immune Systems for Multiobjective Optimization" (Ref. 42435-Y) whose Principal Investigator is Dr. Carlos A. Coello Coello.

Abstract

In this dissertation, a study about constraint handling in evolutionary algorithms is presented. This research has led to five main contributions: (1) A study about exploring the capabilities of using multiobjective concepts to handle constraints in global optimization using four representative approaches. The aim of this study was to make unnecessary the use of a penalty function to handle constraints in an evolutionary algorithm. Based on the results obtained, (2) we proposed a novel and competitive approach to solve constrained problems using an evolutionary algorithm (an evolution strategy in our case) which does not require the use of a penalty function. Instead, it is based on handling the objective function and the constraints of the problems separately. The approach also uses a simple diversity mechanism based on allowing infeasible solutions close to the feasible region and with a good value of the objective function to remain in the population. A simple feasibility-based comparison mechanism is adopted to guide the process towards the feasible region of the search space. Also, the initial stepsize of the evolution strategy is reduced in order to perform a finer search and a combined (discrete/intermediate) panmictic recombination technique improves its exploitation capabilities. We performed experiments in order to know which of those mechanisms (diversity mechanism, fine mutation movements and a combined recombination) was the main responsible for the good performance of the algorithm. This approach was statistically compared against state-of-the-art algorithms using well-known benchmark problems. Furthermore, (3) we performed some statistical analysis that allows to understand better the behavior of the proposed approach. Such analysis was based on the use of three performance measures related to how fast the feasible region is reached, to evaluate how is the progress once inside the feasible region and to how useful is a diversity mechanism to maintain or generate good infeasible solutions. We also performed a study to analyze the sensitivity of our approach to its parameters by means of an analysis of variance and we suggested values derived from such an analysis. In addition, (4) we performed an empirical study which aims to identify the main features that make a constrained problem difficult to solve by our proposed evolutionary algorithm. Finally, based on previous theoretical studies found in the literature, (5) we mathematically proved the global convergence of our proposed algorithm.

Resumen

Este trabajo doctoral consiste en un estudio sobre el manejo de restricciones en algoritmos evolutivos. La investigación versa sobre cinco contribuciones principales: (1) Un estudio empírico para analizar las bondades de utilizar conceptos de optimización multiobjetivo para manejar las restricciones en un problema de optimización gobal; para ello se compararon cuatro técnicas representativas del estado del arte. El objetivo principal de este estudio fue el evitar el uso de las funciones de penalización para resolver problemas con restricciones. Con base en los resultados obtenidos, (2) se propuso una técnica novedosa y competitiva para resolver problemas restringidos utilizando un algoritmo evolutivo (una estrategia evolutiva en este caso), la cual no requiere del uso de una función de penalización, sino que maneja la función objetivo y las restricciones del problema de manera separada. El algoritmo utiliza un mecanismo simple de diversidad que le permite a soluciones no factibles cercanas a la zona factible y con un buen valor de la función objetivo el permanecer en la población de la siguiente generación. Además, usa un mecanismo de comparación de individuos basado en factibilidad para guiar al algoritmo hacia la zona factible del espacio de búsqueda. La longitud de paso inicial del operador de mutación es reducido con el objeto de permitir movimientos más finos en el espacio de búsqueda. Así también, el operador de recombinación se diseñó combinando dos sencillos operadores encontrados en la literatura (recombinación discreta/intermedia) con el objeto de mejorar la capacidad explotatoria del mismo. Se realizaron experimentos para averiguar cuál de estos mecanismos (diversidad, movimientos finos de mutación y recombinación combinada) era el principal responsable del buen desempeño de la técnica, la cual fue comparada con algoritmos del estado del arte, usando un conjunto de problemas con restricciones usualmente utilizado en la literatura. Por otro lado, (3) se llevaron a cabo estudios estadísticos que permitieron entender con mayor detalle el comportamiento del algoritmo. Este análisis se basó en el uso de tres medidas de desempeño relacionadas con la velocidad de llegada a la zona factible, el evaluar el progreso de la búsqueda dentro de ella y analizar el desempeño del mecanismo de diversidad al mantener o generar nuevas soluciones no factibles. Aunado a ésto (4) se realizó un análisis de varianza para conocer la sensibilidad de la técnica a sus parámetros y poder sugerir valores para ellos. Después, se realizó otro estudio empírico, éste para saber las características de un problema que lo hacen difícil de resolver usando la técnica propuesta en este trabajo. Finalmente, con base en resultados teóricos encontrados en la literatura, (5) se realizó la demostración formal de convergencia global del algoritmo

propuesto.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Evolutionary Computation (EC) encompasses a set of algorithms called "evolutionary algorithms", that emulate evolutionary processes based on the "survival of the fittest" principle of natural selection, which is applied to a population of solutions. EC is usually adopted in problems where classical optimization techniques can not be applied, like in real-world problems of great complexity and high dimensionality in which probably neither the objective function nor the constraints are differentiable.

Evolutionary Algorithms (EAs) have been widely used to solve optimization problems [65, 4, 39, 52]. However, their good performance relies mainly on two factors: (1) their stochastic nature and (2) how good is the transformation of the objective function of the problem into a fitness function. This fitness function will lead the search to the desired region of the search space. This transformation problem becomes harder to solve in the presence of constraints. This is because the fitness function must give information not only about the goodness of a solution, but also about its closeness to the feasible region of the search space. The problem becomes more complicated in the presence of a considerable number of linear and nonlinear inequality and equality constraints. It is important to keep in mind that EAs are unconstrained search techniques which lack an explicit mechanism to deal with constrained search spaces. This has motivated the development of a considerable number of approaches to allow the incorporation of constraints [133, 28]. Usually, to incorporate a constraint handling technique to an EA involves adding extra parameters to the algorithm. These new parameters are commonly defined by the user, which increases the effort required to fine-tune the algorithm to be able to provide a reasonably good performance.

The most common approach adopted with EAs to deal with constrained search spaces is the use of penalty functions [158]. When using a penalty function, the amount of constraint violation is used to punish or "penalize" an infeasible solution so that feasible solutions are

favored by the selection process. Despite the popularity of penalty functions, they have several drawbacks from which the main one is that they require a careful fine tuning of the penalty factors that accurately estimates the degree of penalization to be applied so that we can approach efficiently the feasible region [173, 28].

In this dissertation, we study and propose a constraint handling technique for EAs which is not based on penalty functions and whose parameters do not require to be fine-tuned (unlike the penalty factors traditionally used with EAs) by the user.

Contents Road-Map

In order to provide the reader with a general overview of the contents of this dissertation we briefly describe the contents of each of its Chapters:

- **Chapter 1. Introduction**: This chapter.
- **Chapter 2. Evolutionary Computation**: A brief introduction to evolutionary computation is provided. The main related concepts and paradigms are described.
- **Chapter 3. Some Notions of Mathematical Programming**: Basic concepts of global constrained optimization as well as theoretical work regarding how to recognize global optimum solutions is detailed. Some mathematical programming approaches are briefly presented.
- **Chapter 4. Constraint Handling in Evolutionary Algorithms**: Constraint handling techniques incorporated into the fitness function of an evolutionary algorithm are described. State-of-the-art approaches are explained and discussed.
- **Chapter 5. Use of Multiobjective Optimization Concepts to Handle Constraints**: Approaches which handle the objective function and constraints separately and also use Pareto Optimality or population-based concepts are described and empirically compared. The aim is to explore different approaches to handle the objective function and the constraints of a problem separately. Some conclusions which guide the remainder of this work are provided.
- **Chapter 6. A Simple Evolution Strategy to Solve Constrained Problems**: We present a novel approach to solve constrained problems, which uses an evolution strategy as a search engine coupled with a diversity mechanism that maintains infeasible solutions close to the boundaries of the feasible region, a reduction of the initial

stepsize of the mutation operator and a combined recombination operator. The approach handles the objective function and the constraints of the problem separately. Thus, it does not require the use of a penalty function. Statistical results and a comparison against state-of-the-art approaches are presented and discussed. Moreover, the mechanisms of the approach are tested separately in order to determine which one (or what combination of them) is mandatory. We also show that the use of an evolution strategy is more adequate than using a genetic algorithm when solving constrained optimization problems.

- **Chapter 7. Performance Measures**: Statistical tests are performed to analyze how fast our approach reaches the feasible region or even the global optimum solution. Besides, three performance measures are used and statistically tested in order to analyze the improvements achieved once the feasible region is found and how well the diversity mechanism works once most of the solutions in the population are feasible.

- **Chapter 8 What Makes a Constrained Optimization Problem Difficult to Solve by The Proposed Approach**: We empirically show what features of a problem decrease the performance of an evolutionary algorithm that provides good results when tested on the well-known benchmark traditionally adopted in the specialized literature.

- **Chapter 9 Global Convergence Properties of The Proposed Approach**: We use theoretical work found in the literature to prove that our algorithm (which is based on an evolution strategy) converges to the global optimum of a constrained problem assuming infinite time.

- **Chapter 10 Final Remarks**: We provide a summary of our findings, some conclusions based on the results obtained and we present the future work.

- **Appendix A**: The complete descriptions of the test functions used are presented here.

- **Appendix B**: The set of graphics obtained by the statistical test performed using our approach are shown here.

- **Appendix C**: The set of graphics obtained by the statistical test performed using the three performance measures in our approach are shown here.

Chapter 2

Evolutionary Computation

EC is based on the concept of *Neo-Darwinism* [59] which arises from the coupling of Darwin's Natural Selection [41], Weismann's "Germoplasm theory" [91, 10] and Mendel's Laws of Inheritance [10]. Thus, *Neo-Darwinism* can be seen as the biological foundation of EC (see Figure 2.1). Based on this idea, all life on Earth can be explained from the following processes [58]:

1. Reproduction: A way to create new individuals from a current set of them.

2. Mutation: Small changes in the genetic codification of individuals which cause significant changes in their features.

3. Competition: A mechanism to measure how fit is an individual to survive in an specific environment.

4. Selection: Leads to the maintainance or increase of populations fitness, where fitness is defined as the ability to survive and reproduce in an specific environment.

2.1 Description of a Generic Evolutionary Algorithm

The combination of Genetics (representation and relation among individuals), and the natural selection process (fitness, surviving strategies) form the general model of an Evolutionary Algorithm (EA).

Five main elements are required to model an evolutionary process in a computer [128]:

1. A suitable representation for the solutions of the problems to be solved (individuals).

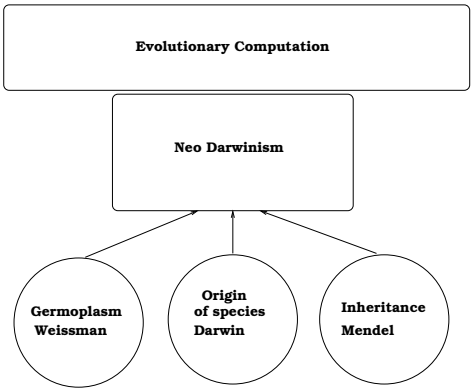


Figure 2.1: Neo-Darwinism as the biological foundation of EC

2. A mechanism to generate an initial population of individuals.

3. A fitness function that plays the role of the environment and is on charge of evaluating an individual's performance. From a practical point of view, it is impossible to include all possible factors of an environment in the definition of a fitness function. Regarding optimization problems, the fitness function represents an abstraction of the environment, including just those variables involved in the optimization task.

4. Reproduction operators used to generate more individuals, (namely crossover and mutation).

5. Values for the parameters of the algorithm (population size, crossover and mutation rates, etc.)

In a general way, to solve a problem using an EA the following steps must be performed [128]:

1. Generate a *random initial population* of $n > 0$ individuals which represent potential solutions to the problem.

2. *Select* the fittest individuals based on their *fitness function* value.

3. Apply reproduction operators (crossover, mutation, etc.) to generate new individuals.

4. Loop until a stop condition is satisfied.

2.2 EC Concepts

2.2.1 Fitness function

The *fitness* of an individual relates to the evaluation of the objective function and it is a measure of goodness of an individual (solution to the problem) with respect to the other individuals in the population. The solutions to a problem are $n-$dimensional parameter vectors $\vec{x} \in \mathbb{R}^n$, and the objective function can be defined as:

$$f : \mathbb{R}^n \rightarrow \mathbb{R} \tag{2.1}$$

In this way, the fitness function is in principle identical to f, i.e. given an individual $\vec{a} \in \mathbb{R}^n$, we have

$$\text{fitness}(\vec{a}) = f(\vec{a}) \tag{2.2}$$

This similarity can change, for example, with the use of penalty functions for constrained optimization. This sort of approach will be discussed detail in Chapters 3 and 4.

2.2.2 Representation

From its biological foundations EC takes some terms: A *chromosome* is a data structure representing an individual in the population. Usually it is an integer array, where each position is known as a *gene*. Each gene normally codifies one value of the decision parameters of the problem. Each location within a gene can get a value, which is known as an *allele*.

There are two levels of representation used in EC: (1) genotypic and (2) phenotypic. The *genotype* is the encoding represented by the chromosome and genes. The *phenotype* is the result of decoding the values of the chromosome into the values of the variables of the problem to be solved. Each iteration of the process is known as a *generation*.

The set of individuals known as a *population* can be divided into *subpopulations*. In this way, *speciation* can be modeled, because normally only individuals of the same subpopulation can be recombined. If some individuals can be transferred to other subpopulations, the *migration* operator is modeled.

The most used representations in EC are:

- Binary [85].
- Binary with gray codes [182].
- Integer [29, 31].

- Real [58, 170].
- Trees [108].
- Variable-length binary lists [68].
- Hybrid [42].

2.2.3 Reproduction operators

Also known as genetic operators, their function is to modify the way in which genetic information is transmitted from parents to offspring. There are three main categories:

1. *Crossover*: It uses parts of the parent chromosomes to generate a new individual. In numerical optimization the most used are:
   - One point [86].
   - Two points [97].
   - Uniform [1, 177].

   There are some other variants depending of the representation [6].
2. *Mutation*: A new individual is formed by small modifications to the genetic information of just one parent [128].
3. *Reordering*: Alters the order of genes of one individual [66].

There are two subprocesses related to the genetic operators [52]: (1) *exploitation*: based on fine movements to sample promising zones of the search space of the problem. Then, either local or even global optima can be found in these zones. Crossover is responsible for creating several individuals in a promising zone to sample it wide enough as to find the global optimum. (2) *exploration*: It refers to the search of these promising zones and avoiding to get trapped in local optima. Mutation is responsible for performing large jumps in the search space to explore it.

2.2.4 Selection

The selection of individuals is the mechanism to guide the search towards good solutions (either a local or global optimum). A taxonomy for selection techniques is as follows:

- *Proportional selection* [67]: Individuals are selected based on their fitness contribution to the total amount of fitness of the population. Some proposals are:
  - Roulette wheel [97].
  - Stochastic remainder:
    - With replacement [20].
    - Without replacement [22].
  - Universal stochastic [9].
  - Deterministic sampling [97].
- *Tournament selection* [180]: Direct comparisons of fitness among individuals. It can be either binary (two individuals) or with more than two individuals. The fittest one wins. There are two main types:
  - Deterministic: The fittest always wins.
  - Probabilistic: The fittest wins with a certain probability (i.e. in some cases, the fittest solution will not be selected).
- *Steady state selection* [181]: It is used on nongenerational genetic algorithms. Useful when individuals solve the problem in a collective way.

Tournament selection has a higher selection pressure than proportional selection. Selection pressure refers to the probability for less fit solutions to survive. When the selection pressure is high, less fit solutions have little or zero probability to survive. On the other hand, when the selection pressure is moderate, they have chances to be selected for reproduction or to be part of the population for the next generation. Among proportional selection techniques, there are some of them where less fit solutions have more probabilities to survive. For example, universal stochastic sampling assigns a higher probability of survival to less fit solutions than, for example, roulette wheel.

It is very important to mention that one problem when using EAs to solve optimization problems is to maintain an adequate diversity in the population. For the case of unconstrained optimization, diversity means to have solutions located in different zones of the search space. In this way, the search space is sampled better and more promising zones can

be explored in order to find the global optimum. For the case of constrained optimization, diversity implies to have feasible solutions as well as infeasible solutions. The aim is to sample the boundaries of the feasible region. The motivation in this case is that the most difficult constrained optimization problems normally have their global optimum located precisely on the boundary between the feasible and the infeasible regions. An additional mechanism is usually added to encourage diversity. This is because EAs tend to converge to a single point over time. Therefore, diversity mechanisms are required to maintain useful solutions (different ones in case of unconstrained optimization and infeasible solutions in case of constrained optimization) besides the best solutions found. In other words, an EA must be capable to generate new solutions in unexplored regions of the search space for the case of unconstrained optimization. On the other hand, for the case of constrained optimization, solutions near the boundaries of the feasible region must be generated.

2.2.5 Adaptation

There are different forms to control the parameter values of an EA. Based on the classification proposed by Hinterding et al. [82] and explained by Bäck [7], the existing approaches are:

- *No parameter control (Static)*: The parameters are given an initial value and they remain the same during all the evolutionary process.

- *Dynamic parameter control*: Parameter settings are modified according to a deterministic schedule prescribed by the developer of the EA (usually parameter values vary depending of the number of the current generation).

- *Adaptive parameter control (On-line adaptation)*: New values of parameters are obtained by a feedback mechanism that monitors evolution and rewards or punishes parameter settings according to whether they have caused an improvement or deterioration in the expected behavior of the EA.

- *Self-adaptive parameter control*: Parameter values are codified and evolved by the evolutionary algorithm by applying evolutionary operators to their codifications in a similar way as to the solution representations. Besides its use to evolve problem's solutions, the EA is also used to determine if the changes of the parameters are advantageous concerning their impact on the expected behavior of the algorithm.

2.3 Paradigms

From the general steps to solve a problem using an EA mentioned at the beginning of this chapter, three different paradigms are known. In their original versions, they differ at the level at which they simulate evolution, on their main and secondary genetic operators and on the type of representation used. Note however, that in the current literature is more common to refer to "evolutionary algorithms" in a general sense, since their specific features tend to be indistinguishable.

2.3.1 Evolutionary programming (EP)

Proposed by Fogel [61] in 1960 where he remarks the inheritance relationships and behavior between parents and offspring. Adaptation is conceived in this paradigm as a type of intelligence.

The EP model simulates evolution at the species level, therefore, there is no crossover operator (in nature it is not possible to mate animals from different species, e.g. a lion with a mouse). The selection technique is based on stochastic tournaments. The main and only operator is mutation. Besides, EP operates at a phenotype level (i.e. it requires no encoding). In the original proposal of EP, self-adaptation of the mutation parameters were not considered.

The basic EP algorithm is presented in Figure 2.2:

```
Begin
    Generate randomly an initial population of solutions.
    Calculate the fitness of the initial population.
    Repeat
        Apply mutation to all the population to create offspring.
        Evaluate each offspring.
        Select (using stochastic tournaments) the individuals
        of the next generation.
    Until a stop condition is satisfied.
End
```

Figure 2.2: Pseudocode of EP algorithm.

Each individual participates in a predefined number of binary tournaments and the individuals with more wins will have a higher probability of being part of the population in the next generation.

The applications of EP cover (among others) the following generic tasks [59]:

- Forecasting.
- Games.
- Automatic control.
- Traveling salesperson problem.
- Routing.
- Design and training of neural networks.
- Pattern recognition.

There is some theoretical research regarding rates of convergence for specific instances of EP [60].

2.3.2 Evolution strategy (ES)

The ES was developed in Germany in 1964 to solve complex hydrodynamical problems. The researchers involved in this work were Ingo Rechenberg, Hans-Paul Schwefel and Paul Bienert [170].

The ES simulates the evolution at an individual level, then, there is a crossover operator, either sexual or panmictic (more than two parents), which, however, acts as a secondary operator. As in EP, mutation is the main operator and it is used with random numbers generated under a Gaussian distribution. The mutation values vary over time and are self-adaptive. ES representation is at a phenotypic level (i.e. no encoding is required) and unlike EP, the selection process is deterministic and extinctive (the worst individuals have zero probabilities of survival).

There are several versions of ES's. The first of them is the $(1+1)$-ES. This version has only one solution which is mutated to create one child. If the child is better than the parent, it will replace it. The first number between parentheses refers to the size of the parents population (one in this case), the "$+$" sign refers to the type of selection (the other possible selection is the "$,$" selection) and the last value refers to the number of offspring created from the parents (also one in this case). There are other types of ES's like the $(\mu+1)$-ES, $(1+\lambda)$-ES, $(\mu+\lambda)$-ES and the $(\mu+\lambda)$-ES. All of them will be explained in detail in Chapter 6 because the ES is the paradigm used in this dissertation. ES's have been applied to solve problems like [169]:

```
Begin
    Generate randomly an initial population of solutions.
    Calculate the fitness of the initial population.
    Repeat
        Apply crossover to create offspring.
        Apply mutation operator to all offspring.
        Evaluate each offspring.
        Select the best individuals for the
        next generation.
    Until a stop condition is satisfied.
End
```

Figure 2.3: Pseudocode of ES algorithm.

- Routing of networks.
- Bio-Chemistry.
- Optics.
- Engineering design.
- Magnetism.

There is theoretical work regarding rates of convergence for the $(1+1)$-ES, the $(1+\lambda)$-ES and the $(\mu+\lambda)$-ES [6].

2.3.3 Genetic algorithm (GA)

GA's were originally called "Reproductive Plans" and were conceived by John H. Holland [86] in the 1960's. The main motivation for this approach was machine learning. Goldberg [66] gives a definition of a GA:

> Genetic Algorithms are search algorithms based on the mechanisms of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search.

The GA works at genotypic level and the sexual crossover is its main operator because GA's emulate evolution at an individual level. Mutation is the secondary operator. The

selection is probabilistic based on fitness. They are usually not self-adaptive. The basic GA is presented in Figure 2.4 [23]:

```
Begin
    Generate randomly an initial population of solutions.
    Calculate the fitness of the initial population.
    Repeat
        Select a pair of parents to create two offspring using crossover.
        Apply mutation to each child.
        Evaluate the mutated offspring.
        All the offspring will be in the new population.
    Until a stop condition is satisfied.
End
```

Figure 2.4: Pseudocode of GA algorithm.

Some applications of GA's are [66] :

- Optimization (numerical and combinatorial).
- Machine learning.
- Query optimization in databases.
- Pattern recognition.
- Grammar generation.
- Robot motion planning.
- Prediction.

One of the most important theoretical contribution in GA's is the Schema Theorem [85]. This theorem gives an idea of the way in which GAs work. Besides, there is a convergence proof for an elitist GA [160] .

2.3.4 Other approaches

There are other EA's which are not included in the three main paradigms but those are commonly used to solve optimization problems nowadays. They are Genetic Programming

[108], Differential Evolution [148], Swarm Intelligence (which covers Particle Swarm Optimization [101], Ant Colony System [50] and Cultural Algorithms [157]) and Artificial Immune Systems [47]. The details of these approaches are beyond the scope of this thesis.

Chapter 3

Some Notions of Mathematical Programming

The body of mathematical results and numerical methods for finding and identifying the best candidate from a collection of alternatives without having to explicitly enumerate and evaluate all possible alternatives is called Optimization [156]. In this chapter, we provide some basic concepts of global constrained optimization. Also, we discuss theoretical work regarding how to recognize if a solution is the feasible global optimum and, finally, we present some mathematical programming approaches used to solve constrained optimization problems.

3.1 Introduction

The set of techniques used to solve optimization problems is known as *mathematical programming* techniques. Moreover, there are other approaches to solve these problems like stochastic techniques and statistical methods. Stochastic methods are used to solve problems stated as a set of random variables with a known probability distribution. Statistical methods are based on analyzing experimental data to elaborate empirical models to estimate a representation of the real problem. Optimization techniques are used to solve a wide set of problems like: optimal paths for spatial vehicles, spatial and airplane design problems, mechanical component design, turbine and heat exchanger design, electrical equipment design, electric and hydraulic network design, schedule optimization, optimization of chemical processes, etc. [149, 156].

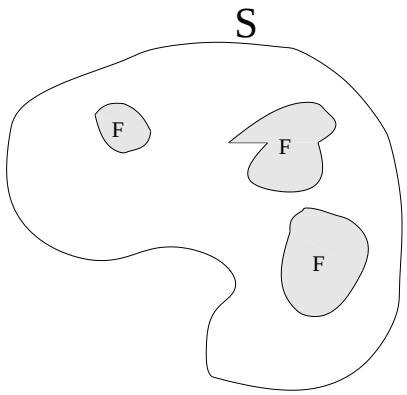


Figure 3.1: Search space S and its feasible region F

3.2 Statement of the Problem

We are interested in the general nonlinear programming (NLP) problem in which we want to:

$$\text{Find } \vec{x} \text{ which optimizes } f(\vec{x}) \tag{3.1}$$

subject to:

$$g_i(\vec{x}) \leq 0, \;\; i = 1, \ldots, m$$

$$h_j(\vec{x}) = 0, \;\; j = 1, \ldots, p$$

where $\vec{x}$ is the vector of solutions $\vec{x} = [x_1, x_2, \ldots, x_n]^T$, m is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear). If we denote with $\mathcal{F}$ to the feasible region and with $\mathcal{S}$ to the whole search space, then it should be clear that $\mathcal{F} \subseteq \mathcal{S}$ (see Figure 3.1). For an inequality constraint that satisfies $g_i(\vec{x}) = 0$, we will say that is active at $\vec{x}$; it is said to be inactive if $g_i(\vec{x}) < 0$. All equality constraints h_j (regardless of the value of $\vec{x}$ used) are considered active at all points of $\mathcal{F}$.

In the following definitions we will assume minimization (without loss of generality). $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ refers to the optimum point and its corresponding value of the objective function $f(\vec{x}^*)$ is called the optimum value. The pair $\vec{x}^*$ and $f(\vec{x}^*)$ is called optimum solution.

Definition 1 (Monotonic function): *A function $f(\vec{x})$ is monotonic (either increasing or decreasing) if, for any two points $\vec{x}_1$ and $\vec{x}_2$ with $\vec{x}_1 \leq \vec{x}_2$, it follows that $f(\vec{x}_1) \leq f(\vec{x}_2)$ (monotonically increasing) or $f(\vec{x}_1) \geq f(\vec{x}_2)$ (monotonically decreasing).* □

Definition 2 (Unimodal function): *A function $f(\vec{x})$ is unimodal on the interval $\vec{a} \leq \vec{x} \leq \vec{b}$ if and only if it is monotonic on either side of the single optimal point $\vec{x}^*$ in the interval. In other words, if $\vec{x}^*$ is the single minimum point of $f(\vec{x})$ in the range $\vec{a} \leq \vec{x} \leq \vec{b}$, then $f(\vec{x})$ is unimodal on the interval if and only if for any two points $\vec{x}_1 \leq \vec{x}_2$:*
$\vec{x}^* \leq x_1 \leq x_2$ implies that $f(\vec{x}^*) \leq f(x_1) \leq f(x_2)$ and
$\vec{x}^* \geq x_1 \geq x_2$ implies that $f(\vec{x}^*) \leq f(x_1) \leq f(x_2)$ □

Based on the type of the objective function: unimodal or multimodal (more than one local minimum in a give interval, see Figure 3.2) there are categories of optimum solutions.

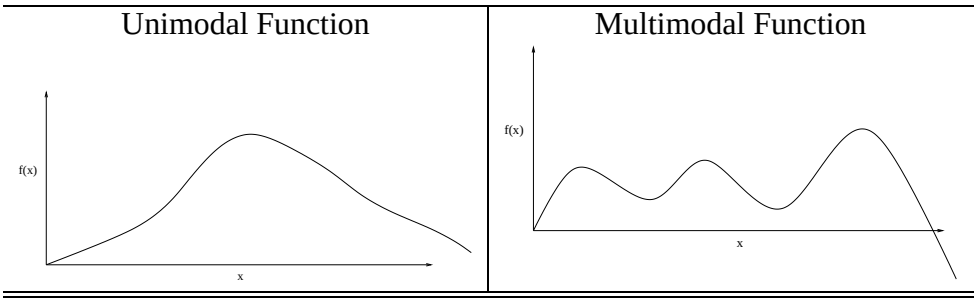


Figure 3.2: Unimodal and multimodal functions with one decision variable.

Definition 3 (Global minimum): *A function $f(\vec{x})$ defined on a set S attains its global minimum at a point $\vec{x}^* \in S$ if and only if: $f(\vec{x}^*) \leq f(\vec{x})$ for all $\vec{x} \in S$* □

Definition 4 (Local Minimum): *A function $f(\vec{x})$ defined on a set S has a local minimum (relative minimum) at a point $\vec{x}^l \in S$ if and only if: $f(\vec{x}^l) \leq f(\vec{x})$ for all $\vec{x}$ within a distance ϵ from $\vec{x}^l$. That is, there exists an $\epsilon > 0$ such that for all $\vec{x}$ satisfying $|\vec{x} - \vec{x}^l| < \epsilon$, $f(\vec{x}^l) \leq f(\vec{x})$.* □

Definition 5 (Convex function): *A function $f(\vec{x})$ is called convex over $\mathbb{R}$ if for any given two vectors $\vec{x}_1 \leq \vec{x}_2 \in \mathbb{R}$,*
$f(\theta\vec{x}_1 + (1-\theta)\vec{x}_2) \leq \theta f(\vec{x_1}) + (1-\theta)f(\vec{x}_2)$
where θ is a scalar in the range $0 \leq \theta \leq 1$. □

A function is strictly convex is for $\vec{x}_1 \neq \vec{x}_2$ the $\leq$ sign can be replaced by a $<$.

Moreover, if the reverse inequality holds, the function is concave. In this way, A function $f(\vec{x})$ is concave if $-f(\vec{x})$ is convex (see Figure 3.3).

Constraints in real world problems are known as design constraints and they can be classified into the following categories:

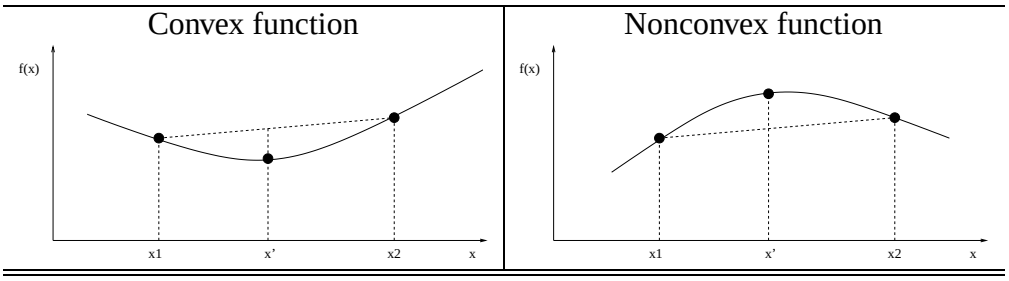


Figure 3.3: Convex and nonconvex functions.

- *Functional constraints*: They represent limitations in the performance of either a system or object.
- *Geometric constraints*: They represent physical restraints like availability and transportation matters.

In this work, we deal with the nonlinear optimization problem in its more general form. Thus, inequality and equality constraints can be linear or nonlinear. In this way, for some problems the feasible region may be convex or non-convex.

Here we define a convex set (see Figure 3.4):

Definition 6 (Convex set): *A set of points is convex set in a n-dimensional space if, for any pair of points $\vec{x}_1$ and $\vec{x}_2$ in the set, the line that joins them is also completely inside the set. In this way, every point $\vec{x}$, where:*
$\vec{x} = \theta\vec{x}_1 + (1-\theta)\vec{x}_2, 0 \leq \theta \leq 1$
is also in the set. □

3.3 Kuhn-Tucker Conditions

Harold Kuhn and Albert Tucker developed the necessary and sufficient optimality conditions for the general nonlinear programming problem, defined previously in Equation. 3.1. Here we rewrite this definition (Equation 3.2) because the Kuhn-Tucker conditions are defined when inequality constraints are satisfied for positive values.

$$\text{Find } \vec{x} \text{ which optimizes } f(\vec{x}) \tag{3.2}$$

subject to:

$$g_j(\vec{x}) \geq 0, \;\; i = 1, \ldots, J$$

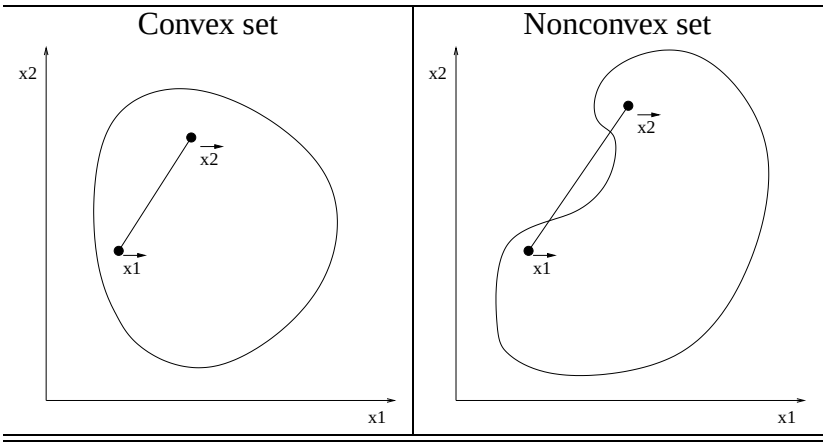


Figure 3.4: Convex and nonconvex sets.

$$h_k(\vec{x}) = 0, \;\; j = 1, \ldots, K$$
$$\vec{x} = [x_1, x_2, \ldots, x_N]$$

If inactive constraints can be identified at the optimum before solving the problem, they can be deleted from the model and reduce the problem size. The main difficulty lies in identifying the inactive constraints before the problem is solved.

Kuhn and Tucker developed the necessary and sufficient optimality conditions for the NLP problem *assuming that the functions* f, g_j, and h_k, *are differentiable*. These optimality conditions, commonly known as the *Kuhn-Tucker conditions* (KTC) consist of finding a solution to a system of nonlinear equations. Therefore, they are also known as the *Kuhn-Tucker problem*. Below we detail it:

Find vectors $\vec{x}_{(NX1)}$, $\vec{u}_{(1XJ)}$ and $\vec{v}_{(1XK)}$ that satisfy:

$$\nabla f(\vec{x}) - \sum_{j=1}^{J} u_j \nabla g_j(x) - \sum_{k=1}^{K} v_k \nabla h_k(\vec{x}) = 0 \tag{3.3}$$

$$\begin{array}{ll} g_j(\vec{x}) \geq 0, & j = 1, 2, \ldots, J; \\ h_k(\vec{x}) = 0 & k = 1, 2, \ldots, K; \\ u_j g_j(\vec{x}) = 0 & j = 1, 2, \ldots, J; \\ u_j \geq 0 & j = 1, 2, \ldots, J; \end{array} \tag{3.4}$$

Now, we state the K-T Necessity and Sufficiency Theorems:

Kuhn-Tucker Necessity Theorem [156]
Consider the NLP problem detailed in Equation 3.2. Let f, g_j and h_k be differentiable functions and $\vec{x}^*$ be a feasible solution to the NLP. Let $I = \{j|g_j(x^*) = 0\}$. Furthermore, $\nabla g_j(x^*)$ for $j \in I$ and $\nabla h_k(\vec{x}^*)$ for $k = 1, \ldots, K$ are linearly independent. If $\vec{x}^*$ is an optimal solution to the NLP problem, then there exists a $(\vec{u}^*, \vec{v}^*)$, such that $(\vec{x}^*, \vec{u}^*, \vec{v}^*)$ solves the K-T problem given in Equations 3.3 and 3.4.

The proof of the theorem can be found in [11]. *Constraint Qualification* [156] are the conditions that $\nabla g_j(x^*)$ for $j \in I$ and $\nabla h_k(\vec{x}^*)$ for $k = 1, \ldots, K$ are linearly independent at the optimum. A constraint qualification implies certain regularity conditions on the feasible region that are frequently satisfied in practical problems. However, it is, in general, quite hard to verify the constraint qualification because the global optimum must be known a-priori.

For some NLP problems the constraint qualification is satisfied when [156]: (1) all the inequality and equality constraints are linear, and (2) when all the inequality constraints are concave functions and the equality constraints are linear and there exists at least one feasible $\vec{x}$ that is strictly inside the feasible region of the inequality constraints. In other words, there exists an $\vec{x}$ such that $g_j(\vec{x}) > 0$ for $j = 1, \ldots, J$ and $h_k(\vec{x}) = 0$ for $k = 1, \ldots, K$. When a constraint qualification is not met at the optimum, there may not exist a solution for the K-T problem.

The K-T necessity theorem helps to identify points that are not optimal. It means that, given a feasible point that satisfies the constraint qualification, the K-T necessity theorem can be used to prove that this point is not optimal if it does not satisfy the K-T conditions. However, if this point does satisfy the K-T conditions (there is a solution for the K-T problem), we can not assure that this point is optimal for the NLP problem. The following theorem gives conditions under which a K-T point (a point which satisfies the K-T necessity theorem) automatically becomes an optimal solution to the NLP problem.

Kuhn-Tucker Sufficiency Theorem[156]
Given the NLP problem in Equation 3.2. Let the objective function f be convex, the inequality constraints $g_j(\vec{x})$ be all concave functions for $j = 1 \ldots, J$, and the equality constraints $h_k(\vec{x})$ for $k = 1, \ldots, K$ be linear. If there exists a solution $(\vec{x}^*, \vec{u}^*, \vec{v}^*)$ that satisfies the K-T conditions given by Equations 3.3 and 3.4, then $\vec{x}^*$ is an optimal solution to the NLP problem.

A proof of the Kuhn-Tucker Sufficiency Theorem can be found in [118]. Note that, when the sufficiency conditions of this theorem hold, finding a K-T point gives an optimal solution to an NLP problem. In this way, the Sufficiency Theorem can be used to prove

that a given solution is optimal for a NLP problem.

It is important to remark that for practical problems the constraint qualification will generally hold. If the functions and constraints of the problem are differentiable, a K-T point is a possible candidate for the optimum. Thus, many NLP methods try to converge to a K-T point. Besides, when the conditions of the Sufficiency theorem hold, a K-T point automatically becomes the global optimum. Unfortunately, the sufficiency conditions are difficult to verify (we need the Hessian matrix of each function to verify if they are positive or negative defined [149]), and often practical problems may not possess these properties. In fact, the presence of one nonlinear equality constraint is enough to violate the assumptions of the Sufficiency Theorem. Finally, the Sufficiency Theorem has been generalized further to nonconvex inequality constraints, nonconvex objective functions and nonlinear equality constraints. These use generalizations of convex functions such as quasi-convex and pseudo-convex functions. More details can be found in [118].

McCormick [119] developed second-order necessary and sufficient optimality conditions that apply to twice-differentiable functions:

Second-order Necessity Theorem[156]
Let f, g and h be twice-differentiable functions, and let $\vec{x}^*$ be feasible. Let the active constraints set at $\vec{x}^*$ be $I = \{j | g_j(\vec{x}^*) = 0\}$. Furthermore, assume that $\nabla g_j(\vec{x}^*)$ for $j \in I$ and $\nabla h_k(\vec{x}^*)$ for $k = 1, 2, \ldots, K$ are linearly independent. Then, the *necessary conditions* that $\vec{x}^*$ be a local minimum to the NLP problem are that:

1. There exists $(\vec{u}^*, \vec{v}^*)$ such that $(\vec{x}^*, \vec{u}^*, \vec{v}^*)$ is a K-T point.

2. For every vector $y_{(1\times N)}$ satisfying:
   $\nabla g_j(\vec{x}^*)y = 0$ for$j \in I$
   $\nabla h_k(\vec{x}^*)y = 0$ for $k = 1, 2, \ldots, K$

   if follows that:
   $y^T \mathbf{H}_L(\vec{x}^*, \vec{u}^*, \vec{v}^*)y \geq 0$

   where
   $L(\vec{x}, \vec{u}, \vec{v}) = f(\vec{x}) - \sum_{j=1}^{J} u_j g_j(\vec{x}) - \sum_{k=1}^{K} v_k h_k(\vec{x})$
   and $\mathbf{H}_L(\vec{x}^*, \vec{u}^*, \vec{v}^*)$ is the Hessian matrix of the second partial derivatives of L with respect to x evaluated at $(\vec{x}^*, \vec{u}^*, \vec{v}^*)$.

When a point satisfies the second-order necessary conditions given above, it becomes a K-T point and a candidate for a local minimum. To verify if it is indeed a local minimum

we need the second-order sufficient conditions. If the point does satisfy these sufficient conditions, it is the global optimum.

Second-order Sufficiency Theorem[156]
Sufficient conditions that a point $\vec{x}^*$ is a strict local minimum of a NLP problem, where f, g_j, and h_k are twice-differentiable functions are that:

1. There exists $(\vec{u}^*, \vec{v}^*)$ such that $(\vec{x}^*, \vec{u}^*, \vec{v}^*)$ is a K-T point.

2. For every nonzero vector $y_{(1\times N)}$ satisfying:
   $\nabla g_j(\vec{x}^*)y = 0 \quad j \in I_1 = \{j|g_j(\vec{x}^*) = 0, \vec{u}^* > 0\}$
   $\nabla g_j(\vec{x}^*)y \geq 0 \quad j \in I_2 = \{j|g_j(\vec{x}^*) = 0, \vec{u}^* = 0\}$
   $\nabla h_k(\vec{x}^*)y = 0 \quad k = 1, 2, \ldots, K$
   $y \neq 0$

   it follows that

   $y^T \mathbf{H}_L(\vec{x}^*, \vec{u}^*, \vec{v}^*)y > 0$

   where $I_1 \cup I2 = I$ is the set of all active constraints at $\vec{x}^*$.

As it can be noted, the Second-order Sufficiency Theorem only varies the following: The first condition of point number two of the Second-order Necessity Theorem does not need to be satisfied for all active constraints and the inequality $y^T \mathbf{H}_L(\vec{x}^*, \vec{u}^*, \vec{v}^*)y \geq 0$ in the Necessity Theorem must be satisfied as an strict inequality in the Sufficiency Theorem ($y^T \mathbf{H}_L(\vec{x}^*, \vec{u}^*, \vec{v}^*)y > 0$).

It is important to note that the second-order Necessity and Sufficiency Theorems do not require convexity of the functions nor linearity of the equality constraints. However, they add additional restrictions (twice-differentiable functions).

There are also optimality criteria for nondifferentiable functions. They are known are Saddlepoint Conditions. However, saddlepoints may not exist for all NLP problems. The existence of saddlepoints is guaranteed only for NLP problems that satisfy the constraint qualification (based on convexity and concavity of the function and constraints). Furthermore, to determine a saddlepoint is generally difficult.

After reviewing theoretical concepts about optimality conditions for constrained problems is clear to note that they are valid only when the problem to solve has some very rigid features. Therefore, constrained optimization is an open problem, Several mathematical programming techniques and also heuristic-based approaches have been proposed to solve this type of problems.

3.4 Methods

There are several mathematical-programming-based methods to solve the NLP problem. They can be classified into two categories:

1. Transformation Methods.

2. Direct Methods.

3.4.1 Transformation methods

In this category, the original constrained problem is transformed into a sequence of unconstrained problems via the penalty function. The structure of the penalty function along with the rules for updating the penalty parameters at the end of each unconstrained minimization stage define the particular method. The penalty function is exact if only one unconstrained minimization is required.

These methods assume that an initial solution $\vec{x}^{(0)}$ of the problem is available. $\vec{x}^{(0)}$ may or may not be feasible. Transformation Methods generate a sequence of points in R^N from $\vec{x}^{(0)}$ to $\vec{x}^{(T)}$, where $\vec{x}^{(t)}$ is the generic point and $\vec{x}^{(T)}$, the limit point, is the best estimated of the global optimum produced by the algorithm. The points $\vec{x}^{(t)}$, $t = 1, 2, \ldots, T$, are stationary points of an associated unconstrained function called a penalty function.

There are two types of penalty functions:

- *Exterior*: More commonly used in evolutionary algorithms (see Chapter 4 for details). In this case, the algorithm starts with infeasible solutions and the search will be guided towards the feasible region of the search space. The penalty value will be low at the beginning of the search and it will be increased over time (i.e. iterations). The idea is to allow the search to move towards the feasible region and, once it is reached, do not leave it.

- *Interior*: Also known as barrier penalties. In this case, the algorithm starts with a feasible solution (which, for some problems, is not easy or computationally efficient to get [173]) and moves inside the feasible region. In this case, the penalty factor is low in zones far from the boundaries of the feasible region and it will be high in zones close to the boundaries. This allows the search to move inside the feasible region trying to locate the global optimum.

The general formula of a penalty function is the following:

$$\phi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{n} r_i \cdot G_i + \sum_{j=1}^{p} c_j \cdot L_j\right] \tag{3.5}$$

where $\phi(\vec{x})$ is the expanded objective function to be optimized, G_i and L_j are functions of the constraints of the problem $g(\vec{x})$ and $h(\vec{x})$, respectively, and r_i y c_j are positive constants called "penalty factors" which determine the severity of the penalty.

Any useful transformation technique should have the following characteristics [156]:

1. The subproblem solutions should approach a solution of the NLP, that is:

$$\lim_{t \to T \leq \infty} \vec{x}^{(t)} = x^*$$

2. The problem of minimizing $\phi(\vec{x})$ should be similar in difficulty to minimizing the original $f(\vec{x})$. That is, the method will be less than useful if the unconstrained subproblems are excessively difficult to solve, no matter how strong the theoretical basis of convergence.

3. The definition of the penalty factors should be simple. That is, the overhead associated with updating them should be small compared to the effort associated with solving the unconstrained subproblems.

The main drawback of penalty functions is indeed the definition by the user of an adequate value for the penalty factors because reaching the feasible region as well as sampling it well enough as to reach the global optimum relies on these factors.

Penalty terms

There are different penalty forms that have been widely used and represent different procedures for handling constraints in an unconstrained setting. Here are the most used:

1. *Parabolic penalty*:

$$\Omega = R\{h(\vec{x})\}^2$$

Used for equality constraints. It equally discourages positive or negative violations of $h(\vec{x})$. Besides, with increasing values of its penalty factor R, the stationary values of $\phi(\vec{x})$ will approach the global optimum $\vec{x}^*$, since in the limit as R grows large $h(\vec{x}^T) = 0$. Finally, Ω is continuous and has continuous derivatives (see Figure 3.5-a).

2. *Infinite barrier*: It assigns an infinite penalty to all infeasible points and no penalty to feasible ones. It is discontinuous along the boundary of the feasible region and, rather obviously, $\nabla\phi(\vec{x})$ does not exist along the boundary. For its practical use, a large positive value (10^{20}) [95] can be used (see Figure 3.5-b).

3. *Log penalty*: It produces a positive penalty for all $\vec{x}$ such that $0 \leq g(\vec{x}) \leq 1$ and a negative penalty for all $\vec{x}$ such that $g(\vec{x}) \geq 1$. In this way, interior points are artificially favored over boundary points. The log penalty is an interior penalty and in fact is not defined for $\vec{x}$ such that $g(\vec{x}) \leq 0$ (see Figure 3.5-c).

4. *Inverse penalty*:
$$\Omega = R\left[\frac{1}{g(\vec{x})}\right]$$
The inverse penalty is also a barrier penalty, where feasible points near the boundaries are assigned quickly decreasing penalties as the interior region is penetrated. Infeasible solutions close to the boundaries are penalized higher than those solutions that are far from the feasible region. Therefore, special safeguards must be implemented in presence of infeasible solutions. $\phi(\vec{x})$ and $\nabla\phi(\vec{x})$ do not exist along the boundary of the feasible region. Beginning with an initial feasible point and R positive, R is decreased toward zero in the limit (see Figure 3.5-d).

5. *Bracket penalty*:
$$\Omega = R < g(\vec{x}) >^2$$
where:
$$< \alpha >= \begin{cases} \alpha & \text{if } \alpha \leq 0 \\ 0 & \text{if } \alpha > 0 \end{cases}$$
It defines an exterior penalty function. Stationary points of $\phi(\vec{x})$ may indeed be infeasible. On the other hand, note that feasible and infeasible points are handled equally well by the term, and in fact, no penalty is assigned to feasible points (either interior or in the boundaries of the feasible region). $\phi(\vec{x})$ exists everywhere and is continuous. R is chosen positive and is increased after each unconstrained stage (see Figure 3.5-e).

3.4.2 Direct methods

These methods operate on the constraints explicitly, that is, they directly take into account the constraint in the course of the optimization iterations. They are useful when solving

problems whose functions are discontinuous or nondifferentiable. Basically, there are two families of methods (both heuristic-based):

1. Those that are essentially adaptations of the unconstrained direct-search methods [156].

2. Those that rely upon the random selection of trial points.

Direct-search methods like Conjugate Directions have been modified to solve constrained optimization problems. In this case, modifications regarding how to choose advisable search directions based on the constraints have been proposed, most of them requiring the use of constraints gradient information [55]. Other methods like the Pattern Search have also been modified [81] to deal with constrained search spaces. In this case, the step-size used in either pattern or explorative moves can be reduced when leading to infeasible solutions. Besides, constraints gradient values are required.

One of the methods that have enjoyed wide use in engineering applications is the complex method. Based on the simplex method by Nelder and Mead [136], and modified by Box [21], the Complex method works by generating and maintaining a pattern of search points and the use of projections of undesirable points through the centroid of the remaining points as the means of finding new trial points. Box [21] proposed to take into account the feasibility of solutions when generating a new set of search points. One of the main drawbacks of the method is that it requires an initial feasible point. Furthermore, the complex method requires the feasible region to be a convex set.

Random search methods are used mainly to find an initial feasible point to be used for direct-search methods like the aforementioned Complex [21]. Random search methods are usually coupled with heuristic hill-climbing methods [156]. These random-based methods are useful to find the vicinity of the optimum, but they are quite inefficient if a closer estimate of the solution is required. Furthermore, these approaches have problems when dealing with problems with a high dimensionality.

Finally, there are methods based on the linearization of a nonlinear problem in order to apply linear programming techniques [156], but they are beyond the discussion of this work.

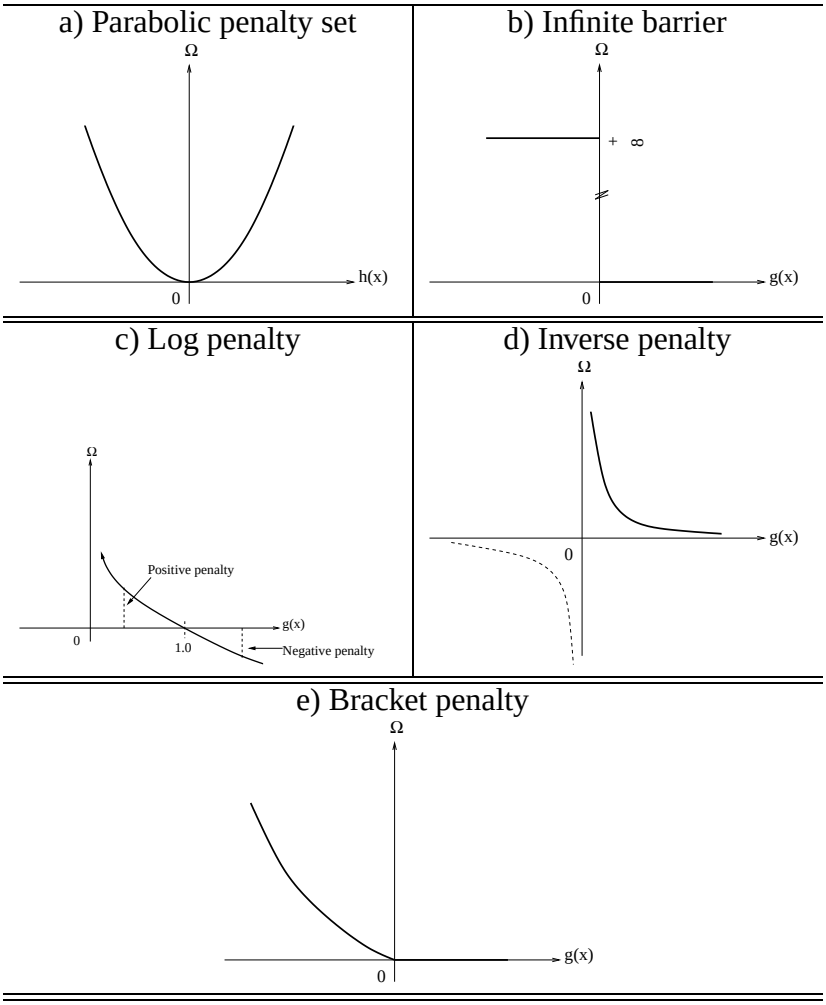


Figure 3.5: Different penalty terms.

Chapter 4

Constraint Handling in Evolutionary Algorithms

EAs are unconstrained search techniques. Thus, incorporating constraints into the fitness function of an EA is an open research area. There is a considerable amount of research regarding mechanisms that allow EAs to deal with equality and inequality constraints; both type of constraints can be linear or nonlinear. Constraint-handling approaches tend to incorporate information about infeasibility (or distance to the feasible region) into the fitness function in order to guide the search. In this chapter, we present a classification and descriptions of constraint-handling approaches used in EAs. In fact, three of the most competitive techniques (stochastic ranking [162], ASCHEA [77] and the homomorphous maps [110]) are used to compare the results obtained by our proposed approach.

4.1 Why to Use a Heuristic

Despite the large number of optimization techniques developed in mathematical programming [156, 150], several problems present characteristics that make them difficult to solve for any of them. For example, problems with nondifferentiable objective functions (and perhaps even nondifferentiable constraints), problems with disjoint feasible regions and problems with objective functions not available in algebraic form will be, in general, difficult for any mathematical programming technique. In fact, as known from mathematical programming, when dealing with the general nonlinear optimization problem, it is relatively simple to state problems in which the Kuhn-Tucker conditions for optimality do not hold and no mathematical programming technique can guarantee convergence to the global optimum [81]. It is precisely in all of those problems in which the use of EAs results

particularly useful and highly competitive, since EAs do not require that the objective function or the constraints of a problem are continuous nor differentiable. Additionally, being population-based techniques, EAs are less prone to get trapped in local optima, even when dealing with fairly large and complex search spaces.

Most constraint-handling approaches used with EAs tend to deal only with inequality constraints. However, in those cases, equality constraints are transformed into inequality constraints of the form:

$$|h_j(\vec{x})| - \epsilon \leq 0 \tag{4.1}$$

where ϵ is the tolerance allowed (a very small value). In the remainder of this document, some of the test functions adopted will have one or more equality constraints. Such equality constraints are transformed into inequality constraints using equation 4.1. Thus, it is important to remark that for these functions, the results found may "improve" the global optimum solution due to the numerical precision considered when defining the tolerance ϵ.

4.2 Penalty Function

The most common form of G_i and L_j is:

$$G_i = \max[0, g_i(\vec{x})]^\beta \tag{4.2}$$

$$L_j = |h_j(\vec{x})|^\gamma \tag{4.3}$$

where β and γ are normally 1 or 2.

Several approaches have been proposed to avoid this dependency of the values of the penalty factors. The most common are the following:

4.2.1 Death penalty

In this case, infeasible solutions are either rejected (and randomly generated again), or get a zero fitness regardless of their amount of constraint violation [8, 169]. No more operations are needed to determine closeness to the feasible region. It is recommended only for convex search spaces and feasible regions of a considerable size with respect to the whole search space. However, death penalty does not use any information about how close is a solution to the feasible region. Its main drawback is for complex problems where the initial population has no feasible solutions. Then, the evolutionary process will "stagnate" because all the individuals will have the same fitness (i.e. zero). Death penalty only works with problems

with inequality constraints due to the fact that there is an obvious difficulty of generating solutions which satisfy equality constraints.

4.2.2 Static penalties

In this case, the penalty factors remain without change during all the evolutionary process. There are several proposals: Kuri [112] calculates the fitness of an individual in the following way:

$$\text{fitness}(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if the solution is feasible} \\ K - \sum_{i=1}^{s}(\frac{K}{m}) & \text{otherwise} \end{cases} \tag{4.4}$$

where s is the number of constraints satisfied, m is the total number of equality and inequality constraints K is a constant (1x10^9). When an individual is infeasible its fitness is calculated depending of the number of violated constraints s. Then, all solutions that violate the same number of constraints will have the same fitness value, regardless of their different distances to the feasible region.

This approach has provided good results with different types of problems. However, sometimes it requires the use of another algorithm to generate feasible solutions in the initial population.

Homaifar et al. [87] proposed to use violation levels where the penalty factors are assigned depending of the amount of violation for each constraint of the problem. The expression is as follows:

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \sum_{i=1}^{m}\left(R_{k,i} \times \max\left[0, g_i(\vec{x})\right]^2\right) \tag{4.5}$$

where $R_{k,i}$ are the penalty coefficients used, m is total the number of constraints (Homaifar et al. [87] transformed equality constraints into inequality constraints), $f(\vec{x})$ is the unpenalized objective function, and $k = 1, 2, \ldots, l$, where l is the number of levels of violation defined by the user. The idea is to define specific penalty factors for each constraint and for each level of violation in order to penalize in more detail with a set of predefined rules.

This approach requires many parameters ($m(2l + 1)$ parameters in total) to be defined by the user which makes it difficult to apply in problems with many constraints.

Hoffmeister & Sprave have proposed to use the following penalty function [84]:

$$\text{fitness}(\vec{x}) = f(\vec{x}) \pm \sqrt{\sum_{i=0}^{m} H[-g_i(\vec{x})]g_i(\vec{x})^2} \tag{4.6}$$

where $H : \mathcal{R} \rightarrow \{0, 1\}$ is the Heavyside function:

$$H(y) = \begin{cases} 1 & : \quad y > 0 \\ 0 & : \quad y \leq 0 \end{cases} \tag{4.7}$$

This is equivalent to a partial penalty approach and was successfully used in some real-world problems [168].

The problem with Hoffmeister & Sprave's approach is that it is based on the assumption that infeasible points will always be valuated worse than feasible ones, and that is not always the case [127].

One of the most competitive approaches to handle constraints known to date uses a static penalty function: the Stochastic Ranking (SR) technique proposed by Runarsson & Yao [162]. The aim of the approach is to balance the influence of the objective function and the penalty function when assigning fitness to a solution. SR does not require the definition of a penalty factor. Instead, the selection process is based on a ranking process and a user-defined parameter called P_f that sets the probability of using only the objective function to compare two solutions when sorting them. Then, when the solutions are sorted using a bubble-sort like algorithm, sometimes, depending of the P_f value, the comparison between two adjacent solutions will be performed using only the objective function. The remaining comparisons will be performed using only the penalty function that consists, in this case, of the sum of constraint violation. The suggested range for the P_f value is $0.4 < P_f < 0.5$. The results obtained using all the functions of the well-known benchmark from Michalewicz and Schoenauer [133] are the best reported to date in the literature. Runarsson & Yao used a $(30, 200)$-ES with $350,000$ evaluations of the fitness function. One drawback of the approach is that the user needs to define the parameter P_f. The sorting algorithm adopted by this approach is shown in Figure 4.1.

4.2.3 Dynamic penalties

The idea of a dynamic penalty approach is to use time (i.e. the current generation number) to influence the computation of the penalty factor of an individual. During the first generations of the process, the penalty factor will be low. However, late in the process, the penalty will become very severe because it is assumed that the feasible region has been reached. Dynamic penalty functions share the same problems with the static penalty functions. If a bad factor is chosen, the EA may converge to either non-optimal feasible solutions (if the penalty is too high) or to infeasible solutions (if the penalty is too low)

Joines and Houck [96] use the generation number to adapt the value of the penalty factor. At generation t and assuming minimization this dynamic function increases the penalty through generations:

```
Begin
    For i=1 to N
        For j=1 to P-1
            u=random(0,1)
            If (φ(I_j) = φ(I_{j+1}) = 0) or (u < P_f)
                If (f(I_j) > f(I_{j+1}))
                    swap(I_j,I_{j+1})
            Else
                If (φ(I_j) > φ(I_{j+1}))
                    swap(I_j,I_{j+1})
        End For
        If (not swap performed)
            break
    End For
End
```

Figure 4.1: Stochastic Ranking sort algorithm [162]. I is an individual of the population. $\phi(I_j)$ is the sum of constraint violation of individual I_j. $f(I_j)$ is the objective function value of individual I_j

.

$$\text{fitness}(\vec{x}) = f(\vec{x}) + (C \times t)^{\alpha} \times SVC(\beta, \vec{x}) \tag{4.8}$$

where C, α and β are constants defined by the user (the authors used $C = 0.5$, $\alpha = 1$ or 2, and $\beta = 1$ or 2), and $SVC(\beta, \vec{x})$ is defined as [96]:

$$SVC(\beta, \vec{x}) = \sum_{i=1}^{n} D_i^{\beta}(\vec{x}) + \sum_{j=1}^{p} D_j(\vec{x}) \tag{4.9}$$

and

$$D_i(\vec{x}) = \begin{cases} 0 & g_i(\vec{x}) \leq 0 \\ |g_i(\vec{x})| & \text{otherwise} \end{cases} \quad 1 \leq i \leq n \tag{4.10}$$

$$D_j(\vec{x}) = \begin{cases} 0 & -\epsilon \leq h_j(\vec{x}) \leq \epsilon \\ |h_j(\vec{x})| & \text{otherwise} \end{cases} \quad 1 \leq j \leq p \tag{4.11}$$

One disadvantage of the approach is its sensitivity to the parameters C, α and β. However, for some problems it has provided good results [125].

Another idea to adapt the penalty factor according to the generation number was proposed by Kazarlis & Petridis [99] and is called Varying Fitness Function Technique or VFF. Their proposed expression used to solve the cutting stock problem and the unit commitment problem is:

$$\text{fitness}(\vec{x}) = f(\vec{x}) + V(g) \times \left(A \sum_{i=1}^{m} (\delta_i \cdot w_i \cdot \Phi(d_i(S))) + B \right) \times \delta_s \tag{4.12}$$

where A is a "severity" factor, m is the total number of constraints, δ_i is 1 if the constraint i is violated and 0 otherwise, w_i is a weight factor for constraint i, $d_i(S)$ is a measure of the degree of violation of constraint i introduced by solution S, $\Phi_i(.)$ is a function of this measure, B is a penalty threshold factor, δ_s is a binary factor ($d_s = 1$ if S is infeasible and is zero otherwise), and $V(g)$ is an increasing function of g (the current generation) in the range $(0 \ldots 1)$.

The dynamic part is the $V(g)$ function which increases with respect to the generation number. The authors suggest:

$$V(g) = \left(\frac{g}{G} \right)^2 \tag{4.13}$$

where g is the current generation number and G is the total number of generations. The VFF provided good results on the two problems used to test it. The main drawback of the approach is that it requires several parameters that depend on the problem and whose definition is not clear at all. VFF has also been used in a later approach by Kazarlis et al. [100]. They proposed to use a micro genetic algorithm (a GA with an very small population of about 5 individuals) inside a traditional GA in order to use it as a generalized hillclimber. The idea was to employ the traditional GA to perform the global search and to use the micro GA to perform local search. They tested it using engineering design problems and the results were competitive, but the problem with the approach is the definition of the parameters for each GA and also the parameters of the penalty function. Moreover, the approach has not been tested with other type of problems, like the well-known benchmark for evolutionary constrained optimization proposed by Michalewicz & Schoenauer [133].

Crossley and Williams [40] performed an experiment with several dynamic penalty coefficients that depended on the generation number. Several types of increments were tested like: constant, linear, quadratic, exponential, standard deviation of the population's fitness and variance of the population's fitness. The authors reported that dynamic updates were better than constant coefficients. They concluded that the best dynamic penalty is really problem-dependent if the goal is to find a good result in the minimum number of generations.

4.2.4 Annealing penalties

Annealing Penalties is a particular case of dynamic penalty functions based on the idea of simulated annealing [105]. Michalewicz and Attia [129], in their approach called GENOCOP II, change the penalty factor once in many generations. The penalty is increased over time (i.e. the temperature decreases over time) so that infeasible individuals are heavily penalized in the last generations. They divide the constraints into four groups: linear equalities, linear inequalities, nonlinear equalities and nonlinear inequalities. The expression used by them is:

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \frac{1}{2\tau}\sum_{i\in\mathcal{A}}\phi_i^2(\vec{x}) \tag{4.14}$$

where τ is the cooling schedule [105], and

$$\phi_i(\vec{x}) = \begin{cases} \max[0, g_i(\vec{x})] & \text{if } 1 \leq i \leq n \\ |h_i(\vec{x})| & \text{if } n+1 \leq i \leq m \end{cases} \tag{4.15}$$

Its main drawback is the sensitivity to the cooling schedule. Carlson [25] proposed a similar approach based on the following expression:

$$\text{fitness}(\vec{x}) = \mathcal{A} \cdot f(\vec{x}) \tag{4.16}$$

where $\mathcal{A}$ depends on two parameters: M, which measures the amount by which a constraint is violated (it takes a zero value when no constraint is violated), and T, which is a function of the running time of the algorithm. T tends to zero as evolution progresses. Using the basic principle of simulated annealing, Carlson et al. [25] defined $\mathcal{A}$ as:

$$\mathcal{A} = e^{-M/T} \tag{4.17}$$

so that the initial penalty factor is small and it increases over time. This will discard infeasible solutions in the last generations.

As in Michalewicz's approach [129], the critical issue in Carlson's approach [25] is the definition of the initial and final values of the temperatures.

4.2.5 Adaptive penalties

The aim of adaptive penalties is to use information of the evolutionary process itself (instead of a pre-defined variation function as in the case of dynamic penalties) to update the value of the penalty factors.

Bean and Hadj-Alouane [14, 71] developed a method that has a penalty function which uses feedback from the search process. Each individual is evaluated using the formula:

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \lambda(t) \left[\sum_{i=1}^{n} g_i^2(\vec{x}) + \sum_{j=1}^{p} |h_j(\vec{x})| \right] \tag{4.18}$$

where $\lambda(t)$ is updated at every generation t in the following way:

$$\lambda(t+1) = \begin{cases} (1/\beta_1) \cdot \lambda(t), & \text{if case } \#1 \\ \beta_2 \cdot \lambda(t), & \text{if case } \#2 \\ \lambda(t), & \text{otherwise,} \end{cases} \tag{4.19}$$

where cases #1 and #2 denote situations where the best individual in the last k generations was always (case #1) or was never (case #2) feasible, $\beta_1, \beta_2 > 1$, $\beta_1 > \beta_2$, and $\beta_1 \neq \beta_2$.

The penalty component $\lambda(t+1)$ for the generation $t+1$ is decreased if all the best individuals in the last k generations were feasible or is increased if they were all infeasible. If there are some feasible and infeasible individuals tied as best in the population, then the penalty does not change. This suggests that the approach is able to explore the boundaries of the feasible region. Nonetheless, its main drawback is the definition of the extra parameters β_1, β_2 and λ_0.

There are several proposals based on adaptive penalty approaches that have been used to solve combinatorial optimization problems like Smith and Tate's approach [174] approach where the magnitude of the penalty is dynamically modified according to the fitness of the best solution found so far. Norman & Smith [138] further applied Coit & Smith's approach to facility layout problems. Yokota et al. [187] proposed a variant of Smith, Tate and Coit's approach in which they use a multiplicative form of the fitness function (instead of an addition as in Smith et al. [174]). Gen and Cheng [63] introduced an approach based on a more severe penalty for infeasible solutions. Eiben & van der Hauw [51] proposed an adaptive penalty function that was successfully applied to the 3-coloring problem.

Rasheed [151] proposed one of the first adaptive penalties used for numerical optimization, inspired on Smith & Tate's approach [174], proposed to adjust the penalty factors so that there is always an equilibrium between feasible and infeasible solutions in the population in his Genetic Algorithm Design Optimization (GADO). The idea is to compare the solution with the lowest sum of constraint violation and the solution with the best fitness value. If they are the same, then the penalty factor is adequate, if not, it is increased. This approach was successfully applied to several engineering optimization problems [151].

Hamda & Schoenauer [75] proposed the use of an adaptive penalty function which adapts its value depending of a desired ratio of feasible solutions in the population. In this case, the user must define an interval within which this ratio should oscillate. This work was tested on a few test problems and it provided good results. The proposal by Hamda

& Schoenauer [75] was the first step for the development of one of the most competitive constraint-handling approaches known to date, which is called Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA). This approach was proposed by Hamida & Schoenauer [76, 77]. ASCHEA is based on three components:

- *An adaptive penalty function*: The expression used is:

$$\text{fitness}(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if the solution is feasible} \\ f(\vec{x}) - \text{penal}(\vec{x}) & \text{otherwise} \end{cases} \tag{4.20}$$

  where

$$\text{penal}(\vec{x}) = \alpha \sum_{j=1}^{q} g_j^+(\vec{x}) + \alpha \sum_{j=q+1}^{m} |h_j(\vec{x})| \tag{4.21}$$

  where $g_j^+(\vec{x})$ is the positive part of $g_j(\vec{x})$ and α is the penalty coefficient for all the constraints of the problem. The penalty factor is adapted according to a desired ratio of feasible solutions τ_{target} and the current ratio in the generation t, τ_t in the following way:

$$\begin{aligned} \text{if}(\tau_t > \tau_{\text{target}}) \quad & \alpha(t+1) = \alpha(t)/\text{fact} \\ \text{otherwise} \quad & \alpha(t+1) = \alpha(t) * \text{fact} \end{aligned}$$

  where fact > 1 and τ_{target} are user-defined parameters and

$$\alpha(0) = \left| \frac{\sum_{i=1}^{n} f_i(\vec{x})}{\sum_{i=1}^{n} V_i(\vec{x})} \right| * 1000 \tag{4.22}$$

  where $V_i(\vec{x})$ is the sum of constraint violation of individual i.

- *Constraint-driven recombination (crossover)*: Combine an infeasible solution with a feasible one and apply it when there is a low number of feasible solutions with respect to τ_{target}. If $\tau_t > \tau_{\text{target}}$ the recombination is performed in the traditional way.

- *Segregational selection based on feasibility*: The aim is to choose a defined ratio (τ_{select}) of feasible solutions based on their fitness to be part of the population for the next generation. The remaining individuals are selected in the traditional way (proportional selection) based on their penalized fitness. τ_{select} is another user-defined parameter.

In ASCHEA's new version [77], the authors propose to use a penalty factor for each constraint of the problem. Each factor is adapted independently:

$$\text{penal}(\vec{x}) = \sum_{j=1}^{q} \alpha_j g_j^+(\vec{x}) + \sum_{j=q+1}^{m} \alpha_j |h_j(\vec{x})| \tag{4.23}$$

and the adaptation process is now as follows:

$$\begin{aligned} \text{if}(\tau_t(j) > \tau_{target}) \quad & \alpha_j(t+1) = \alpha_j(t)/fact \\ \text{otherwise} \quad & \alpha_j(t+1) = \alpha_j(t) * fact \end{aligned}$$

also, the authors used a niching mechanism to improve the performance of the algorithm in multimodal functions. Finally, they added both, a dynamic and an adaptive scheme to decrease the tolerance value used to handle equality constraints. All these three new mechanisms also add more user-defined parameters, which makes more difficult to tune them to solve an specific problem. The approach uses a $(100 + 300)$-ES and requires $1,500,000$ fitness function evaluations to provide good results in 11 functions of the aforementioned benchmark for constrained evolutionary optimization [162].

One of the most recent constraint handling approaches is also based on an self-adaptive penalty function and was proposed by Farmani and Wright [56]. They prevent the user to define any extra parameter in their algorithm. The approach is implemented in three stages: (1) The normalized sum of constraint violation called "infeasibility" is assigned to each solution. (2) After that, the best and worst solutions in the population are identified. The best solution is that feasible solution with the best value of the objective function. If there are only infeasible solutions it will be the solution with the lowest sum of constraint violation (calculated in stage 1). The worst solution is calculated based on the best solution. The worst solution is usually that with the highest sum of of constraint violation and a lower value of the objective function. Finally (3) a two-part penalty function is applied only to infeasible solutions.

The two parts of the adaptive penalty function are:

- *First part*: Applies only if one or more infeasible solutions have a lower value of the objective function than the best solution. The goal of this first part is to increase the objective function value of the infeasible solutions such that the worst of the infeasible solutions has an objective function value that is equal to that of the best solution.

- *Second part*: It increases the objective function values such that the penalized objective function of the worst infeasible solution is equal to that of the worst objective individual.

The goal of this step manipulation of the penalty function is to let infeasible individuals close to the feasible region to have a similar fitness to that assigned to feasible solutions. It may help the search to sample the boundaries of the feasible region where the feasible global optimum is usually located for the hardest optimization problems.

The approach uses a binary-coded GA with gray codes and similar features to those used in the Homomorphous maps proposed by Koziel and Michalewicz [110]. In addition, the approach is tested using 11 benchmark problems. The results are competitive but the number of evaluations required to compete against other state-of-the-art approaches like the Stochastic Ranking [162] or ASCHEA [77] is quite high (about 1,400,000 evaluations of the objective function).

4.2.6 Co-evolutionary penalties

Coello [35] proposed to evolve the penalty factors in one subpopulation while evolving the solutions to the original problem in another subpopulation. In fact, there are two nested GAs, each one evolving one subpopulation. Therefore, the approach requires the definition of parameters for each GA and the time required by both GAs increases when more evaluations are necessary. The approach considers the amount of constraint violation and also the number of constraints violated. The expression is as follows:

$$\text{fitness}(\mathbf{X}) = f(\mathbf{X}) - (\text{coef} \times w_1 + \text{viol} \times w_2) \tag{4.24}$$

The high computational time required and the definition of several parameters makes the approach difficult to use in a wide set of problems.

4.2.7 Segregated genetic algorithm

Another way to have a balance between heavy and moderated penalty factors was proposed by Le Riche et al. [159]. The idea is to have a population of solutions. Each of them is evaluated using the two different penalty factors. After that, solutions are ranked in two lists, one based on a moderate penalization and the other one based on a highly penalized fitness. The best solutions from both lists are selected for reproduction. The best solutions between the parents and offspring will be in the population for the next generation. Linear ranking was used to decrease the high selection pressure that could cause premature convergence. This approach was used to solve a laminated design problem, providing excellent results [159]. The obvious disadvantage of the approach is to define two different penalty factors for any given problem because it has been only used to solve one specific problem.

4.2.8 Fuzzy penalties

Proposed by Wu & Yu [184]. They use a set of fuzzy rules to update the value of the penalty factor. The expression to assign fitness to a solution is:

$$\text{fitness}_i(\vec{x}) = \begin{cases} f_i(\vec{x}) & \text{If the solution is feasible} \\ f_i(\vec{x}) - r_f \cdot G(\vec{x}) & \text{otherwise} \end{cases} \tag{4.25}$$

where $G(\vec{x}) = \sum_{j=1}^{m} g_j^+(\vec{x}) + \sum_{i=1}^{k} |h_i(\vec{x})|$ and r_f is defined by a membership function which updates the penalty factor using information of the objective function values and the amount of constraint violation. The technique was evaluated using some benchmark functions and provided good results. However the approach was not compared against other state-of-the-art algorithms.

4.3 Special Representations and Operators

When the traditional representation of solutions (e.g. binary) is not suitable, some researchers have opted to propose alternative representations and associated operators suitable for the proposed representation. In most cases, special encodings are adopted to generate feasible solutions and *ad-hoc* operators are used to preserve their feasibility during all the evolutionary process. The main application of this approach is in problems in which it is extremely difficult to locate at least a single feasible solution, or in problems in which traditional encodings do not perform well.

- *Davis' applications*: Davis [46] proposes some examples of evolutionary algorithms with different representation and operators used to solve real world problems, but despite their success, they are hard to generalize for similar problems. For example, Davidor [44] used a varying-length genetic algorithm to generate robot trajectories, and defined a special crossover operator called analogous crossover. Other examples are schedule optimization [178], synthesis of neural networks architecture [78], and conformational analysis of DNA [117].

- *Random keys*: Proposed by James C. Bean [12, 13], and used to solve combinatorial problems like: job shop scheduling, parallel machine tool scheduling, and facility layout. Random Keys consists of a special representation which is used to eliminate the need of special crossover and mutation operators in certain sequencing and optimization problems. The effect is that all the permutations represented by real numbers are feasible. Despite its advantages, some researchers have reported a poor

performance of this encoding with respect to a traditional permutation representation [144, 145].

- *GENOCOP*: The (GEnetic algorithm for Numerical Optimization for COnstrained Problems) was developed by Michalewicz [128]. The first step of GENOCOP algorithm is to eliminate equality constraints and a number of variables. With the remaining inequality constraints forming a convex set, the EA starts its search with a population that consists of several copies of a feasible point obtained by sampling the feasible region or that was provided by the user The genetic operators are linear combinations to ensure that their offspring will also be feasible (the approach was proposed for convex search spaces). GENOCOP assumes a feasible starting point, which is an important drawback because for some problems is very difficult to generate one. Also, its application is limited because GENOCOP only deals with linear constraints.

- *Constraint consistent GAs*: Kowalczyk [107] proposed the use of constraint consistency [111] to prune the search space by preventing variable instantiations that are not consistent with the constraints of the problem. Kowalczyk used a special initialization procedure to generate feasible solutions or at least partially feasible solutions which are easier to find. He also used real numbers encoding with special operators to preserve feasibility. The approach has not been widely tested and its main drawback is to generate the initial partially feasible population.

- *Locating the boundary of the feasible region*: Many optimization problems usually have active constraints at the global optimum. Therefore, it is important to explore the boundaries of the feasible region. The idea, called strategic oscillation, was proposed by Glover [64]. Two components are required: (a) an initialization procedure that can generate feasible points, and (b) genetic operators that explore the feasible region. Schoenauer and Michalewicz [166] have proposed special initialization methods and special operators for two specific problems. The results provided are better than those provided by other methods, but the disadvantage, again, is the difficulty to generalize the special operators to other types of problems.

- *Decoders*: The emphasis of these approaches is to map chromosomes into feasible solutions of the problem to solve. Decoders must satisfy some conditions:

  - For each feasible solution s there must be a decoded solution d.
  - Each decoded solution d must correspond to a feasible solution s.

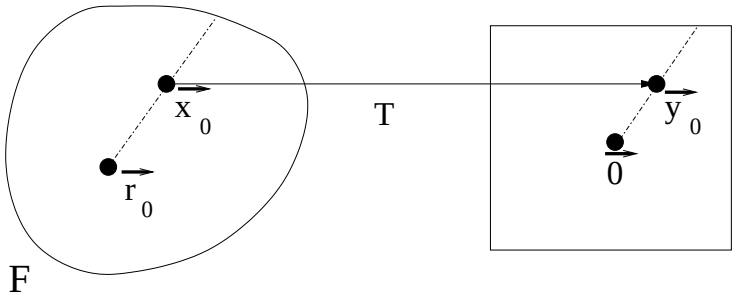


Figure 4.2: Projection of points between a convex feasible region and an n-dimensional cube $[-1, 1]^n$ (two-dimensional case) where T is the procedure to encode and decode

- All feasible solutions should be represented by the same number of decodings d.
- The transformation T is computationally fast.
- It has locality feature in the sense that small changes in the decoded solution result in small changes in the solution itself [43].

The best known decoder approach was proposed by Koziel & Michalewicz [109, 110]. They perform a homomorphous mapping (HM) between an n-dimensional cube and a feasible search space (either convex or non-convex). The main idea of this approach is to transform the original problem into another (topologically equivalent) function that is easier to optimize by the EA. HM handles two cases:

1. *Convex feasible region*: A mapping between a convex feasible region and an n-dimensional hypercube $[-1, 1]^n$ is performed (see Figure 4.2). It is important to note that an arbitrary point $\vec{y_0} \in [-1, 1]^n$ defines a segment from $\vec{0}$ to the perimeter of the hypercube. Also, the feasible point $\vec{x_0} \in \mathcal{F}$ (with respect to some reference point $\vec{r_0}$) corresponding to $\vec{y_0} \in [-1, 1]^n$ is $\vec{x_o} = \vec{r_0} + \vec{y_0} \cdot \tau$ where τ in its maximum value maps a point in the frontier of the feasible region $\mathcal{F}$.

   The definition of the domain of the optimized function is affected by the location of the reference point $\vec{r_0}$. The utopic location of $\vec{r_0}$ would be the geometrical center of $\mathcal{F}$. It is important because the EA does not optimize the original function but optimizes a topologically equivalent function. This version for convex feasible regions fulfills the properties of a good decoder (discussed above).

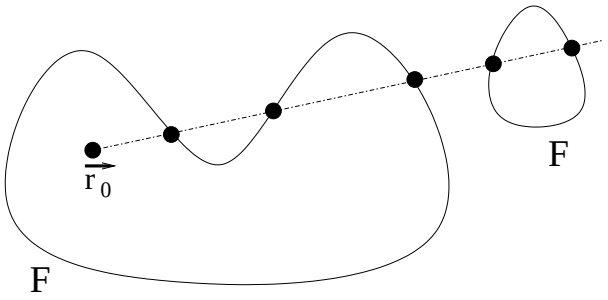


Figure 4.3: A segment intersecting more than one point of the boundaries of a nonconvex and disjoint feasible region.

2. *Nonconvex feasible region*: It is a more complex process because any segment L originated in the reference point $r_0 \in F$ can intersect the boundaries of the feasible region in more than one point. (See Figure 4.3). Then, the mapping process T must be re-designed. First, an additional one-to-one mapping between the $[-1, 1]^n$ hypercube and the search space S that is the Cartesian product of all the domains of the variables of the problem. Then, it is necessary to obtain segments between a reference point $\vec{r_0}$ and points $\vec{s_i}$ in the boundaries of the feasible region (see Figure 4.4). After that, the segments obtained are merged into one segment by another mapping process. A user-defined parameter v is used in this part of the mapping.

HM uses a binary-coded GA with Gray codes, proportional selection without elitism and traditional crossover and mutation operators. The results provided for 12 benchmark problems were very good. HM was considered the best algorithm for evolutionary constrained optimization before the Stochastic Ranking [162] technique was published. The main drawbacks of HM have to do with the fact that it is not an algorithm easy to implement. Also, the number of fitness function evaluations required by the approach is quite high (about 1,400,000). The version for convex feasible regions fulfills the properties of a good decoder (discussed above). However the version for nonconvex feasible solutions lacks the property of locality (last point of the list of the conditions for decoders shown above). Moreover, the user must define experimentally the v parameter for nonconvex feasible regions. Finally, HM requires a binary search to find the intersection of a line with the boundary of the feasible region (which is the core of the technique).

Kim and Husbands [102, 103] had an earlier proposal of a similar approach that used Riemann mappings to transform the feasible region into a simple geometric form.

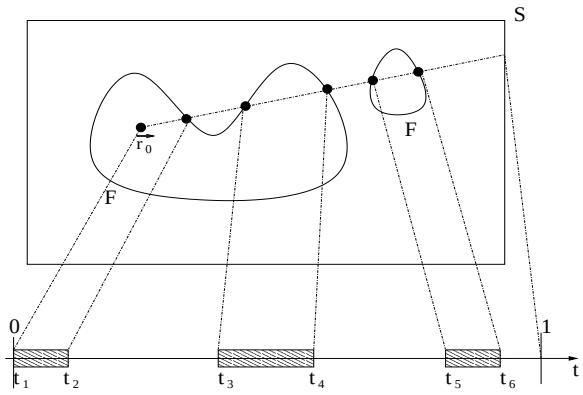


Figure 4.4: A nonconvex feasible region where the feasible intervals are mapped in the interval $[0, 1]^n$ (two-dimensional case).

They proposed two mappings: (1) Thurston packing circles and (2) a mesh generator. One of its main drawbacks are that the mapping process requires a considerable computational effort. Also, it has been used only to solve problems with a low number of decision variables.

4.4 Repair Algorithms

Repair in the context of constraint handling means to make feasible an infeasible solution. Such a repaired version can be used either for evaluation only, or it can also replace (with some probability) the original individual in the population.This idea has been widely used in combinatorial optimization, more than in numerical optimization. Some of the open questions related to repair algorithms are, for example, if the repaired solution must be inserted in the population or if it should be used for evaluating fitness [115, 116, 135]. Another question is how to design efficient and effective (and even generalizable) repair algorithms. One application of repair algorithms for numerical optimization was proposed by Michalewicz and his GENOCOP III [132]. The aim is to incorporate the original GENOCOP system [128] (which handles only linear constraints) and also use two different populations where results in one population influence evaluations of individuals in the other population. Individuals in the first population are search points which satisfy linear constraints of the problem. These solutions are kept as feasible by using special operators. Solutions in the second population are feasible reference points. Then, solutions from the first population are repaired in order to be similar to those of the second population. The main drawback of the approach is that the effort to repair an infeasible solution can be-

come more costly than the entire algorithm. Also, repair methods are not usually easy to generalize.

For combinatorial optimization, Liepins et al. [115, 116] have shown, through an empirical study on a diverse set of constrained combinatorial optimization problems, that a repair algorithm can provide better results than other approaches in both speed and performance.

Other areas of application of repair algorithms is robotics. Xiao et al. [186] used a repair algorithm to transform an infeasible path of a robot trying to move between two points in the presence of obstacles, so that the path would become feasible. The difficult part of this work was the design of the repair operators.

4.5 Separation of Constraints and Objectives

Unlike penalty functions which combine the value of the objective function and the constraints of a problem to assign fitness, these approaches handle constraints and objectives separately.

The most representative are:

- *Co-evolution*: Proposed by Paredis [141] to solve constraint satisfaction problems. His approach uses two populations: the first contains the constraints to be satisfied and the second contains potential, and maybe infeasible, solutions to the problem to be solved. Like in a predator-prey model, the selection pressure on members of one population depends on the fitness of the members of the other population [141]. A solution with a high fitness in the second population is a solution that satisfies many constraints. On the other hand, an individual with a high fitness in the population of constraints represents a constraint violated by many solutions in the second population. Individuals of both populations have encounters. A history of its encounters, and its fitness is computed according to the sum of the last n encounters. The approach provided impressive results but it has not been extended to numerical optimization.

- *Superiority of feasible points*: The idea is to assign always a higher fitness to feasible solutions. Powell and Skolnick [146] proposed to map feasible solutions into the interval $(-\infty, 1)$, and infeasible solutions into the interval $(1, \infty)$. Individuals are evaluated using [146]:

$$\text{fitness}(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if feasible} \\ 1 + r\left(\sum_{i=1}^{n} g_i(\vec{x}) + \sum_{j=1}^{p} h_j(\vec{x})\right) & \text{otherwise} \end{cases} \tag{4.26}$$

$f(\vec{x})$ is scaled into the interval ($-\infty$,1), $g_i(\vec{x})$ and $h_j(\vec{x})$ are scaled into the interval (1, ∞), and r is a constant.

A similar approach was proposed by Deb [49]. The expression to assign fitness to an individual is the following:

$$\text{fitness}_i(\vec{x}) = \begin{cases} f_i(\vec{x}) & \text{If the solution is feasible} \\ f_{\text{worst}} + \sum_{j=1}^{n} g_j(\vec{x}) & \text{otherwise} \end{cases} \tag{4.27}$$

where f_{worst} is the objective function value of the worst feasible solution in the current population. If there is no any feasible solution, f_{worst} is equal to 0.

The approach uses binary tournaments as a selection technique and it is based on the following criteria:

- A feasible solution is always preferred over an infeasible one.
- Between 2 feasible solutions, that one with the best value of the objective function is preferred.
- Between 2 infeasible solutions, that one with the lowest amount of constraint violation is preferred.

The main disadvantage of the approach is that it requires a niching mechanism to maintain diversity. It is worth reminding that niching requires at least one user-defined parameter. Besides, the approach uses high mutation rates in order to improve its explorative abilities.

Oyman and Deb [140] used Deb's approach [49], but with an evolution strategy as a search engine. They compare their approach against a death penalty. Oyman and Deb [140]'s approach outperformed the death penalty scheme. However, they tested it using only three problems (two of them are from the well-known benchmark from Michalewicz & Schoenauer [133]). Moreover, they only tested evolution strategies without a self-adaptation mechanism.

Another approach that uses the same criteria defined by Deb was proposed by Lampinen [113] for Differential Evolution (DE) [148]. Based on the direct comparison between parents and offspring used in DE, Lampinen proposed to use the aforementioned criteria to define the solution (between parents and offspring) that will survive for the next generation. The approach was tested with some benchmark functions [162] and the results were very promising, but the experimentation part was performed using different values for the parameters of the approach for each test function, which makes hard to compare this approach with respect to other state-of-the-art techniques.

- *CONGA (COnstraint based Numeric Genetic Algorithm)*: Proposed by Hinterding and Michalewicz [83]. The idea is to perform the search in two phases: (1) the search concentrates on finding feasible individuals and the objective function value is not used. As the amount of feasible individuals increases, the search focuses on fine-tuning the best of them. The aim of the selection proposed by the authors is to select the mate who best "complements" the parent previously selected. This mate should satisfy the constraints than the first selected parent does not satisfy. Therefore, the aim is that crossover will create new individuals who satisfy more constraints than any of their parents. CONGA was tested only with five benchmark functions and compared against GENOCOP II and III, but it requires further refinement and validation because it has problems to maintain diversity.

- *Behavioral memory*: Schoenauer and Xanthakis [167] proposed to satisfy, sequentially, the constraints of a problem. Death penalty is used because solutions that do not satisfy at least one constraint are eliminated from the population (the algorithm must generate new solutions that satisfy the number of constraints satisfied at the corresponding time of the process). Once a certain percentage of the population satisfies the first constraint, an attempt to satisfy the second constraint (while still satisfying the first) will be made. This method requires that there is a linear ordering of all constraints, and the order in which the constraints are processed influences the results provided by the algorithm. Also, it requires a sharing scheme to keep diversity in the population. Finally the approach has problems when the feasible region is quite large [167].

- *Multiobjective optimization concepts*: This category is detailed in Chapter 5.

4.6 Hybrid Methods

Within this category, we consider methods that are coupled with another technique (another heuristic or a mathematical programming approach) to deal with constrained spaces:

- *Constrained optimization by random evolution*: (CORE) is a hybrid approach proposed by Belur [16] which combines a random evolution search combined with the Nelder and Mead's simplex method [136]. When a solution is infeasible, the following constraint functional is minimized:

$$C(\vec{x}) = \sum_{i \in C_1} h_i^2(\vec{x}) - \sum_{j \in C_2} g_j(\vec{x}) \tag{4.28}$$

where

$$C_1 = \{i = 1, \ldots, n/|h_i(\vec{x})| > \varepsilon_c\} \tag{4.29}$$

$$C_2 = \{j = 1, \ldots, q/g_j(\vec{x}) < 0\} \tag{4.30}$$

and ε_c is the tolerance allowed in the equality constraints $h_i(\vec{x})$.

The approach has problems similar to those of repair-based algorithm.

- *Fuzzy logic*: T. Van Le [114] proposed to replace constraints of the form $g_i(\vec{x}) \leq b_i$ by a set of fuzzy constraints $C_1, \ldots, C_m$, $i = 1, \ldots, m$ defined as:

$$\mu C_i(\vec{x}) = \mu_{\sigma(b_i,\epsilon_i)}(g_i(\vec{x})), \;\; i = 1, \ldots, m \tag{4.31}$$

where ϵ_i is a positive real number that represents the tolerable violation of the constraints, and:

$$\mu_{\sigma(a,s)}(\vec{x}) = \begin{cases} 1 & \text{if } \; x \leq a, \\ \frac{e^{-p\left(\frac{x-a}{s}\right)^2} - e^{-p}}{1-e^{-p}} & \text{if } \; a < x \leq a+s \\ 0 & \text{if } \; x > a+s \end{cases} \tag{4.32}$$

The aim of these changes is to allow a broad constraint violation.

The fitness function is then redefined as:

$$\text{fitness}(\vec{x}) = f(\vec{x}) \times \min(\mu_{C_1}(\vec{x}), \ldots, \mu_{C_m}(\vec{x})) \tag{4.33}$$

This approach requires the definition by the user of two parameters. Additionally, it has not been tested widely [114].

- *Ant system*: AS was originally proposed to solve combinatorial optimization problems [50]. The main algorithm is a multi-agent system where low level interactions between single agents (i.e. artificial ants) result in a complex behavior of the whole ant colony. There have been also some attempts to use AS for numerical optimization. Bilchev and Parmee [19] proposed to represent a finite number of directions whose origin is a common base point called the *nest*. Since the idea is to cover eventually all the continuous search space, these vectors evolve over time according to the fitness values of the ants. To eliminate infeasible solutions Bilchev and Parmee [18, 19] proposed to use the concept of "unacceptable" food source for the ants. In

this way, some food sources that were acceptable before, will vanish, as constraints are tightened. This works analogously to a dynamic penalty function that eliminates directions to infeasible regions of the search space. Bilchev and Parmee obtained very good results [19] with their approach. However, several parameters must be tuned before using it. Moreover, an additional procedure has to be used to locate the *nest* and it is necessary to provide a model for the exhaustion of the food source to avoid that the ants pass through the same path more than once.

- *Simulated annealing hybrid*: Wah & Chen [179] proposed a hybrid of Simulated Annealing (SA) and a GA. The first part of the search is guided by SA. After that, the best solution is refined using a GA. To deal with constraints, Wah & Chen use Lagrangian Multipliers. Also, they calculate the optimal number of generations required by the algorithm. The results are good compared with respect to a dynamic penalty function and strategic oscillation. However, the main drawback of the approach is its high computational cost derived from the use of two heuristics.

- *Lagrangian multipliers* Adeli and Cheng [2] proposed a penalty function method hybridized with the primal-dual method [147]. It is based on sequential minimization of the Lagrangian method, and uses a fitness function of the form:

$$\text{fitness} = f(\vec{x}) + \frac{1}{2}\sum_{j=1}^{m}\gamma_j\left\{[g_j(\vec{x}) + \mu_j]^+\right\}^2 \tag{4.34}$$

where $\gamma_i > 0$, μ_i is a parameter associated with the ith constraint, and m is the total number of constraints. Also:

$$[g_j(\vec{x}) + \mu_j]^+ = \max[0, g_j(\vec{x}) + \mu_j] \tag{4.35}$$

μ_j must be defined in terms of the previously registered maximum violation of its associated constraint and scale it using a parameter β, which is a user-defined parameter. γ_j is increased using the parameter β. In this way, the penalty factor increases over generations. The approach was tested on engineering design problems providing good results. However, despite the fact that no penalty factors must be defined, the approach requires other parameters.

Another combination of Lagrangian multipliers with an EA was proposed by Kim and Myung [104, 134]. The approach guarantees the generation of feasible solutions during the search process. The fist phase of the algorithm consists on optimizing:

$$\text{fitness}(\vec{x}) = f(\vec{x}) + \frac{C}{2}\left(\sum_{i=1}^{n}(\max[0, g_i])^2(\vec{x}) + \sum_{j=1}^{p}|h_j(\vec{x})|^2\right) \tag{4.36}$$

where C is a constant. The first phase finishes when constraint violations have been decreased as much as the user wants. The second phase uses Lagrange multipliers to adjust the penalty function according to the feedback information received from the environment during the evolutionary process, in a way akin to the proposal of Adeli and Cheng [2]. The main disadvantage of the approach is the definition by the user of extra parameters required by the approach.

Smith [175] used a combination of a penalty function with an augmented Lagrangian multiplier. The expression to assign fitness in the presence of equality constraints is the following:

$$L_A(x, \lambda, \varrho, h_i) = f(\vec{x}) + \lambda h_i(\vec{x}) + \frac{\varrho}{2}h_i^2(\vec{x}) \tag{4.37}$$

and for inequality constraints the formula is:

$$L_A(x, \mu, \varrho, h_i) = f(\vec{x}) + \mu g_i(\vec{x}) + \frac{\varrho}{2}g_i^2(\vec{x}) \tag{4.38}$$

where ϱ is the penalization factor defined by the user and λ and μ are the Lagrangian multipliers. A small ϱ is chosen and the unconstrained problem is solved. The value of ϱ is incremented using the best solution obtained. Then, the new problem is solved. The approach has been used to solve problems with a moderate dimensionality (about 10 decision variables). However, like in a traditional penalty approach, the ϱ value is critical.

Finally, Barbosa proposed the use of a co-evolutionary GA coupled with an augmented Lagrangian function to solve constrained problems written as min-max problems. The idea is to use two populations, each one with an independent GA (A which minimizes and B that maximizes) to solve:

$$min_x max_{\lambda_i \in \mathbb{R}_+^m} \quad \phi(\vec{x}) + 0.5\sum_{i=1}^{m} r_i[(g_i(\vec{x}) + \theta_i)^2 + \vartheta] \tag{4.39}$$

where $\lambda_r = r_i\vartheta_i$ and the penalization is incremented by using $r_i^k = r_i^0 \times Z^k$ where $t > 1$ and $r_i^0 > 0$ $i = 1\ldots, m$ are user-defined parameters. $\mathbb{R}_+^m$ is the subset of vectors in $\mathbb{R}^m$ with nonnegative components.

The approach was tested with 4 benchmark problems. However, there is no discussion about the computational cost of it.

- *Immune system*: Hajela and Lee [72, 73] extended the Artificial Immune System to solve constrained optimization problems. The idea is to adapt infeasible solutions to the current feasible individuals using a bit matching process. The performance of the approach depends on the selection of antibodies (infeasible individuals) that are exposed to the antigens during the simulation. Another technique, called *expression strategies* was proposed by Hajela and Yoo [74]. In this case, feasible and infeasible individuals are combined using uniform crossover [177] in such a way that their chromosomic material is exchanged. The drawbacks of these approaches is that it is not clear if the genotype of an infeasible individual is more similar to the genotype of a feasible individual (it is important to remind that these approaches are based on bit to bit matching process). Coello & Cruz [36] used a negative selection model of an artificial immune system to solve constrained problems. They also proposed a parallel version of their algorithm [30] and the results obtained improved with respect to the serial version of the algorithm. The approach was tested using some benchmark problems. The main drawback is the definition by the user of some extra parameters.

- *Cultural algorithms*: Chung and Reynolds [27] proposed to use a hybrid of evolutionary programming and GENOCOP [131] in which they incorporated an interval constraint-network [45, 90] to represent the constraints of the problem at hand. Jin and Reynolds [94] proposed an n-dimensional regional-based schema, called *belief-cell*, as an explicit mechanism that supports the acquisition, storage and integration of knowledge about non-linear constraints in a cultural algorithm. This *belief-cell* can be used to guide the search of an EA (evolutionary programming in this case) by pruning the instances of infeasible individuals and promoting the exploration of promising regions of the search space. The key aspect of this work is precisely how to represent and save the knowledge about the problem constraints in the belief space of the cultural algorithm. A similar approach based on Evolutionary Programming coupled with the use of spatial data structures to improve the memory administration of the belief space was proposed by Coello and Landa [38]. They tested their approach with several benchmark problems and the results were very good.

  The main drawback of these types of techniques is the memory usage required for managing the belief maps mainly in problems with a high dimensionality.

Chapter 5

Use of Multiobjective Optimization Concepts to Handle Constraints

Among the several approaches that have been proposed as alternatives to the use of penalty functions, there is a group of techniques in which the constraints of a problem are handled as objective functions (i.e. a single-objective constrained problem is restated as an unconstrained multiobjective problem). This chapter is devoted to these techniques. The aim of this chapter is to explore the capabilities of using multiobjective optimization concepts to solve global optimization problems and to avoid the use of a penalty function, whose main idea is to combine the objective function and the constraints of a problem into a single value.

5.1 Basic Concepts

In this Section we will define some basic concepts from multiobjective optimization.

Definition 7 (General multiobjective optimization problem): *Find the vector* $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ *which will satisfy the* m *inequality constraints:*

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{5.1}$$

the p *equality constraints*

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{5.2}$$

and will optimize the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{5.3}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ *is the vector of decision variables.* □

Having several objective functions, the notion of "optimum" changes, because in multiobjective optimization problems, the aim is to find good compromises (or "trade-offs") rather than a single solution as in global optimization. The notion of "optimum" that is most commonly adopted is that originally proposed by Francis Ysidro Edgeworth in 1881 and later generalized by Vilfredo Pareto (in 1896) [142]. This notion is normally referred to as "Pareto optimality" and is defined next.

Definition 8 (Pareto optimality:): *A point* $\vec{x}^* \in \mathcal{F}$ *(*$\mathcal{F}$ *is the feasible region) is* **Pareto optimal** *if for every* $\vec{x} \in \mathcal{F}$ *and* $I = \{1, 2, \ldots, k\}$ *either (assuming minimization),*

$$\forall_{i \in I}(f_i(\vec{x}^*) \leq f_i(\vec{x})) \tag{5.4}$$

and, there is at least one $i \in I$ *such that*

$$f_i(\vec{x}^*) < f_i(\vec{x}) \tag{5.5}$$

□

In words, this definition says that $\vec{x}^*$ is Pareto optimal if there exists no feasible vector $\vec{x}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. The phrase "Pareto optimal" is considered to mean with respect to the entire decision variable space, unless otherwise specified.

Other important definitions associated with Pareto optimality are the following:

Definition 9 (Pareto dominance): *A vector* $\vec{u} = (u_1, \ldots, u_k)$ *is said to dominate* $\vec{v} = (v_1, \ldots, v_k)$ *(denoted by* $\vec{u} \preceq \vec{v}$*) if and only if u is partially less than v, i.e.* $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$. □

Definition 10 (Pareto optimal set): *For a given multiobjective optimization problem,* $\vec{f}(x)$*, the Pareto optimal set (*$\mathcal{P}^*$*) is defined as:*

$$\mathcal{P}^* := \{x \in \mathcal{F} \mid \neg\exists\ x' \in \mathcal{F}\ \ \vec{f}(x') \preceq \vec{f}(x)\}. \tag{5.6}$$

□

5.2 Multiobjective Optimization Concepts to Handle Constraints

The main idea of adopting multiobjective concepts to handle constraints is to redefine the single-objective optimization of $f(\vec{x})$ as a multiobjective optimization problem in which we will have $m+1$ objectives, where m is the total number of constraints (obviously, the additional objective is the original objective function of the problem). Then, we can apply any multiobjective optimization technique [39] to the new vector

$$\bar{v} = (f(\vec{x}), f_1(\vec{x}), \ldots, f_m(\vec{x})) \tag{5.7}$$

where

$$f_1(\vec{x}), \ldots, f_m(\vec{x}) \tag{5.8}$$

are the original constraints of the problem. An ideal solution $\vec{x}$ would thus have $f_i(\vec{x})$=0 for $1 \leq i \leq m$ and $f(\vec{x}) \leq f(\vec{y})$ for all feasible $\vec{y}$ (assuming minimization).

Three are the mechanisms taken from evolutionary multiobjective optimization that are the most frequently incorporated into constraint-handling techniques:

1. Use of Pareto dominance as a selection criterion.

2. Use of Pareto ranking [66] to assign fitness in such a way that nondominated individuals (i.e. feasible individuals in this case) are assigned a higher fitness value.

3. Split the population in subpopulations that are evaluated either with respect to the objective function or with respect to a single constraint of the problem. This is the selection mechanism adopted in the Vector Evaluated Genetic Algorithm (VEGA) [164]. In the remainder of the document we will refer to this mechanism as a "population-based" approach.

To solve single-objective optimization problems it is necessary to maintain a balance between feasible and infeasible solutions in order to sample the feasible region of the search space widely enough as to reach the global optimum solution.

In multiobjective optimization the goal is to find a set of trade-off solutions which are considered good in all the objectives to be optimized. In contrast, in global optimization we want to reach only the global optimum. Therefore, some changes must be done to those approaches to adapt them to reach only one solution: the global optimum. These new criteria are the following: Feasible solutions must be considered better than infeasible solutions and closeness to the feasible region should be used as a selection criteria. Furthermore, a mechanism to maintain diversity should be incorporated to avoid premature convergence.

5.3 A Review of Techniques

We will now provide a description of some of the most representative approaches that have been designed to apply the concepts discussed before.

COMOGA

Surry & Radcliffe [176] used a combination of the Vector Evaluated Genetic Algorithm (VEGA) [164] and Pareto Ranking to handle constraints in an approach called COMOGA (Constrained Optimization by Multi-Objective Genetic Algorithms).

In this technique, individuals are ranked depending of their sum of constraint violation (number of individuals dominated by a solution). However, the selection process is based not only on ranks, but also on the fitness of each solution. COMOGA uses a non-generational GA and extra parameters defined by the user (e.g. a parameter called ϵ is used to define the change rate of P_{cost}). One of these parameters is P_{cost}, that sets the rate of selection based on fitness. The remaining $1 - P_{cost}$ individuals are selected based on ranking values. P_{cost} is defined by the user at the beginning of the process and it is adapted during the evolutionary process using as a basis the percentage of feasible individuals that one wishes to have in the population.

COMOGA was applied to a gas network design problem and it was compared against a penalty function approach. Although COMOGA showed a slight improvement in the results with respect to a penalty function, its main advantage is that it does not require a fine tuning of penalty factors or any other additional parameter. The main drawback of COMOGA is that it requires several extra parameters, although its authors argue that the technique is not particularly sensitive to their values [176].

The algorithm of COMOGA [176] is presented in Figure 5.1.

VEGA

Coello [34] used a population-based approach similar to VEGA [164] to handle constraints in single-objective optimization problems. At each generation, the population was split into $m + 1$ subpopulations of equal fixed size, where m is the number of constraints of the problem. The additional subpopulation handles the objective function of the problem and the individuals contained within it are selected based on the unconstrained objective function value. The m remaining subpopulations take one constraint of the problem each as their fitness function. The aim is that each of the subpopulations tries to reach the feasible region corresponding to one individual constraint. By combining these different

Begin
Create M random solutions for the initial population.
Compute constraint violation for all solutions.
While stopping criterion is not satisfied **Do**
Rank solutions based on constraint violation (nondominance checking).
Evaluate the fitness of solutions.
Select two parents using the P_{cost} ratio based on
fitness and the remaining $1 - P_{cost}$ based on constraint ranking.
Apply genetic operators to generate one offspring s_{new}
If the new solution is better than the worst solution of the
population s_w **Then**
$s_w = s_{new}$
Endif
Adjust P_{cost}: Decreasing it favors feasible solutions; Increasing it
favors lower cost solutions (high fitness)
End While
End

Figure 5.1: Pseudocode of COMOGA [176]

subpopulations, the approach will reach the feasible region of the problem considering all of its constraints.

The algorithm of this approach is shown in Figure 5.2.

The fitness assignment scheme of the approach is the following:

if $g_j(\mathbf{x}) < 0.0$	**then**	fitness = $g_j(\mathbf{x})$
else if $v \neq 0$	**then**	fitness = $-v$
else		fitness = $f(\mathbf{x})$

where $g_j(\mathbf{x})$ refers to the jth constraint of the problem, v is the number of violated constraints ($v \leq m$) and $f(\mathbf{x})$ is the value of the objective function of the individual.

As can be seen above, each subpopulation tries to satisfy one single constraint. If the encoded solution does not violate the constraint of its corresponding subpopulation, then the fitness of an individual will be determined by the total number of constraints violated. Finally, if the solution is feasible, then the feasible criterion is to optimize the objective function. Therefore, any feasible individuals will be merged with the subpopulation on charge of optimizing the original (unconstrained) objective function.

Begin
Create M random solutions for the initial population.
Split the population into $m + 1$ sub-populations
Evaluate all M individuals
Assign a fitness value to all M individuals depending of their corresponding subpopulation.
While stopping criterion is not satisfied **Do**
Insert the best individual of the current population into the next population
While the next population is not full **Do**
Select 2 parents p_1 and p_2 based on **tournament selection with n candidates** from all the M individuals of the main population
Apply crossover to p_1 and p_2 to generate 2 offspring c_1 and c_2
Apply mutation to offspring c_1 and c_2
Insert c_1 and c_2 into the next population
End While
Split the population into $m + 1$ subpopulations
Evaluate all M new individuals
Assign a fitness value to all M individuals depending of their corresponding subpopulation.
End While
End

Figure 5.2: Pseudocode of Coello's version of VEGA for constraint-handling [34]

The genetic operators are applied to the entire population and every individual in a given subpopulation is allowed to mate with any other in any subpopulation (including its own, of course). In this way, individuals who satisfy constraints are combined with individuals with a good fitness value. At the end, it is expected to have a population of feasible individuals with high fitness values.

This approach was tested with some engineering problems [34] in which it produced competitive results. It has also been successfully used to solve combinational circuit design problems [37]. The main drawback of this approach is that the number of subpopulations required increases linearly with the number of constraints of the problem. This has some obvious scalability problems. Furthermore, it is not clear how to determine appropriate sizes for each of the subpopulations used.

MOGA

Coello [33] proposed the use of Pareto dominance selection to handle constraints in EAs. This is an application of Fonseca and Fleming's Pareto ranking process [62] (called Multi-Objective Genetic Algorithm, or MOGA) to constraint-handling. In this approach, feasible individuals are always ranked higher than infeasible ones. Based on this rank, a fitness value is assigned to each individual. This technique also includes a self-adaptation mechanism that avoids the usual empirical fine-tuning of the main genetic operators.

Coello's approach uses a real-coded GA with universal stochastic sampling selection (to reduce the selection pressure caused by the Pareto ranking process).

The algorithm of this approach is presented in Figure 5.3.

Begin
  Create M random solutions for the initial population.
  Evaluate the M individuals in the population.
  Compute the rank for each of the M individuals in the population.
  Assign a fitness value to all M individuals depending on rank
  While stopping criterion is not satisfied **Do**
    Insert the best individual of the current
    population into the next population
    While the next population is not full **Do**
      Select 2 parents p_1 and p_2
      using **Stochastic Universal Sampling**
      Apply crossover to p_1 and p_2 to generate 2 offspring c_1 and c_2
      Apply mutation to offspring c_1 and c_2
      Insert c_1 and c_2 into the next population
    End While
    Evaluate the M new individuals in the population
    Compute the rank for each one of the M
    individuals in the population.
    Assign a fitness value to all M individuals depending on rank
  End While
End

Figure 5.3: Pseudocode of Coello's version of MOGA to handle constraints [33]

To compute the **rank** of an individual $\vec{x}_i$, this approach uses the following procedure:

- Evaluate:

$$\text{rank}(\vec{x}_i) = \text{count}(\vec{x}_i) + 1 \tag{5.9}$$

where count($\vec{x}_i$) is computed according to the following rules:

1. Compare $\vec{x}_i$ against every other individual in the population. Assuming pairwise comparisons, we call $\vec{x}_j$ ($j = 1, \ldots, pop\_size$ and $j \neq i$) the other individual against which, x_i is being compared at any given time.
2. Initialize count($\vec{x}_i$)(for $i = 1, \ldots, pop\_size$) to zero.
3. If both $\vec{x}_i$ and $\vec{x}_j$ are feasible, then both are given a rank of zero and count($\vec{x}_i$) remains without changes.
4. If $\vec{x}_i$ is infeasible and $\vec{x}_j$ is feasible, then count($\vec{x}_i$) is incremented by one.
5. If both $\vec{x}_i$ and $\vec{x}_j$ are infeasible, but $\vec{x}_i$ violates more constraints than $\vec{x}_j$, then count($\vec{x}_i$) is incremented by one.
6. If both $\vec{x}_i$ and $\vec{x}_j$ are infeasible, and both violate the same number of constraints, but $\vec{x}_i$ has a total amount of constraint violation larger than the constraint violation of $\vec{x}_j$, then count($\vec{x}_i$) is incremented by one.

   If any constraint $g_k(\vec{x})$ ($k = 1, \ldots, m$, where m is the total amount of constraints) is considered satisfied if $g_i(\vec{x}) \leq 0$, then, the total amount of constraint violation for an individual $\vec{x}_i$ (denoted as coef($\vec{x}_i$)) is given by:

$$\text{coef}(\vec{x}_i) = \sum_{k=1}^{p} g_k(\vec{x}_i) \text{ for all } g_k(\vec{x}_i) > 0 \tag{5.10}$$

To compute **fitness**, the following rules are adopted:

1. If $\vec{x}_i$ is feasible, then rank($\vec{x}_i$) $= fitness(\vec{x}_i)$, else
2. rank($\vec{x}_i$) $= \frac{1}{\text{rank}(\vec{x}_i)}$

Then, individuals are selected based on rank($\vec{x}_i$) (stochastic universal sampling is used). Note that the values produced by $fitness(\vec{x}_i)$ must be normalized to ensure that the rank of feasible individuals is always higher than the rank of infeasible ones.

This approach has been used to solve some engineering design problems [33] in which it produced very good results. Furthermore, the approach showed great robustness and a relatively low number of fitness function evaluations with respect to traditional penalty functions. Additionally, it does not require any extra parameters. Its main drawback is the computational cost ($O(M^2)$, where M is the population size) derived from the Pareto ranking process.

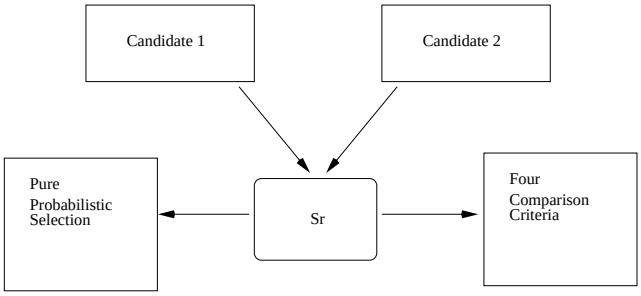


Figure 5.4: Diagram that illustrates the role of S_r in the selection process of Coello and Mezura's algorithm.

NPGA

Coello and Mezura [32] implemented a version of the Niched-Pareto Genetic Algorithm (NPGA) [88, 89] to handle constraints in single-objective optimization problems. The NPGA is a multiobjective optimization approach in which individuals are selected through tournaments based on Pareto dominance. However, unlike the original NPGA, Coello and Mezura's approach does not require niches (or fitness sharing [48]) to maintain diversity in the population. The NPGA is a more efficient technique than traditional multiobjective optimization algorithms, since it does not compare every individual in the population with respect to each other (as in traditional Pareto ranking), but uses only a sample of the population to estimate Pareto dominance. This is the main advantage of this approach with respect to Coello's proposal based on MOGA [33].

Note however that Coello and Mezura's approach requires an additional parameter called S_r that controls the diversity of the population. S_r indicates the percentage of parents selected by four comparison criteria described below. The remaining $1 - S_r$ parents will be selected by a pure probabilistic approach. Thus, this mechanism is responsible for keeping infeasible individuals in the population (i.e. the source of diversity that keeps the algorithm from converging to a local optimum too early in the evolutionary process).

A graphical illustration of the role of the parameter S_r is shown in Figure 5.4.

Tournaments in this approach are decided using as a basis four comparison criteria:

1. If both individuals are feasible, the individual with the higher fitness wins.

2. If one is feasible and the other is infeasible, the feasible individual wins.

3. If both are infeasible: Nondominance checking is applied (tournament selection as in the NPGA [89]).

4. If both are nondominated or dominated, the individual with the lowest amount of constraint violation wins.

The pseudocode of the algorithm of this approach is shown in Figure 5.5.

```
Begin
    Create M random solutions for the initial population.
    Evaluate the M individuals in the population.
    While stopping criterion is not satisfied Do
        Insert the best individual of the current
        population into the next population
        While the next population is not full Do
            Select 2 parents p1 and p2 based on S_r value
            Apply crossover to p1 and p2 to generate 2 offspring c1 and c2
            Apply mutation to offspring c1 and c2
            Insert c1 and c2 into the next population
        End While
        Evaluate the M new individuals in the population
    End While
End
```

Figure 5.5: Pseudocode of Coello & Mezura's constraint handling approach based on the NPGA [32]

This approach has been tested with several benchmark problems and was compared against several types of penalty functions [120]. Obtained results indicated that the approach was robust, efficient and effective. However, it was also found that the approach had scalability problems (its performance degrades as the number of decision variables increases).

Other approaches based on multiobjective concepts are the use of VEGA [164] by Parmee & Purchase [143]. Camponogara & Talukdar [24] proposed to combine a bi-objective problem definition with a linear search algorithm. An approach similar to a min-max formulation used in multiobjective optimization [26] combined with tournament selection was proposed by Jiménez and Verdegay [93]. Ray et al. [153] proposed the use of a Pareto ranking approach that operates on three spaces: objective space, constraint space and the combination of these two. Jiménez et al. [92] proposed an algorithm that uses Pareto dominance inside a preselection scheme to solve several types of optimization problems (multiobjective, constraint satisfaction, global optimization, and goal programming prob-

lems). Ray [152] explored an extension of his previous work on constraint-handling [153] in which the emphasis was robustness.

5.4 A Comparative Study

Four techniques were selected to perform a small comparative study that aims to illustrate some practical issues of constraint-handling techniques [79]. The techniques selected were the following: COMOGA [176], the use of VEGA proposed by Coello [34], the NPGA to handle constraints [32] and the approach that uses MOGA [33]. In order to simplify our notation, the last three techniques previously indicated will be called HCVEGA, HCNPGA and HCMOGA, respectively.

5.4.1 Experimental design

To evaluate the performance of the selected techniques, we decided to use the well-known benchmark proposed in [133] plus four engineering design problems used in [33]. The expressions of each test function are provided in Appendix A.

To get an estimate of how difficult is to generate feasible solutions for these problems, a ρ metric (as suggested by Michalewicz and Schoenauer [133]) was computed using the following expression:

$$\rho = |F|/|S| \tag{5.11}$$

where $|F|$ is the number of feasible solutions and $|S|$ is the total number of solutions randomly generated. In this work, $S = 1,000,000$ random solutions.
The different values of ρ and the main features for each of the functions chosen are shown in Table 5.1. It is important to note that the set of selected problems provides different levels of difficult to generate feasible solutions. Thus, this benchmark contains problems like g05, g07 and g13 whose feasible solutions are very difficult to generate, as well as problems like g02 and vessel where feasible solutions are relatively easy to find.

In our comparative study, we used a binary-gray-coded GA with two-point crossover and uniform mutation. Equality constraints were transformed into inequalities using a tolerance value of 0.001 (see Section 3.2). The number of fitness function evaluations is the same for all the approaches under study ($80,000$). The parameters adopted for each of the methods were the following:

Problem	**n**	**Type of function**	ρ	**LI**	**NI**	**LE**	**NE**
g01	13	quadratic	0.0003%	9	0	0	0
g02	20	nonlinear	99.9973%	2	0	0	0
g03	10	nonlinear	0.0026%	0	0	0	1
g04	5	quadratic	27.0079%	4	2	0	0
g05	4	nonlinear	0.0000%	2	0	0	3
g06	2	nonlinear	0.0057%	0	2	0	0
g07	10	quadratic	0.0000%	3	5	0	0
g08	2	nonlinear	0.8581%	0	2	0	0
g09	7	nonlinear	0.5199%	0	4	0	0
g10	8	linear	0.0020%	6	0	0	0
g11	2	quadratic	0.0973%	0	0	0	1
g12	3	quadratic	4.7697%	0	9^3	0	0
g13	5	nonlinear	0.0000%	0	0	1	2
beam	4	quadratic	2.6859%	6	1	0	0
vessel	4	quadratic	39.6762%	3	1	0	0
spring	3	quadratic	0.7537%	1	3	0	0
truss	10	nonlinear	46.8070%	0	22	0	0

Table 5.1: Values of ρ for the 17 test problems chosen. n is the number of decision variables, LI is the number of linear inequality constraints , NI the number of nonlinear inequality constraints, LE is the number of linear equality constraints and NE is the number of nonlinear equality constraints.

- COMOGA:
  - Population size = 200
  - Crossover rate = 1.0
  - Mutation rate = 0.05
  - Desired proportion of feasible solutions = 10 %
  - $\epsilon = 0.01$
- HCVEGA:
  - Population size = 200
  - Number of generations = 400
  - Crossover rate = 0.6
  - Mutation rate = 0.05
  - Tournament size= 5
- HCNPGA:
  - Population size = 200
  - Number of generations = 400
  - Crossover rate = 0.6
  - Mutation rate = 0.05
  - Size of sample of the population = 10
  - Selection ratio = 0.8
- HCMOGA:
  - Population size = 200
  - Number of generations = 400
  - Crossover rate = 0.6
  - Mutation rate = 0.05

A total of 100 runs per technique per problem were performed. Statistical results are presented in Tables 5.2, 5.3, 5.4 and 5.5. The symbol "*" and the number in parenthesis "(n)"mean that only in n runs the approach was able to reach the feasible region, $0.0 \leq F_p \leq 1.0$ is the average percentage of feasible solutions found during a single run (with respect to the entire population).

	COMOGA						
P	**Optimal**	**Best**	**Median**	**Mean**	**St. Dev.**	**Worst**	F_p
g01	−15.000	−5.000	0.000	−0.673	1.35E+0	0.000	0.001
g02	0.803619	0.027258	0.021285	0.020641	3.60E-3	0.008432	0.999
g03	1.000	0.126	0.000	−1399999.995	3.47E+6	–	0.0002
g04	−30665.539	−30533.057	−30328.199	−30329.563	74.79E+0	−30141.033	0.002
g05	5126.498	–	–	–	–	–	–
g06	−6961.814	−6808.696	−5408.743	−5255.105	9.94E+2	−1341.318	0.002
g07*(8)	24.306	485.579	1987.981	1567.294	9.24E+2	3149.046	0.0003
g08*(99)	0.095825	0.095782	0.094613	0.094343	1.34E-3	0.089900	0.003
g09	680.63	733.001	979.748	983.626	1.15E+2	1314.536	0.011
g10*(71)	7049.25	10865.434	18994.847	18924.576	38.51E+2	26625.984	0
g11	0.75	**0.75**	**0.75**	**0.75**	0	**0.75**	0.0002
g12	1.000	−0.999	−0.999	−0.999	0	−0.993	0.070
g13	0.053950	–	–	–	–	–	–
beam	1.728226	1.835381	2.039148	2.030360	9.48E-2	2.328703	0.0006
vessel	6059.946341	6369.428223	7889.838867	7795.411538	7.01E+2	9147.520508	0.004
spring	0.012681	0.012929	0.014263	0.014362	8.64E-4	0.017113	0.021
truss	5152.636136	6283.198730	6675.126709	6660.455649	1.26E+2	6968.627441	0.007

Table 5.2: Experimental results using COMOGA with the 17 test problems. The "-" symbol means that no feasible solutions were found during the experiments. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

	HCVEGA						
P	**Optimal**	**Best**	**Median**	**Mean**	**St. Dev.**	**Worst**	F_p
g01	−15.000	−11.221	−9.702	−9.716	4.94E-1	−8.387	0.146
g02	0.803619	−0.000037	−0.000270	−0.000315	1.78E-4	−0.000983	0.999
g03	1.000	−0.000	−0.000	0.000	0	−0.000	0.002
g04	−30665.539	−30647.246	−30628.588	−30628.469	7.88E+0	−30607.240	0.408
g05	5126.498	–	–	–	–	–	–
g06	−6961.814	−6942.747	−6758.277	−6762.048	1.01E+2	−6516.471	0.043
g07	24.306	28.492	34.528	34.558	2.93E+0	41.786	0.154
g08	0.095825	**0.095825**	**0.095825**	**0.095825**	0	**0.095825**	0.393
g09	680.63	693.642	736.191	739.306	25.17E+0	806.855	0.046
g10*(63)	7049.25	9842.453	17708.880	17605.588	3.88E+3	27627.635	0.00005
g11	0.75	**0.75**	0.81	0.80	2.58E-2	0.85	0.011
g12	1.000	**1.000**	**1.000**	**1.000**	0	**1.000**	0.517
g13	0.053950	–	–	–	–	–	–
beam	1.728226	**1.726772**	1.735865	1.736439	5.93E-3	1.769726	0.346
vessel	6059.946341	6064.723633	6238.489746	6259.963745	1.70E+2	6820.944824	0.425
spring	0.012681	0.012688	0.012789	0.012886	2.09E-4	0.013784	0.25
truss	5152.636136	5327.418457	5453.446045	5455.871895	56.74E+0	5569.240723	0.676

Table 5.3: Experimental results using HCVEGA to handle constraints with the 17 test problems. The "-" symbol means that no feasible solutions were found during the experiments. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

	HCNPGA						
P	**Optimal**	**Best**	**Median**	**Mean**	**St. Dev.**	**Worst**	F_p
g01*(52)	−15.000	−10.565	−6.461	−6.496	2.18E+0	−1.276	0.000
g02	0.803619	0.785141	0.731922	0.724332	3.42E-2	0.528319	0.914
g03	1.000	0.975	0.866	0.861	5.4E-2	0.715	0.002
g04	−30665.539	−30661.033	−30635.347	−30630.883	20.47E+0	−30544.324	0.345
g05	5126.498	–	–	–	–	–	–
g06	−6961.814	−6941.307	−6748.525	−6644.539	3.36E+2	−5164.000	0.018
g07	24.306	26.986	30.882	31.249	2.32E+0	38.459	0.049
g08	0.095825	**0.095825**	**0.095825**	**0.095825**	0	**0.095825**	0.425
g09	680.63	680.951	682.121	682.335	8.36E-1	684.816	0.24
g10*(29)	7049.25	8183.303	12691.962	13716.704	48.04E+2	23585.121	0.0004
g11	0.75	**0.75**	**0.75**	**0.75**	1.2E-2	0.83	0.026
g12	1.000	**1.000**	**1.000**	**1.000**	0	**1.000**	0.446
g13	0.053950	–	–	–	–	–	–
beam	1.728226	**1.727014**	1.751112	1.766413	4.38E-2	1.978103	0.262
vessel	6059.946341	**6059.926270**	6127.618408	6172.527373	123.9E+0	6845.770508	0.331
spring	0.012681	0.012683	0.012736	0.012752	6.2E-5	0.013132	0.105
truss	5152.636136	5179.740723	5256.108154	5259.013174	37.66E+0	5362.890625	0.504

Table 5.4: Experimental results using HCNPGA to handle constraints with the 17 test problems. The "-" symbol means that no feasible solutions were found during the experiments. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

	HCMOGA						
Problem	**Optimal**	**Best**	**Median**	**Mean**	**St. Dev.**	**Worst**	F_p
g01	−15.000	−13.968	−12.900	−12.753	8.16E-1	−9.329	0.014
g02	0.803619	−0.425258	−0.512270	−0.515214	4.03E-2	−0.618569	0.999
g03	1.000	−0.036	−0.219	−0.247	1.43E-1	−0.601	0.010
g04	−30665.539	−30649.959	−30570.755	−30568.918	53.53E+0	−30414.773	0.345
g05	5126.498	8879.123	8879.977	8879.945	1.0E-1	8879.999	0.000
g06	−6961.814	−6939.440	−6699.262	−6678.926	15.6E+1	−6258.591	0.017
g07	24.306	29.573	40.376	45.589	15.17E+0	114.547	0.013
g08	0.095825	**0.095825**	**0.095825**	**0.095825**	0	**0.095825**	0.074
g09	680.63	681.708	689.956	692.966	10.96E+0	734.002	0.049
g10	7049.25	7578.336	9026.873	9504.359	15.06E+2	16473.984	0.011
g11	0.75	**0.75**	**0.75**	**0.75**	0	0.75	0.017
g12	1.000	**1.000**	**1.000**	**1.000**	0	**1.000**	0.090
g13	0.053950	–	–	–	–	–	–
beam	1.728226	1.729384	1.791587	1.825219	7.98E-2	2.126669	0.064
vessel	6059.946341	6066.969727	6561.483154	6629.064048	3.85E+2	7547.403320	0.453
spring	0.012681	0.012680	0.012815	0.012960	3.63E-4	0.014754	0.047
truss	5152.636136	5336.618652	5745.238281	5748.839526	2.10E+2	6474.041992	0.604

Table 5.5: Experimental results using HCMOGA to handle constraints with the 17 test problems. The "-" symbol means that no feasible solutions were found during the experiments. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

5.4.2 Discussion of results

We will present next a discussion of the results obtained from the experimental design previously proposed. We will discuss three aspects: Quality, of the results achieved, consistency and capability of each approach to maintain diversity.

Quality of the results

HCNPGA gave the best results in 9 problems and in six of them it reached the global optimum. HCMOGA is superior in 7 of them and it reached the global optimum in 4. HCVEGA got better results only in 4 problems and it found the global optimum in 3. COMOGA was clearly surpassed because it did not give the best results in any given problem.

Consistency

The statistics indicate that HCNPGA and HCVEGA presented the lowest standard deviation and the best median and average solutions in the same number of problems (7). The second best approach was HCMOGA which produced the best results in 4 problems. COMOGA was the best in only 2 functions. Despite the fact that there was a tie between HCNPGA and HCVEGA, the last one showed premature convergence in most of the test problems.

Diversity

An utopical behavior for an ideal constraint handling technique is defined in our case as keeping at all times half of the population with feasible solutions and the other half with infeasible ones. However, in practice such a balance may be very difficult to achieve. Therefore, we provide a relative comparison among the approaches under study. The most balanced approach based on the number of feasible and infeasible solutions in the population during all the evolutionary process was HCVEGA. HCNPGA ranked second in this category. HCMOGA had results very close to those of HCNPGA. This suggests that a population-based approach should help to maintain diversity. Pareto dominance as a selection criteria is helpful too, based on the results obtained by HCNPGA. However, Pareto Ranking seems to give a less balanced population, probably because of its higher selection pressure.

The overall poor performance provided for all four approaches suggests that maintaining infeasible solutions is not the only goal in order to obtain better results. Perhaps, the utopical number of 50% of infeasible solutions is not as useful as we first thought. This

Category	Problem
Very difficult	g05, g13
Difficult	g01, g02, g03, g10
Average	g04, g06, g07, g09, vessel, truss
Easy	g08, g11, g12, beam, spring

Table 5.6: Classification of test functions based on the experimental phase.

discussion led us to believe that it is necessary to know what type of infeasible solutions are useful. This idea is further developed in Chapter 6.

Difficulty of the test problems adopted

Based on the results obtained, we classified the seventeen functions adopted in three categories: Very difficult, difficult, average and easy. The corresponding classification is presented in Table 5.6.

Remarks

Some issues can be stated based on the previous study:

- None of the four multiobjective-based techniques to handle constraints compared in this study work too well with large feasible regions. This means that these approaches have problems to move towards the global optimum once the feasible region is reached (e.g. g02 where the feasible region covers 99% of the search space).

- When the problem to be solved has more than one nonlinear equality (e.g. g05 and g13) it turns out to be very difficult to solve it using any of the approaches considered in this study.

- The performance of all the techniques compared, degrades in the presence of high dimensionality (e.g. g01 with 13 decision variables, g02 with 20 decision variables, g03 with 10 decision variables and g10 with 8 decision variables).

- The four compared techniques are efficient in problems with small feasible regions (joint, disjoint, non convex) formed by a significant number of inequality constraints (both linear and nonlinear).

- For problems with a low dimensionality (2 or 3 decision variables) and with either equality or inequality constraints (linear or nonlinear), the four compared approaches find very good results (g08, g11, and g12).

- HCVEGA prematurely converges to local optima in most of the problems.

- HCMOGA was the only one to find feasible solutions in one very difficult and one difficult function (g5 and g13). However the solutions found are far from the global optimum.

- The results suggest that splitting the population into subpopulations promotes a better diversity balance, but also presented premature convergence.

- Pareto dominance as a selection criterion has proved to give better results (in terms of optimality) than Pareto Ranking or a population-based approach. However, Pareto dominance cannot reach the global optima in problems with either high dimensionality, large feasible regions or several nonlinear equality constraints.

- The overall results of the COMOGA's steady-state GA suggests that this approach is not a good option to solve nonlinear global optimization problems. This is mainly due to its high selection pressure which tends to produce premature convergence.

5.4.3 Final conclusions of this study

Although not conclusive, this study seems to indicate that Pareto dominance, Pareto ranking and population-based mechanisms can solve constrained optimization problems. However they present limitations related with high dimensionality, equality constraints and large feasible regions. Thus, the overall results suggest that the use of multiobjective concepts to solve constrained problems with only one objective function is not adequate. The main reason is that, multiobjective optimization by definition aims to find good trade-off solutions, and such compromise solutions are not necessarily good when the search aims to find only the global optimum of a single-objective optimization problem. After this study was finalized, other authors came to similar conclusions in their research. Runarsson and Yao [163] concluded that the use of Pareto Ranking causes the search to spend most of the time searching in the infeasible region; therefore, the approach is unable to find feasible solutions (or finds feasible solutions with a poor value of the objective function). Hernandez et al. [80] proposed an approach named IS-PAES which is based on a multiobjective algorithm called Pareto Archive Evolution Strategy (PAES) originally proposed by Knowles and Corne [106]. IS-PAES uses an external memory to store the best set of solutions

found. Furthermore, IS-PAES requires a shrinking mechanism to reduce the search space. The multiobjective concept is used in this case as a secondary criterion (Pareto dominance is used only to decide whether or not a new solution is inserted in the external memory). The authors acknowledge that the most important mechanisms of IS-PAES are its shrinking procedure and the information provided by the external memory which is used to decide the shrinking of the search space. Furthermore, despite its good performance as a global optimizer, IS-PAES is an approach far from simple to implement.

As a final conclusion from this comparative study, we decided to keep the idea of handling the objective function and the constraints of the problem separately in order to avoid the use of a penalty function. However, we also decided to avoid the use of any multiobjective optimization concept in the selection mechanism of an evolutionary algorithm. Instead, we will put our emphasis on improving the sampling capabilities of the search engine used (this implies a change of evolutionary algorithm, as will be seen next), and in finding a constraint handling mechanism as simple as possible. All these objectives are further developed in Chapter 6.

Chapter 6

A Simple Evolution Strategy to Solve Constrained Problems

After deciding to handle the objective function and constraints separately and to avoid the use of multiobjective concepts, in this chapter we present a constraint handling technique coupled with a novel diversity mechanism based on maintaining good infeasible solutions. The approach is based on an evolution strategy (ES). We empirically show that the ES is able to sample the constrained search space in a more convenient way than a genetic algorithm under the same constraint handling and diversity mechanism.

Statistics presented here are based on samples of 30 independent runs. We show the best result (the closest approximation to the global optimum or best known solution), the mean, median, worst (the poorest approximation to the global optimum or best known solution) and standard deviations from these 30 runs. The discussion of results relates to finding which result is quantitatively better among those provided by the different approaches compared. In this work, a "better" result given by an approach means a closer approximation to the global optimum or best known solution than that provided by another compared method. A "similar" result provided by an algorithm means that an approximation to the optimum is very similar in its value (it only has a small difference of percentage error) to the corresponding approximation given by another technique, and this small difference is not enough to consider it a better solution. A "worst" result obtained by an algorithm indicates a poorer approximation to the global optimum or best known solution of a problem than the provided by another approach.

We discuss the obtained results based on quality and robustness. Quality refers to closeness of the best solution found with respect to the global optimum or best known solution (i.e. an approach which finds the global optimum in at least one run has a higher quality than an algorithm which only reaches a good approximation but not exactly the global op-

timum in its best result). An algorithm is considered robust when its mean and median values are close approximations to the global optimum or best known solution. Such behavior should also be accompanied by a low standard deviation. A robust algorithm is one that consistently reaches very good approximations of the global optimum or best known solution (or reaches the global optimum in every single run).

6.1 The Evolution Strategy

ES were proposed by Bienert, Rechenberg and Schwefel who used them to solve hydrodynamical problems [154, 171]. The first ES version was the $(1 + 1)$-ES which uses just one individual that is mutated using a normally distributed random number with mean zero and an identical stepsize for each decision variable. The best solution between the parent and the offspring is chosen and the other one is eliminated. Rechenberg derived a convergence rate theory and proposed a rule for changing the stepsize of mutations in a $(1+1)$-ES. This is the so-called "$1/5$-success rule" [155].

The first multimembered ES was the $(\mu + 1)$-ES, which was designed by Rechenberg and is described in detail in [8]. In this approach, μ parent solutions recombine to generate one offspring. This solution is also mutated and, if it is better, it will replace the worst parent solution. Note however that the $(\mu+1)$-ES has not been too popular in the literature. However, it provided the transition to the state-of-the-art multimembered ES.

The $(\mu + \lambda)$-ES and the (μ, λ)-ES were proposed by Schwefel [169]. In the first one, the best μ individuals out of the union of the μ original parents and their λ offspring will survive for the next generation. On the other hand, in the (μ, λ)-ES the best μ will be selected only from the λ offspring.

The $(\mu + \lambda)$-ES uses an implicit elitist mechanism and solutions can survive more than one generation. Meanwhile, in the (μ, λ)-ES solutions only survive one generation (this is the type of selection traditionally adopted in genetic algorithms [66]). Instead of the "$1/5$-success rule", each individual includes a stepsize value for each decision variable. Moreover, for each combination of two stepsize values, a rotation angle is included. These angles are used to perform a correlated mutation. This mutation allows each individual to look for a search direction. The stepsize values and the angles of each individual are called strategy parameters. They are also recombined and mutated. A $(\mu + \lambda)$-ES or (μ, λ)-ES individual can be seen as follows: $a(i)(\vec{x}, \vec{\sigma}, \vec{\theta})$, where i is the number of individual in the population, $\vec{x} \in \mathbb{R}^n$ is a vector of n decision variables, $\vec{\sigma}$ is a vector of n stepsize values and $\vec{\theta}$ is a vector of $n(n-1)/2$ rotation angles where $\theta_i \in [-\pi, \pi]$. One of the main differences between a genetic algorithm and en evolution strategy relies on the way in which a solution is represented (see Figure 6.1).

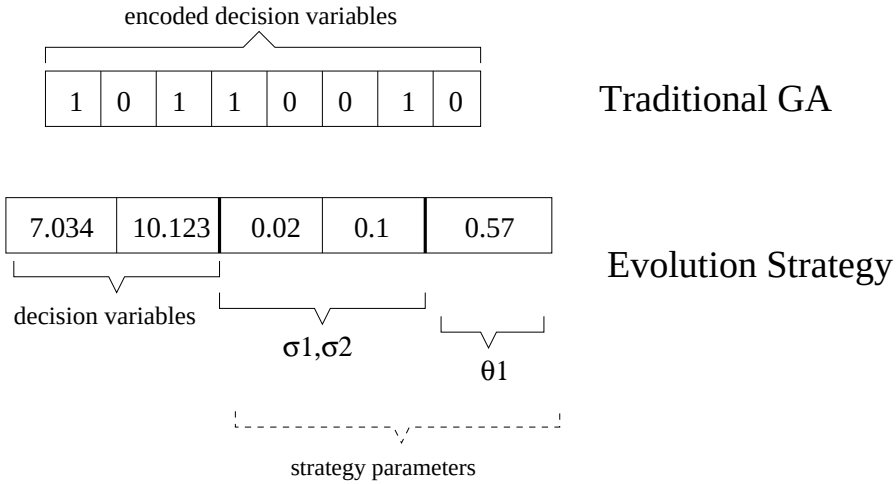


Figure 6.1: Representation of individuals of a genetic algorithm and an evolution strategy. Note that we do not use correlated mutation (we do not include rotation angles θ).

Recombination can be sexual (two parents) or panmictic (more than two parents). It is worth reminding that recombination can be applied to the decision variables of the problem as well as to the strategy parameters. There are two main types of recombination: (1) Discrete and (2) Intermediate. Both can be either sexual or panmictic. Also, Schwefel [170] proposed to generalize intermediate recombination by allowing arbitrary weight factors from the interval $[0, 1]$ to be used anew for each component of the chromosome. For a complete description of the recombination operator we provide the following list, complimented with Figure 6.2 which presents a 2-dimensional schema for different recombination mechanisms [6]:

$$\text{offspring}_i = \begin{cases} \textbf{Operation} & \textbf{Type of Recombination} \\ P1_i \text{ or } P2_i & \text{discrete} \\ P1_i \text{ or } PJ_i & \text{panmictic discrete} \\ P1_i + ((P2_i - P1_i)/2) & \text{intermediate} \\ P1_i + ((PJ_i - P1_i)/2) & \text{panmictic intermediate} \\ P1_i + \chi((P2_i - P1_i)/2) & \text{generalized intermediate} \\ P1_i + \chi_i((PJ_i - P1_i)/2) & \text{panmictic generalized intermediate} \end{cases}$$

where $P1$ and $P2$ are the parents for the sexual recombination, PJ means a different parent for each gene (variable of the problem) in the chromosome. χ_i is the weight factor created anew for each decision variable and used in the generalized recombination.

Based on Figure 6.2 where $P1$ and $P2$ are the parents, only the corners of the rectangle

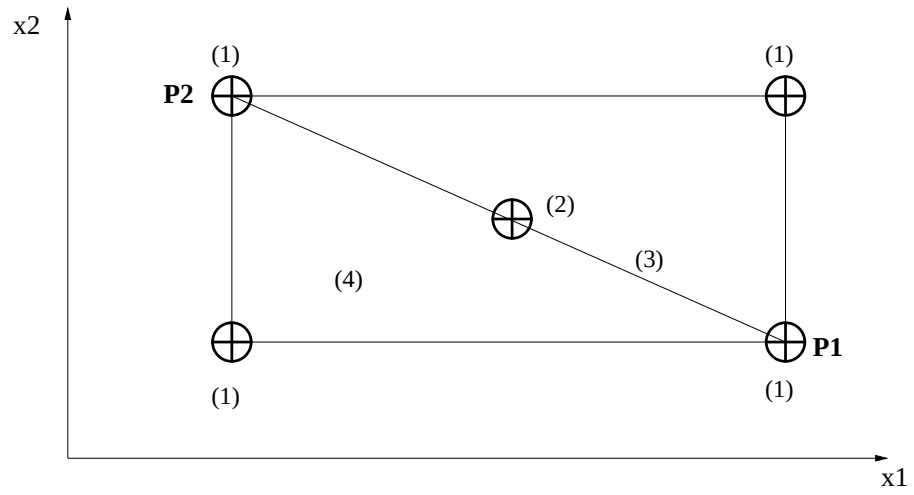


Figure 6.2: 2-dimensional scheme for different recombination mechanisms

(indicated by (1) in the figure) can be reached by discrete recombination. Intermediate recombination yields the center of the rectangle's diagonal lines (indicated by (2) in the figure). Generalized intermediate recombination allows for a result located anywhere on the diagonal line (indicated by (3) in the figure) connecting $P1$ and $P2$. Finally, panmictic generalized intermediate recombination allows for the creation of an arbitrary point located somewhere within the rectangle (indicated by (4) in the figure).

The mutation operator works on the decision variables and also on the strategy parameters. The mutation is calculated in the following way:

$$\sigma_i' = \sigma_i \cdot exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \tag{6.1}$$

$$\theta_j' = \theta_j + \beta \cdot N_j(0,1) \tag{6.2}$$

$$\vec{x}' = \vec{x} + \vec{N}(\vec{0}, C(\vec{\sigma'}, \vec{\theta'})) \tag{6.3}$$

where τ and τ' are interpreted as "learning rates" and are defined by Schwefel [6] as: $\tau = (\sqrt{2\sqrt{n}})^{-1}$ and $\tau' = (\sqrt{2n})^{-1}$ and $\beta \approx 0.0873$. $N_i(x,y)$ is a function that returns a real normal-distributed random number with mean x and standard deviation y. The index i indicates that this random number is generated anew for each decision variable (gene of the chromosome).

$C(\vec{\sigma'}, \vec{\theta'})$ is the covariance matrix represented by the set of n stepsizes and the $n(n-1)/2$ rotation angles. The mutation on Equation 6.3 is implemented as follows: To calculate this $\vec{N}(\vec{0}, C(\vec{\sigma'}, \vec{\theta'}))$, which represents the vector of stepsizes but now updated using

correlated mutation (we call this vector $\vec{\sigma''}$) we perform the following: For each angle θ'_k, we calculate its corresponding two stepsize values in its corresponding axes σ'_i and σ'_j and we calculate the following: $\sigma''_i = \sigma'_i \cdot \cos\theta_k - \sigma'_j \cdot \sin\theta_k$ and $\sigma''_j = \sigma'_i \cdot \sin\theta_k + \sigma'_j \cdot \cos\theta_k$ [170]. In this way, the $\vec{\sigma''}$ values are now mutated in a correlated way and can be used to mutate the $\vec{x}$ vector of decision variables.

Some authors use correlated mutation, but it implies an extra computational effort to process the value of each angle and also to rotate the individual. Moreover, some extra memory space is needed to store all the different angles per individual (the angles are formed by the combination of all the axis based on the number of decision variables of the problem). If non-correlated mutation is preferred, the computational cost and the storage space for each individual get lower.

If a non-correlated mutation is used, the mutation expressions are:

$$\sigma'_i = \sigma_i \cdot exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \tag{6.4}$$

$$x'_i = x_i + \sigma'_i \cdot N_i(0,1) \tag{6.5}$$

The detailed ES algorithm is shown in Figure 6.3.

6.2 Motivation

Evolution Strategies (ES) have been widely used to solve global optimization problems [172, 70, 69]. ES have been found not only efficient in solving a wide variety of optimization problems, but also have a strong theoretical background, [170, 6, 17] mainly focused in two aspects: global convergence and convergence velocity. The first one concentrates of proving that the algorithm is capable of finding the global optimum, and the last one refers to the speed of the algorithm to approach a local optimum. We decided to use an approach in which the objective function and the constraints are handled separately but instead of adopting a multiobjective approach, we will use comparisons based on feasibility rules and simple diversity mechanisms. The main aim is to propose a competitive evolutionary approach to solve constrained problems.

The hypothesis that originated this work is the following: (1) The self-adaptation mechanism of an ES helps to sample the search space well enough as to reach the feasible region reasonably fast and (2) the addition of simple feasibility rules to an ES should be enough to guide the search in such a way that the global optimum can be approached efficiently. Also, self-adaptation or some form of online-adaptation must be used to adapt extra parameters if they can not be fixed. Also, the approach must solve in a competitive way all the functions of the well known benchmark used in the literature [162].

```
Begin
    t=0
    Create μ random solutions for the initial population.
    Evaluate all μ individuals
    Assign a fitness value to all μ individuals
    For t=1 to MAX_GENERATIONS Do
        Produce λ offspring by recombination of the μ parents
        Mutate each child
        Evaluate all λ offspring
        Assign a fitness value to all λ individuals
        If Selection = "+" Then
            Select the best μ individuals from the μ + λ individuals
        Else
            Select the best μ individuals from the λ individuals
        End If
    End For
End
```

Figure 6.3: Detailed ES algorithm

6.3 The First Version: a $(\mu + 1)$-ES

Motivated by the fact that some of the most recent and competitive approaches to incorporate constraints into an EA use an ES (see for example [162, 77]), we conducted an experimental study in which we evaluated five types of ES [122]:

1. A variation of a $(\mu + 1)$-ES.
2. A $(\mu + \lambda)$-ES with correlated mutation.
3. A $(\mu + \lambda)$-ES without correlated mutation.
4. A (μ, λ)-ES with correlated mutation.
5. A (μ, λ)-ES without correlated mutation.

We decided to add just a comparison mechanism based on three simple criteria to the ES to deal with constrained search spaces. The three simple selection criteria used are the following:

1. Between 2 feasible solutions, the one with the highest fitness value wins (assuming a maximization problem/task).

2. If one solution is feasible and the other one is infeasible, the feasible solution wins.

3. If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred.

For our comparative study, we set a total of $350,000$ fitness function evaluations. We performed 30 runs for each problem and for each type of ES. Equality constraints were transformed into inequalities using a tolerance value of 0.0001. For the $(\mu + 1)$-ES, the initial values are: $\sigma = 4.0$, $C = 0.99$, $\mu = 5$, and the maximum number of generations was set to $350,000$. For the $(\mu + \lambda)$-ES and (μ, λ)-ES, we adopted panmictic discrete recombination both for the strategy parameters and for the decision variables. The learning rates values were calculated as indicated in [6]. The initial values for the stepsize were 3.0 for all the decision variables. The initial values for the remaining ES are: $\mu = 100$, $\lambda = 300$, and maximum number of generations = 1166. This unusual number of generations was chosen in order to adjust the total number of objective function evaluations to $350,000$. Based on the classification of problems proposed in Section 5.4.2, we decided to perform this experiment using just the first 13 test problems. The results for the $(\mu + 1)$-ES are shown in Table 6.1, results for the $(\mu + \lambda)$-ES with correlated mutation are listed in Table 6.2, results for the $(\mu + \lambda)$-ES without correlated mutation appear in Table 6.3, results for the (μ, λ)-ES with correlated mutation are in Table 6.4 and finally, results for the (μ, λ)-ES without correlated mutation are shown in Table 6.5.

Based on the results obtained, the most competitive ES was the $(\mu + 1)$-ES in which there is just one current individual. Then, μ mutations from this only solution are created, producing μ new individuals. However, these new individuals are not evaluated. Their only role is to be combined into one offspring, using panmictic discrete recombination, but not fixing one parent. Instead, for each decision variable we allow all individuals to be selected to inherit their value (they can be chosen more than once). This new offspring is evaluated and it will compete against the current individual. The best solution (between the current solution and the new offspring) is selected as the new current solution. This approach is based on the two mechanisms previously indicated.

The details of the $(\mu + 1)$-ES algorithm are shown in Figure 6.4.

However, the $(\mu+1)$-ES approach has a tendency to get trapped either in local optimum solutions or in the infeasible region [122]. This is due to the high selection pressure generated by the comparison mechanism, which gives infeasible solutions a zero probability of remaining for the next generation, regardless of their closeness to the feasible region.

	$(\mu+1)$-**ES**					
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	-15.000	$\mathbf{-15.000}$	-14.847	-14.998	-12.999	4.1E-1
g02	0.803619	0.793083	0.698932	0.708804	0.576079	6.2E-2
g03	1.000	**1.000**	**1.000**	**1.000**	**1.000**	1.4E-5
g04	-30665.539	$\mathbf{-30665.539}$	-30665.442	$\mathbf{-30665.539}$	-30663.496	3.9E-1
g05	5126.498	–	–	–	–	–
g06	-6961.814	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	0
g07	24.306	24.368	24.703	24.731	25.517	2.4E-1
g08	0.095825	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0
g09	680.63	680.632	680.674	680.659	680.915	5.2E-2
g10	7049.25	–	–	–	–	–
g11	0.75	**0.75**	0.78	0.77	0.88	3.73E-2
g12	1.000	**1.000**	**1.000**	**1.000**	**1.000**	0
g13	0.053950	–	–	–	–	–

Table 6.1: Statistical results obtained with the $(\mu+1)$-ES [122] in the 13 test functions. "-" means no feasible solutions were found. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

	Correlated $(\mu+\lambda)$-**ES**					
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	-15.000	-14.999	-14.998	-14.999	-14.973	4.62E-3
g02	0.803619	0.803594	0.796618	0.792588	0.785246	5.86E-3
g03	1.000	0.472	0.202	0.185	0.086	1.00E-1
g04	-30665.539	-30665.529	-30665.520	-30665.520	-30665.508	5.17E-3
g05	5126.498	–	–	–	–	–
g06	-6961.814	-6961.761	-6960.628	-6960.972	-6957.259	1.15E+0
g07	24.306	24.330	24.422	24.413	24.563	6.52E-2
g08	0.095825	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0
g09	680.63	680.633	680.638	680.638	680.645	2.70E-3
g10	7049.25	7294.707	10857.808	9929.193	20743.082	3335.12E+0
g11	0.75	**0.75**	0.752	**0.75**	0.81	1.13E-2
g12	1.000	**1.000**	**1.000**	**1.000**	**1.000**	0
g13	0.053950	0.999998	1.000	1.000	1.000	0

Table 6.2: Results obtained with the correlated $(\mu+\lambda)$-ES in the 13 test problems. "-" means no feasible solutions were found. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

	Non-correlated $(\mu + \lambda)$-ES					
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	-15.000	-14.986	-14.974	-14.974	-14.954	7.79E-3
g02	0.803619	0.803607	0.800743	0.803503	0.792375	4.64E-3
g03	1.000	0.474	0.238	0.243	0.027	1.14E-1
g04	-30665.539	-30664.838	-30651.001	-30653.475	-30619.619	13.16E+0
g05	5126.498	–	–	–	–	–
g06	-6961.814	$\mathbf{-6961.814}$	-6938.453	-6961.811	-6567.754	83.16E+0
g07	24.306	24.329	24.391	24.393	24.478	4.67E-2
g08	0.095825	**0.095825**	0.095823	**0.095825**	0.095771	1.0E-5
g09	680.63	680.631	680.640	680.636	680.666	1.04E-2
g10	7049.25	7075.010	7802.033	7531.349	10083.972	762.99E+0
g11	0.75	0.751	0.88	0.90	0.99	8.53E-2
g12	1.000	**1.000**	**1.000**	**1.000**	0.999	1.0E-6
g13	0.053950	–	–	–	–	–

Table 6.3: Results obtained with the non-correlated $(\mu + \lambda)$-ES in the 13 test problems. "-" means no feasible solutions were found. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

	Correlated (μ, λ)-ES					
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	-15.000	-14.931	-14.915	-14.915	-14.889	9.78E-4
g02	0.803619	0.797201	0.777913	0.784871	0.748130	1.25E-2
g03	1.000	0.445	0.108	0.041	*0.000	1.40E-1
g04	-30665.539	-30664.217	-30662.855	-30662.591	-30661.170	7.72E-1
g05	5126.498	–	–	–	–	–
g06	-6961.814	-6802.235	-6538.026	-6541.951	-6277.651	127.24E+0
g07	24.306	24.651	24.887	24.915	25.238	1.42E-1
g08	0.095825	**0.095825**	0.095822	0.095823	0.095811	4.0E-6
g09	680.63	680.775	681.139	681.136	681.498	1.43E-1
g10	7049.25	12146.522	17457.792	18413.144	29076.020	4163.69E+0
g11	0.75	0.88	0.95	0.96	0.99	2.80E-2
g12	1.000	**1.000**	**1.000**	**1.000**	**1.000**	0
g13	0.053950	–	–	–	–	–

Table 6.4: Results obtained with the correlated (μ, λ)-ES in the 13 test problems. "-" means no feasible solutions were found. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

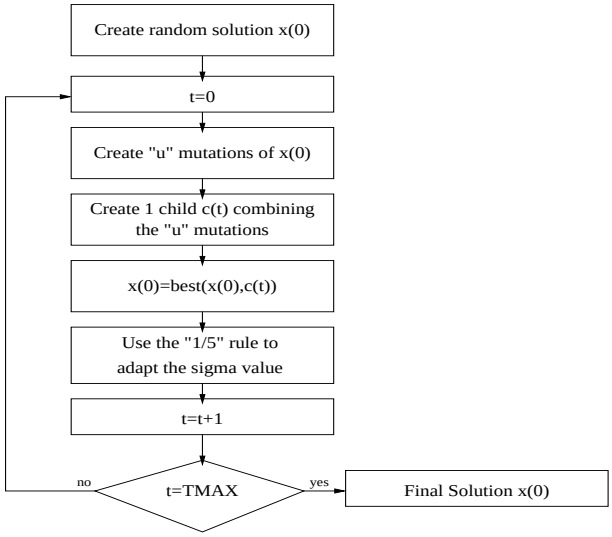


Figure 6.4: Diagram of our version of the $(\mu + 1)$-ES algorithm

6.4 The Second Version: a $(1 + \lambda)$-ES with a Diversity Mechanism

In order to improve the quality and robustness of the results, a diversity mechanism was added to our $(1 + \lambda)$-ES as reported in [124]. In this case, a $(1 + \lambda)$-ES was adopted and the diversity mechanism consisted on allowing solutions with a good value of the objective function to remain as a new starting point for the search at each generation, regardless of feasibility. Additionally, we introduced a self-adaptive parameter called Selection Ratio (S_r), which refers to the percentage of selections that will be performed in a deterministic way (as used in the $(\mu + 1)$-ES [122], where the child replaces the current solution based on the comparison mechanism previously stated). In the remaining $1 - S_r$ selections, there were two choices: (1) either the parent (out of the λ) with the best value of the objective function would replace the current solution (regardless of its feasibility) or (2) the best parent (based on the comparison mechanism) would replace the current solution (see Figure 6.6). Both options are given a 50% probability each. See Figure 6.5.

The S_r parameter is adapted online using the fitness value of the current solution during an interval of time (number of generations). The "mean deviation" (M_d) of the current solution over a certain number of generations is calculated in order to know how much it has changed. All the fitness values are normalized in order to obtain a value between 0 and 1. The expression used to adapt the S_r value is the following:

	Non-correlated (μ, λ)-ES					
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	-15.000	-14.995	-14.971	-14.975	-14.931	1.56E-2
g02	0.803619	0.792393	0.779795	0.784977	0.753796	1.20E-2
g03	1.000	0.465	0.165	0.155	0.007	1.34E-1
g04	-30665.539	-30432.131	-30309.273	-30297.313	-30204.131	52.56E+0
g05	5126.498	–	–	–	–	–
g06	-6961.814	-6916.590	-6711.116	-6789.254	-6068.743	206.01E+0
g07	24.306	24.484	24.929	25.016	25.485	2.71E-1
g08	0.095825	**0.095825**	**0.095825**	**0.095825**	0.095821	1.0E-6
g09	680.63	680.809	681.351	681.324	682.871	4.85E-1
g10	7049.25	8024.88	11721.52	11677.32	16982.54	2319.20E+0
g11	0.75	–	–	–	–	–
g12	1.000	**1.000**	**1.000**	**1.000**	**1.000**	0
g13	0.053950	–	–	–	–	–

Table 6.5: Results obtained with the non-correlated (μ, λ)-ES in the 13 test problems. "-" means no feasible solutions were found. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

$$S_r(t) = \begin{cases} S_r(t - \text{interval})/1.001 & \text{if } M_d < 0.1 \\ S_r(t - \text{interval}) * 1.001 & \text{if } M_d > 0.2 \\ S_r(t - \text{interval}) & \text{if } 0.1 \leq M_d \leq 0.2 \end{cases} \tag{6.6}$$

where *interval* is defined as a percentage of the maximum number of generations. For example if the interval is defined as 0.05 and the number of generations is 100, the update process will take place at every 5 generations. As it can be seen, S_r will be decreased if the current solution has not significantly changed during the given interval (i.e. $M_d < 0.1$) allowing a candidate solution (which may be infeasible) with a good fitness value to replace the current solution. This is meant to increase diversity. On the other hand, S_r will increase if the solution has been significantly different (i.e. $M_d > 0.2$) during the interval, thus favoring deterministic selection to impel convergence. S_r will keep its current value if the variation of the current solution in the interval has been moderated (i.e. $0.1 \leq M_d \leq 0.2$).

In order to always keep the best solution found during the process, a super elitist mechanism was included. This super elitist mechanism is required because the diversity mechanism adopted causes that the new current solution is not necessarily feasible. The implementation of this super elitist mechanism does not add any significant extra computational or storage cost to the algorithm.

To deal with equality constraints, a parameterless dynamic mechanism similar to that used in ASCHEA [77] was adopted. The tolerance value ϵ is decreased with respect to the current generation using the following expression:

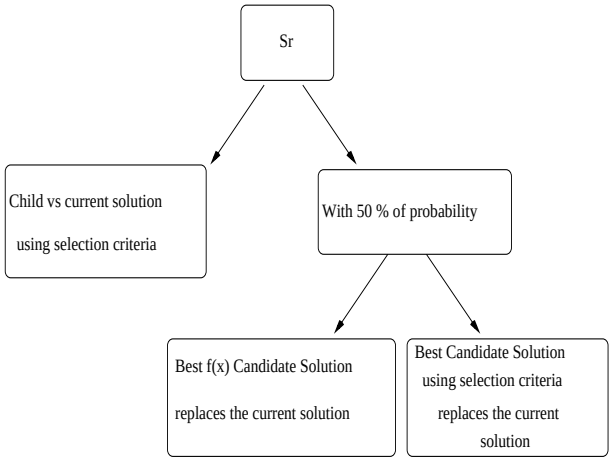


Figure 6.5: Diagram of diversity mechanism implemented in the the $(1+\lambda)$-ES algorithm.

$$\epsilon_j(t+1) = \epsilon_j(t)/1.000001 \tag{6.7}$$

The value 1.000001 was obtained empirically in order to allow an adequate decrease of the tolerance for equality constraints.

The details of the $(1+\lambda)$-ES algorithm are shown in Figure 6.7.

The parameters used in the experiments are the following (30 runs were performed for each problem): the total number of fitness function evaluations was set to 330,000. Equality constraints were transformed into inequalities using an initial tolerance value of 0.001. The initial values for the $(1+\lambda)$-ES parameters were: $\sigma = 4.0$, $C = 0.99$, $\lambda = 3$, and maximum number of generations = $275,000$. The interval of the S_r updates was almost negligible (0.9). This means that the update will take place until generation $247,500$. We anticipated that our approach would not be too sensitive to the S_r parameter and our experiments confirmed this hypothesis. The statistical results are presented in Table 6.6. For problems g03, g04, g08 and g11 the results seem to improve the corresponding optimum. This is due to the fact that either we use a small tolerance for equality constraints (g03 and g11), or we round off the results (g04 and g08). The results improved in quality and robustness with respect to the previous version of the algorithm, but for one test problem (g05) no feasible solutions could be found and for other functions the statistical results indicated a lack of robustness.

In Table 6.7 the $(1+\lambda)$-ES [124] is compared against the variation of a $(\mu+1)$-ES [122].

The $(1+\lambda)$-ES was able to converge to the global optimum in 7 of the test 13 functions

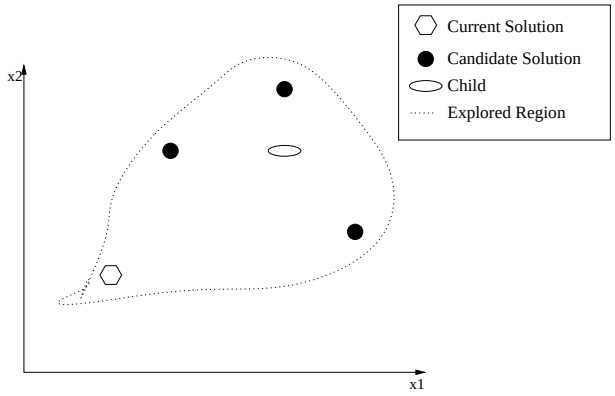


Figure 6.6: Diagram that illustrates the explored region of the search space in the $(1+\lambda)$-ES version. In the variation of a $(\mu+1)$-ES only the points in white (child and current solution) can be selected.

	$(1+\lambda)$-**ES**					
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	-15.000	$\mathbf{-15.000}$	$\mathbf{-15.000}$	$\mathbf{-15.000}$	$\mathbf{-15.000}$	0
g02	0.803619	0.803569	0.769612	0.782466	0.702322	2.75E-2
g03	1.000	**1.004**	**1.003**	**1.003**	**1.002**	4.23E-4
g04	-30665.539	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	0
g05	5126.498	$-$	$-$	$-$	$-$	$-$
g06	-6961.814	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	0
g07	24.306	24.314	24.419	24.419	24.561	7.12E-2
g08	0.095825	**0.095825**	0.095784	**0.095825**	0.095473	1.04E-4
g09	680.63	680.669	680.810	680.798	681.200	1.23E-1
g10	7049.25	7057.04	10771.42	10935.45	16375.27	2524.08E+0
g11	0.75	**0.75**	**0.75**	**0.75**	0.751	3.16E-4
g12	1.000	**1.000**	**1.000**	**1.000**	**1.000**	0
g13	0.053950	0.053964	0.264135	0.438692	0.544346	2.06E-1

Table 6.6: Statistical results obtained with the $(1+\lambda)$-ES [124] in the 13 test functions. "**-**" means no feasible solutions were found. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

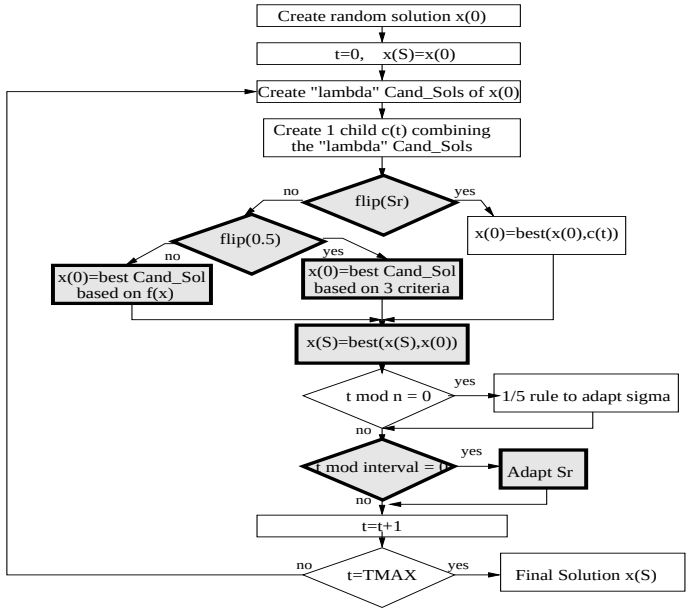


Figure 6.7: Diagram of the $(1 + \lambda)$-ES algorithm. The shaded boxes indicate the steps added to the first version of the algorithm

used (g01, g03, g04, g06, g08, g11 and g12), and it was able to converge very close to the optimum in g02, g07, g09, g10 and g13.

With respect to the $(\mu + 1)$-ES (Table 6.7), the $(1 + \lambda)$-ES improved the robustness of the results in problems g01, g02, g04, g07 and g11. Also, the quality of the results was improved in problems g10 and g13.

Measuring the computational cost, the number of fitness function evaluations (FFE) is as follows: The $(1 + \lambda)$-ES performed $330,000$ FFE and the $(\mu + 1)$-ES required $350,000$.

The results were improved with respect to the $(\mu + 1)$-ES, but for one test problems (g05) no feasible solutions could be found and for other functions the statistical results indicated lack of robustness.

The $(1 + \lambda)$-ES was also used to solve some engineering design problems [125] making emphasis on the low computational cost of the approach, measured by the number of objective function evaluations performed. The number of evaluations of the objective function was fixed to $36,000$. The results compared against penalty functions approaches are presented in Tables: 6.8 for the Pressure Vessel Design, in Table 6.9 for the Welded Beam design, in Table 6.10 for the Tension/Compression Spring design and, finally, in Table 6.11 for the Speed Reducer Design Problem.

Those results evidence the lack of consistency of the penalty-function-based approaches

		Best Result		Mean Result		Worst Result	
P	**Optimal**	$(1+\lambda)$-**ES**	$(\mu+1)$-**ES**	$(1+\lambda)$-**ES**	$(\mu+1)$-**ES**	$(1+\lambda)$-**ES**	$(\mu+1)$-**ES**
g01	-15.000	$\mathbf{-15.000}$	$\mathbf{-15.000}$	$\mathbf{-15.000}$	-14.847	$\mathbf{-15.000}$	-12.999
g02	0.803619	**0.803569**	0.793083	**0.769612**	0.698932	**0.702322**	0.576079
g03	1.000	**1.004**	**1.000**	**1.003**	**1.000**	**1.002**	**1.000**
g04	-30665.539	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	-30663.496
g05	5215.498	–	–	–	–	–	–
g06	-6961.814	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$
g07	24.306	**24.314**	24.368	**24.419**	24.703	**24.561**	25.517
g08	0.095825	**0.095825**	**0.095825**	**0.095784**	**0.095825**	0.095473	**0.095825**
g09	680.63	680.669	**680.632**	680.810	**680.674**	681.200	**680.915**
g10	7049.25	**7057.04**	–	**10771.42**	–	**16375.27**	–
g11	0.75	**0.75**	**0.75**	**0.75**	0.79	**0.751**	0.88
g12	1.000	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
g13	0.053950	**0.053964**	–	**0.264135**	–	**0.544346**	–

Table 6.7: Comparison of results between the $(1+\lambda)$-ES [124] and the $(\mu+1)$-ES [122]. "-" means no feasible solutions were found. A result in **boldface** indicates a better results (or best known or optimal solution found) for the corresponding approach.

Pressure Vessel	**Statistical Comparison**				
Design	**Death Penalty**	**Static**	**Dynamic**	**Adaptive**	$(1+\lambda)$-**ES**
Best	6129.827637	52.177204∗	6104.700195	88.063927∗	**6059.714355**
Mean	7191.641992	53.757520∗	6670.045085	3335.592351	**6355.343115**
Median	7239.383545	52.870346∗	6688.087646	604.228973∗	**6392.557129**
Worst	8876.304688	67.266701∗	7788.871094	11983.214844	**6846.628418**
St. Dev	636.043280	2.897835	397.492272	4172.602199	**256.043795**

Table 6.8: Comparison of results obtained with the $(1+\lambda)$-ES [125] for the pressure vessel design problem. "*" means infeasible. A value in **boldface** indicates a better result for the corresponding approach.

Welded Beam	**Statistical Comparison**				
Design	**Death Penalty**	**Static**	**Dynamic**	**Adaptive**	$(1+\lambda)$-**ES**
Best	1.748899	**1.742388**	1.752588	1.657890∗	1.748594
Mean	2.032251	1.910210	2.067771	2.108092	**1.870860**
Median	1.998613	1.876692	1.982786	2.016899	**1.852371**
Worst	2.973882	2.252468	2.855598	3.351592∗	**2.232832**
St. Dev	0.263312	0.135383	0.277771	0.361124	**0.106377**

Table 6.9: Comparison of results obtained with the $(1+\lambda)$-ES [125] for the welded beam design problem. "*" means infeasible. A value in **boldface** indicates a better result for the corresponding approach.

Ten./Comp. Spring Design	Statistical Comparison				
	Death Penalty	**Static**	**Dynamic**	**Adaptive**	$(1+\lambda)$**-ES**
Best	0.012732	0.012729	0.012689	0.012729	**0.012688**
Mean	0.014527	0.013774	0.013681	0.013675	**0.013014**
Median	0.014141	0.013621	0.013436	0.013606	**0.012756**
Worst	0.017723	0.016407	0.016597	**0.015933**	0.017037
St. Dev	0.001457	0.001045	0.000934	**0.000720**	0.000801

Table 6.10: Comparison of results obtained with the $(1 + \lambda)$-ES [125] for the tension/compression spring design problem. A value in **boldface** indicates a better result for the corresponding approach.

Speed Reducer Design	Statistical Comparison				
	Death Penalty	**Static**	**Dynamic**	**Adaptive**	$(1+\lambda)$**-ES**
Best	1399.237427∗	1671.386475∗	1265.959229∗	1679.101318∗	**3025.005127**
Mean	1399.237427∗	267997861.844320	20032.261825	12174.159896	**3088.777816**
Median	524036.802128∗	3661.611328∗	3179.876343∗	3272.115234	**3078.591797**
Worst	2913.118042∗	8024214016.0∗	243416.062500∗	244393.671875∗	**3226.248291**
St. Dev	15600199.00	1440295898.14	59873.421388	43472.043832	**47.361890**

Table 6.11: Comparison of results obtained with the $(1+\lambda)$-ES [125] for the speed reducer design problem. "*" means infeasible. A value in **boldface** indicates a better result for the corresponding approach.

	Best Result		Mean Result		Worst Result	
Problem	$(1+\lambda)$-**ES**	**SB**	$(1+\lambda)$-**ES**	**SB**	$(1+\lambda)$-**ES**	**SB**
Press.Vessel	**6059.728027**	6171.00	**6295.362964**	6335.05	6887.999023	**6453.65**
Welded Beam	**1.742510**	2.4426	**1.876213**	2.5215	**2.023696**	2.6315
Speed Red.	3038.684082	**3008.08**	3088.530505	**3012.12**	3199.294922	**3028.28**

Table 6.12: Comparison of results between the $(1 + \lambda)$-ES [125] and the Socio Behavioral approach (SB) [3]. A value in **boldface** indicates a better result for the corresponding approach.

(based on the difficult to define their required parameters) [125]. On the other hand, the $(1+\lambda)$-ES obtained the lowest standard deviations so far. Also, the quality of results of our approach, generally speaking, was also better. Finally, the $(1+\lambda)$-ES was compared against one state-of-the-art engineering design algorithm, called the "Socio-Behavioral approach" [3] showing a very competitive approach. The Socio-Behavioral approach uses a particle-swarm like algorithm [101] to solve engineering design problems. The idea is to simulate the behavior of a civilization formed by societies, where each society has a leader which guides the remaining individuals in the society to promising areas of the search space. Also, there is a society of leaders which is guided by the best solution of all the civilization. See Table 6.12 for a comparison of results.

6.5 The Final Version: a $(\mu + \lambda)$-ES with an Improved Diversity Mechanism

The two previous versions of the algorithm [122, 124] are based on a single-membered ES and they both lack of explorative power to sample large search spaces. Thus, we decided to re-evaluate the use of a $(\mu + \lambda)$-ES to overcome this limitation, but in this case, improving the diversity mechanism implemented in the second version of our approach [123] and eliminating the use of the self-adaptive S_r parameter. The new version of the algorithm, called *"Simple Multimembered Evolution Strategy (SMES)"* is based on the same concepts of its predecessors as discussed before.

The detailed features of our algorithm are the following:

- *Diversity mechanism*: With an idea similar to that used in the $(1 + \lambda)$-ES version, we allow infeasible solutions to remain in the population. However, unlike this previous approach, where the best parent based only on the objective function (regardless of its feasibility) can survive, in this new approach we allow the infeasible individual with the best value of the objective function and with the lowest amount of constraint

violation to survive for the next generation. This solution (called by us the best infeasible solution) can be chosen either from the parents or the offspring population, with 50% probability. The process of allowing that solution to survive for the next generation happens 3 times every 100 during the same generation. However, it is a desired behavior because a few copies of that solution will allow its recombination with several solutions in the population, especially with feasible ones. Recombining feasible solutions with infeasible solutions in promising areas (based on the good value of the objective function) and close to the boundary of the feasible region will allow the ES to reach global optimum solutions located precisely on the boundary of the feasible region of the search space (which are normally the most difficult solutions to reach). Following the idea of allowing just a few infeasible solutions (one in the case of the $(1 + \lambda)$-ES approach), we allow the best infeasible solution to be copied into the population for the next generation just 3 times for every 100 attempts. This works in the following way: When the deterministic replacement is used to form the population for the next generation in an ES, the best individuals from the union of parents and offspring are selected based on the comparison mechanism previously indicated (in a deterministic way). The process will pick feasible solutions with a better value of the objective function first, followed by infeasible solutions with a lower value of constraint violation. However, 3 times from every 100 picks, the best infeasible solution (from either the parents or the offspring population with 50% probability each) is copied in the population for the next generation. The pseudocode is listed in Figure 6.8.

Based on the empirical evidence observed in the previous version of the approach [124] where we used a population of 3 offspring, we decided to use a small number of copies of the best infeasible solutions for the next generation of our approach. For values larger than 3, the quality and robustness of our approach tend to decrease. It is worth remarking that in the case where no infeasible solutions are found in the population, a random solution is copied to the population for the next generation. Therefore, it is possible, at any given generation, to have an entirely feasible parents population. However, the mechanism will allow, when the offspring are generated, to have infeasible individuals again.

- *Combined recombination*: We use panmictic recombination, but with a combination of the discrete and intermediate recombination operators. Each gene in the chromosome can be processed with any of these two recombination operators with 50% probability. This operator is applied to both, strategy parameters (sigma values) and decision variables of the problem. The pseudocode is shown in Figure 6.9. Note that we use intermediate recombination by just computing the average between the values

```
function population_for_next_generation()
    For i=1 to μ Do
        If flip(0.97)
            Select the best individual based on the comparison mechanism
            from the union of the parents and offspring population,
            add it to the population for the next generation and delete
            it from this union.
        Else
            If flip(0.5)
                Select the best infeasible individual from the parents
                population and add it to the population for the next
                generation.
            Else
                Select the best infeasible individual from the offspring
                population and add it to the population for the next
                generation.
            End If
        End If
    End For
End
```

Figure 6.8: Pseudocode of the generation of the population for the next generation with the diversity mechanism incorporated. flip(P) is a function that returns TRUE with probability P.

of the variable of each parent (as originally proposed by Schwefel [170]).

- *Reduction of the initial stepsize of the ES*: The previous versions of our algorithm are based on a variation of a $(\mu + 1)$-ES [122] and a $(1 + \lambda)$-ES [124]. These approaches do not use a population of solutions and employ the most simple scheme of an ES where only one sigma value is used for all the decision variables. We observed that when that sigma value was close to zero, the previous approaches were capable of reaching the global optimum, or at least improve the value of the final solution. Therefore, in our new approach based on a multimembered ES, we decided to favor finer movements in the search space. We experimented with just a percentage of the quantity obtained by the formula proposed by Schwefel [170]. We initialize the sigma values (we use one for each decision variable) for each individual in the initial population with only a 40% of the value obtained by the following formula (where n is the number of decision variables):

```
function combined_recombination()
    Select mate_1 from the parents population
    For i=1 to NUMBER_OF_VARIABLES Do
        Select mate_2 from the parents population
        If flip(0.5)
            If flip(0.5)
                child_i = mate_1_i
            Else
                child_i = mate_2_i
            End If
        Else
                child_i = mate_1_i + ((mate_2_i − mate_1_i/2, 0)
        End If
    End For
End
```

Figure 6.9: Pseudocode of the panmictic combined (discrete-intermediate) recombination operator used by our approach. flip(P) is a function that returns TRUE with probability P.

$$\sigma_i(0) = 0.4 \times \left(\frac{\Delta x_i}{\sqrt{n}} \right) \tag{6.8}$$

where Δx_i is approximated with the expression (suggested in [162]), $\Delta x_i \approx x_i^u - x_i^l$, where $x_i^u - x_i^l$ are the upper and lower bounds of the decision variable i.

Summarizing, our approach works over a simple multimembered evolution strategy: $(\mu + \lambda)$-ES. The only modifications introduced are the reduction of the initial stepsize of the sigma values, the panmictic combined (discrete-intermediate) recombination and the changes to the original deterministic replacement of the ES (made by sorting the solutions with respect to the comparison mechanism based on feasibility discussed at the beginning of this section), allowing the best infeasible solution, from either the parents or the offspring population, to remain in the next generation. The details of our approach are presented in Figure 6.10.

A graphical example of the expected behavior of the approach can be found in Figure 6.11. We used a 2-dimensional test problem g08, which is a problem easy to solve by the approach; it requires about 5400 evaluations of the objective function (18 generations) to reach the global optimum, but it helps to visualize how our approach works. The definition of this problem can be found in Appendix A.

	Statistical Results of the Simple Multimembered Evolution Strategy (SMES)					
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	-15.000	$\mathbf{-15.000}$	$\mathbf{-15.000}$	$\mathbf{-15.000}$	$\mathbf{-15.000}$	0
g02	0.803619	0.803601	0.785238	0.792549	0.751322	1.67E-2
g03	1.000	**1.000**	**1.000**	**1.000**	**1.000**	2.09E-4
g04	-30665.539	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	$\mathbf{-30665.539}$	0
g05	5126.498	5126.599	5174.492	5160.198	5304.167	50.06E+0
g06	-6961.814	$\mathbf{-6961.814}$	-6961.284	$\mathbf{-6961.814}$	-6952.482	1.85E+0
g07	24.306	24.327	24.475	24.426	24.843	1.32E-1
g08	0.095825	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0
g09	680.63	680.632	680.643	680.642	680.719	1.55E-2
g10	7049.25	7051.903	7253.047	7253.603	7638.366	136.02E+0
g11	0.75	**0.75**	**0.75**	**0.75**	**0.75**	1.52E-4
g12	1.000	**1.000**	**1.000**	**1.000**	**1.000**	0
g13	0.053950	0.053986	0.166385	0.061873	0.468294	1.77E-1

Table 6.13: Statistical results obtained by our SMES for the 13 test functions over 30 independent runs. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

As it can be observed, in Generation 1 there are a few feasible as well as several infeasible solutions. The behavior of the approach can be observed in generation 3, where there are more feasible solutions than those in generation 1 and also there are infeasible solutions surrounding the feasible region. In this way, helped by the combined crossover and the finer mutation movements the feasible region is sampled well-enough as to find promising areas (three areas in the example). This is shown in generation 6, where there is still an infeasible solution in the population. It is worth noticing that this infeasible solution is close to the area where the global optimum is located; this can be seen in generation 10 where the infeasible solution has disappeared but the approach has found the vicinity of the constrained global optimum. Our algorithm has converged to the constrained global optimum in generation 18.

Unlike Deb's [49] technique, our approach does not use niches in order to maintain diversity in the population. This is because inside the replacement process used to produce the population for the next generation, we incorporate a mechanism that allows slightly infeasible solutions with a good objective function value to be considered better than feasible ones. This ES-like replacement makes also a difference with respect to Powell and Skolnick's approach [146], which uses proportional selection (with linear ranking) on a GA-based approach.

6.5.1 Experiments and results

To evaluate the performance of the proposed approach we used the first 13 test functions adopted in section 5.4 (which are the most difficult based on the results of our comparison study of Chapter 5). Their expressions are provided in Appendix A at the end of this document.

We performed 30 independent runs for each test function. The learning rates values were calculated using the formulas proposed by Schwefel [170] (where n is the number of decision variables of the problem):

$$\tau = \left(\sqrt{2\sqrt{n}}\right)^{-1} \tau' = \left(\sqrt{2n}\right)^{-1} \tag{6.9}$$

The initial values for the stepsize were calculated using equation (6.8).

In the experiments the following parameters were used:

- $\mu = 100$.
- $\lambda = 300$.
- Number of generations = 800.
- Number of objective function evaluations = $240,000$.

The combined recombination operator was used both for the decision variables of the problem and for the strategy parameters (sigma values). Note that we do not use correlated mutation [121].

To deal with equality constraints, a dynamic mechanism originally proposed in ASCHEA [77] and used in [124] is adopted. The tolerance value ϵ is decreased with respect to the current generation using the following expression:

$$\epsilon_j(t+1) = \epsilon_j(t)/1.00195 \tag{6.10}$$

The initial ϵ_0 was set to 0.001. Note that the use of the value 1.00195 in equation (6.10) causes the allowable tolerance for the equality constraints to go from 0.001 (initial value) to 0.0004 (final value) given the number of iterations adopted by our approach (if more iterations are performed, this value will tend to zero).

For problem g13, ϵ_0 was set to a much larger value (3.0), because in this case it is very difficult to generate feasible solutions during the initial generations of our approach. Thus, by using a large tolerance value, more individuals will be able to satisfy the equality constraints and will serve as reference solutions that the algorithm will improve over time.

Given that this larger value is adopted, we also changed the constant decreasing value. So, instead of using 1.00195, we adopt, in this case, a value of 1.0145. Such a value causes the allowable equality constraint violation to go from 3.0 (initial value) to 0.00003 (final value) given the number of iterations adopted by our approach. Note that the final allowable tolerance is smaller in this case, despite the initial larger value. As a matter of fact, we recommend to use this second setup for the tolerance of the equality constraints in problems in which no feasible solutions can be found by our algorithm when using a small initial ϵ_0.

Additionally, for problems $g03$ and $g13$ the initial stepsize required a more dramatic decrease. They were defined as 0.01 (just a 5% instead of the 40% used for the other test functions) for $g03$ and 0.05 (2.5%) for g13. Those two test functions seem to provide better results with very smooth movements. It is important to note that those two problems share the following features: moderately high dimensionality (five or more decision variables), nonlinear objective function, one or more equality constraints, and moderate size of the search space (based on the range of the decision variables). Those common features suggest that for these types of problems, finer movements provide a better sampling of the search space using an evolution strategy.

The statistical results of our SMES are summarized in Table 6.13.

We compare our approach against the results found in the literature for three state-of-the-art approaches: the Homomorphous Maps (HM) [110], Stochastic Ranking (SR) [162] and the Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA) [77]. The best results obtained by each approach are shown in Table 6.14. The mean values provided are compared in Table 6.15 and the worst results are presented in Table 6.16. The results provided by these approaches were taken from the original references for each method.

6.5.2 Discussion of results

As described in Table 6.13, our approach was able to find the global optimum in seven test functions (g01, g03, g04, g06, g08, g11 and g12) and it found solutions very close to the global optimum in the remaining six (g02, g05, g07, g09, g10, g13).

When compared with respect to the three state-of-the-art techniques previously indicated, we found the following (see Tables 6.14, 6.15 and 6.16):

Compared with the homomorphous maps (HM)

Our approach found a "better" best solution in ten problems (g01, g02, g03, g04, g05, g06, g07, g09, g10 and g12) and a "similar" best result in other two (g08 and g11). Also, our

P	Comparison of the best solution obtained.							
	Optimal	HM	SR	ASCHEA	SMES	GA	Recomb.	Recomb.& Step.Reduc.
g01	-15.000	-14.7886	$\mathbf{-15.000}$	$\mathbf{-15.0}$	$\mathbf{-15.000}$	-14.440	$\mathbf{-15.000}$	$\mathbf{-15.000}$
g02	0.803619	0.79953	0.803515	0.785	**0.803601**	0.796231	0.803589	0.803592
g03	1.000	0.9997	**1.000**	**1.0**	**1.000**	0.990	0.800	**1.000**
g04	-30665.539	-30664.5	$\mathbf{-30665.539}$	-30665.5	$\mathbf{-30665.539}$	-30626.053	-30665.445	-30665.422
g05	5126.498	–	**5126.497**	5126.5	5126.599	–	5133.935	5126.988
g06	-6961.814	-6952.1	$\mathbf{-6961.814}$	-6961.81	$\mathbf{-6961.814}$	-6952.472	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$
g07	24.306	24.620	**24.307**	24.3323	24.327	31.097	24.360	24.343
g08	0.095825	**0.0958250**	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**
g09	680.63	680.91	**680.630**	**680.630**	680.632	685.994	680.632	680.631
g10	7049.25	7147.9	7054.316	7061.13	**7051.903**	9079.770	7231.497	7062.754
g11	0.75	**0.75**	**0.750**	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**
g12	1.000	0.999999857	**1.000000**	NA	**1.000**	**1.000**	**1.000**	**1.000**
g13	0.053950	NA	**0.053957**	NA	0.053986	0.134057	0.171855	0.058037

Table 6.14: Comparison of the best solutions found by our SMES against the Homomorphous Maps (HM), Stochastic Ranking (SR), ASCHEA, our GA version and two other versions of our SMES: one that uses only recombination and another one that uses both recombination and stepsize reduction. A result in **boldface** indicates either a better result by the corresponding approach or that the global optimum (or best known solution) was reached. "-" means that no feasible solutions were found. NA = Not available.

Problem	Comparison of the mean solution obtained.							
	Optimal	HM	SR	ASCHEA	SMES	GA	Recomb.	Recomb.& Step.Reduc.
g01	-15.000	-14.7082	$\mathbf{-15.000}$	-14.84	$\mathbf{-15.000}$	-14.236	$\mathbf{-15.000}$	$\mathbf{-15.000}$
g02	0.803619	0.79671	0.781975	0.59	0.785238	0.788588	**0.802376**	0.798786
g03	1.000	0.9989	**1.000**	0.99989	**1.000**	0.976	0.529	**1.000**
g04	-30665.539	-30655.3	$\mathbf{-30665.539}$	-30665.5	$\mathbf{-30665.539}$	-30590.455	-30665.445	-30661.106
g05	5126.498	–	**5128.881**	5141.65	5174.492	–	5133.935	5158.739
g06	-6961.814	-6342.6	-6875.940	$\mathbf{-6961.81}$	-6961.284	-6872.204	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$
g07	24.306	24.826	**24.374**	24.66	24.475	34.980	24.472	24.474
g08	0.095825	0.0891568	**0.095825**	**0.095825**	**0.095825**	0.095799	**0.095825**	**0.095825**
g09	680.63	681.16	680.656	680.641	680.643	692.064	**680.637**	**680.637**
g10	7049.25	8163.6	7559.192	**7193.11**	7253.047	10003.225	7355.564	7193.887
g11	0.75	**0.75**	**0.750**	**0.75**	**0.75**	**0.75**	0.752	0.752
g12	1.000	0.999134613	**1.000000**	NA	**1.000**	**1.000**	**1.000**	**1.000**
g13	0.053950	NA	**0.057006**	NA	0.166385	–	0.787648	0.247404

Table 6.15: Comparison of the mean solutions found by our SMES against the Homomorphous Maps (HM), Stochastic Ranking (SR), ASCHEA, our GA version and two other versions of our SMES: one that uses only recombination and another one that uses both recombination and stepsize reduction. A result in **boldface** indicates either a better result by the corresponding approach or that the global optimum (or best known solution) was reached. "-" means that no feasible solutions were found. NA = Not available.

Problem	Comparison of the worst solution obtained.							
	Optimal	HM	SR	ASCHEA	SMES	GA	Recomb.	Recomb.& Step.Reduc.
g01	-15.000	-14.6154	$\mathbf{-15.000}$	NA	$\mathbf{-15.000}$	-14.015	$\mathbf{-15.000}$	$\mathbf{-15.000}$
g02	0.803619	**0.79119**	0.726288	NA	0.751322	0.779140	0.787626	0.785255
g03	1.000	0.9978	**1.000**	NA	**1.000**	0.956	0.294	0.999
g04	-30665.539	-30645.9	$\mathbf{-30665.539}$	NA	$\mathbf{-30665.539}$	-30567.105	-30649.424	-30647.484
g05	5126.498	–	**5142.472**	NA	5304.167	–	5246.968	5201.935
g06	-6961.814	-5473.9	-6350.262	NA	-6952.482	-6784.255	-5218.657	$\mathbf{-6961.814}$
g07	24.306	25.069	**24.642**	NA	24.843	38.686	24.658	24.789
g08	0.095825	0.0291438	**0.095825**	NA	**0.095825**	0.095723	**0.095825**	**0.095825**
g09	680.63	683.18	680.763	NA	680.719	698.297	**680.649**	680.664
g10	7049.25	9659.3	8835.655	NA	7638.366	11003.533	7548.530	**7368.333**
g11	0.75	**0.75**	**0.750**	NA	**0.75**	0.752	0.785	0.767
g12	1.000	0.991950498	**1.000000**	NA	**1.000**	0.999	**1.000**	**1.000**
g13	0.053950	NA	**0.216915**	NA	0.468294	–	–	0.466266

Table 6.16: Comparison of the worst solutions found by our SMES against the Homomorphous Maps (HM), Stochastic Ranking (SR), ASCHEA, our GA version and two other versions of our SMES: one that uses only recombination and another one that uses both recombination and stepsize reduction. A result in **boldface** indicates either a better result by the corresponding approach or that the global optimum (or best known solution) was reached. "-" means that no feasible solutions were found. NA = Not available.

technique reached "better" mean and worst results in ten problems (g01, g03, g04, g05, g06, g07, g08, g09, g10 and g12). A "similar" mean and worst result was found in problem g11. The Homomorphous maps found a "better" mean and worst result in function g02. No comparisons were made with function g13 because such results were not available for HM.

Compared with stochastic ranking (SR)

With respect to SR, our approach was able to find a "better" best result in functions g02 and g10. In addition, it found a "similar" best solution in seven problems (g01, g03, g04, g06, g08, g11 and g12). Slightly "better" best results were found by SR in the remaining functions (g05, g07, g09 and g13). Our approach found "better" mean and worst results in four test functions (g02, g06, g09 and g10). It also provided "similar" mean and worst results in six functions (g01, g03, g04, g08, g11 and g12). Finally, SR found again "better" mean and worst results in function g05, g07 and g13.

Compared with the adaptive segregational constraint handling evolutionary algorithm (ASCHEA)

Compared against ASCHEA, our algorithm found "better" best solutions in three problems (g02, g07 and g10) and it found "similar" best results in six functions (g01, g03, g04, g06, g08, g11). ASCHEA found slightly "better" best results in function g05 and g09.

Additionally, our approach found "better" mean results in four problems (g01, g02, g03 and g07) and it found "similar" mean results in three functions (g04, g08 and g11). ASCHEA surpassed our mean results in four functions (g05, g06, g09 and g10). We did not compare the worst results because they were not available for ASCHEA. Also, We did not perform comparisons with respect to ASCHEA using functions g12 and g13 for the same reason.

As we can see, our approach showed a very competitive performance with respect to these three state-of-the-art approaches.

Confidence intervals

In order to predict the average performance of our approach we performed an statistical test to calculate the confidence intervals for the mean statistic. First of all, we discarded from the test those test functions where our approach reached the global optimum in all 30 runs (g01, g03, g04, g06, g08, g11 and g12). Thus, we plotted the histogram and density line for the remaining test problems (g02, g05, g07, g09, g10 and g13). They are presented in Figure B-1 and B-2. These graphics suggest that none of the distributions are close to be normal. To verify it, we performed a one-sample Kolmogorov-Smirnov test for each sample for each function. In all cases the results proved that the distributions were not close to a normal one. After that, we performed a bootstrapping test. The bootstrapping distribution and also the normal quantile obtained are shown in Figures B-3, B-4 and B-5. As it can be seen, these bootstrapping distributions were close to a normal. For function g13 we calculated the bootstrap bias-corrected accelerated (BCa) to obtain more accurate results (because the bootstrapping distribution was skewed). The summary of results with the confidence intervals for the mean statistic, with 95% confidence is presented in Table 6.17.

The confidence intervals for the mean suggest that the SMES either reaches the global optimum or provides a very good approximation to it. The test problems that presented more difficult to the SMES were g02, g05, g10 and g13; g05 and g13 have 3 nonlinear equality constraints each and g10 has the widest search space based on the bounds defined by each decision variable. For problem g02, which presents many problems to optimization algorithms to consistently reach the vicinity of the best known solution, the mean confidence interval obtained by the SMES is considered competitive. Besides, For problem g10, whose size of the search space is the widest, the obtained confidence interval is also considered competitive. On the other hand, for problems g05 and g13, which contain nonlinear equality constraints, the results obtained were not very competitive. This issue is discussed in-depth in Chapter 8.

Based on confidence intervals, we can allow the SMES to stop when the mean confidence interval seems to have no significative change in a period of time (a certain number

Problem	Optimum	Confidence Interval for the Mean statistic
g01	-15.000	$[\mathbf{-15.000}, \mathbf{-15.000}]$
g02	0.803619	$[0.766579, 0.783869]$
g03	1.000	$[\mathbf{1.000}, \mathbf{1.000}]$
g03	-30665.539	$[\mathbf{-30665.539}, \mathbf{-30665.539}]$
g05	5126.498	$[5158.953, 5200.400]$
g06	-6961.814	$[\mathbf{-6961.814}, \mathbf{-6961.814}]$
g07	24.306	$[24.462, 24.578]$
g08	0.095825	$[\mathbf{0.095825}, \mathbf{0.095825}]$
g09	680.63	$[680.639, 680.649]$
g10	7049.25	$[7236.243, 7325.380]$
g11	0.75	$[\mathbf{0.75}, \mathbf{0.75}]$
g12	1.000	$[\mathbf{1.000}, \mathbf{1.000}]$
g13	0.053950	$[0.111559, 0.244255]$

Table 6.17: Confidence intervals for the sampled mean of the SMES. These intervals were generated using a bootstrapping process. A result in boldface means that the optimum was reached in the 30 independent runs.

of generations). In this case, the number of evaluations can be limited for the ability of the approach to find new solutions. Nonetheless, in this work we centered our effort in comparing it against other approaches where the number of fitness function evaluations is fixed *a-priori*. Anyway, this stop option can be considered when the SMES is used to solve real world engineering design problems.

Advantages of the approach

Our approach can deal with moderately constrained problems (g04), highly constrained problems, problems with low (g06, g08), moderated (g09) and high (g01, g02, g03, g07) dimensionality, with different types of combined constraints (linear, nonlinear, equality and inequality) and with very large (g02), very small (g05 and g13) or even disjoint (g12) feasible regions. Also, the algorithm is able to deal with large search spaces (based on the intervals of the decision variables) and with a very small feasible region (g10). Furthermore, the approach can find the global optimum in problems where such optimum lies on the boundaries of the feasible region (g01, g02, g04, g06, g07, g09). This behavior suggests that the mechanism of maintaining the best infeasible solution helps the search to sample the boundaries between the feasible and infeasible regions.

6.5.3 Finding the strength of the approach

Once we corroborated the effectiveness of our approach, it became particularly relevant to identify the key component (or combination of them) that was mainly responsible for the good performance of our algorithm. For that sake, we designed two experiments.

The aim of the first experiment was to to know which of the three modifications to the $(\mu+\lambda)$-ES was crucial, or if only the combined effect of all three made the algorithm work.

The goal of the second experiment was to reinforce our hypothesis regarding the effectiveness of the self-adaptation mechanism of an ES to sample constrained search spaces.

The experiments consisted on the following:

- *Cross-validation of our ES' mechanisms*: We tested our SMES using each of its mechanisms separately and combining them in pairs, in order to recognize which of them was mandatory. It is important to note that removing the diversity mechanism implies disallowing the best infeasible solution to remain in the population for the next generation of the algorithm. The comparison mechanism based on feasibility remains in all cases in order to guide the search to the feasible region of the search space.

- *ES against GA*: Our second experiment consisted on implementing a real-coded GA with the same combined recombination and the same diversity mechanism used in our SMES. Here, we wanted to see if the use of a GA instead of an ES would make any significant difference in terms of performance.

We will discuss next the results obtained in each of these two experiments.

Cross-validation of our ES' mechanisms

We tested six different versions of our SMES:

- Only combined recombination.
- Only diversity mechanism.
- Only stepsize reduction.
- Combined recombination & diversity mechanism.
- Combined recombination & stepsize reduction.
- Stepsize reduction & diversity mechanism.

	Best solutions obtained by our SMES with its three mechanisms analyzed separately			
Problem	**Optimal**	**Only Combined Recombination**	**Only Diversity Mech.**	**Only Stepsize Reduction**
g01	-15.000	$\mathbf{-15.000}$	$\mathbf{-15.000}$	$\mathbf{-15.000}$
g02	0.803619	**0.803589**	0.763226	0.744524
g03	1.000	0.800	**0.995**	0.482
g04	-30665.539	$\mathbf{-30665.445}$	-30663.625	-30664.609
g05	5126.498	5133.935	5127.187	**5126.938**
g06	-6961.814	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$
g07	24.306	**24.360**	24.576	24.429
g08	0.095825	**0.095825**	**0.095825**	**0.095825**
g09	680.63	**680.632**	680.654	680.654
g10	7049.25	7231.497	7078.823	**7059.549**
g11	0.75	**0.75**	**0.75**	**0.75**
g12	1.000	**1.000**	**1.000**	**1.000**
g13	0.053950	**0.171855**	$0.025667*$	$0.013617*$

Table 6.18: Best solutions found by our SMES with its three mechanisms analyzed separately. "*" means infeasible. A result in **boldface** indicates either a better result by the corresponding approach or that the global optimum (or best known solution) was found.

The parameters used in these six versions are exactly the same used in the experiments described in Section 6.5.1. Thus, the number of evaluations of the objective function is also the same ($240,000$).

The best results obtained for the three first versions (with only one feature) are presented in Table 6.18. Mean results are shown in Table 6.19 and worst results are shown in Table 6.20. The best, mean and worst results obtained for the last three versions (combination of two features) are shown in Tables 6.21, 6.22 and 6.23.

From the results shown in Tables 6.18, 6.19 and 6.20, it is clear that the version with only the combined recombination provided the "better" best results as well as the "better" mean and worst results for most of the functions. The version with only the diversity mechanism obtained "better" best, mean and worst results only for function g03 and was unable to reach the feasible region in g13. The version with only the stepsize reduction obtained "better" best results for functions g05 and g10; and it also obtained a "better" worst result for function g06. However, this version was also unable to reach the feasible region in g13.

With respect to the comparison among versions which use two (out of three) combined mechanisms, the results indicate that the combination of the recombination with the stepsize reduction provided the best and more robust results (see Tables 6.21, 6.22 and 6.23). This version obtained "better" best results for problems g02, g03, g07, g09, and g10. Also, it found "similar" best results for problems g01, g03, g06, g08, g11 and g12. The combined recombination coupled with the stepsize reduction obtained "better" mean results for problems g02, g03, g05, g07, g09, g10, g11 and g13; and it obtained "similar" mean results

	Mean solutions obtained by our SMES with its three mechanisms analyzed separately			
Problem	**Optimal**	**Only Combined Recombination**	**Only Diversity Mech.**	**Only Stepsize Reduction**
g01	-15.000	$\mathbf{-15.000}$	-14.055	-14.493
g02	0.803619	**0.802376**	0.674	0.627237
g03	1.000	0.529	**0.692**	0.212
g04	-30665.539	$\mathbf{-30665.445}$	-30630.231	-30633.003
g05	5126.498	**5133.935**	5373.424	5271.296
g06	-6961.814	$\mathbf{-6961.814}$	-6950.373	-6961.439
g07	24.306	**24.472**	26.883	26.694
g08	0.095825	**0.095825**	**0.095825**	**0.095825**
g09	680.63	**680.637**	681.098	681.299
g10	7049.25	**7355.564**	7783.965	7527.588
g11	0.75	**0.752**	0.755	**0.752**
g12	1.000	**1.000**	**1.000**	**1.000**
g13	0.053950	**0.787648**	$0.052238*$	$0.201332*$

Table 6.19: Mean solutions found by our SMES with its three mechanisms analyzed separately. "*" means infeasible. A result in **boldface** indicates either a better result by the corresponding approach or that the global optimum (or best known solution) was found.

	Worst solutions obtained by our SMES with its three mechanisms analyzed separately			
Problem	**Optimal**	**Only Combined Recombination**	**Only Diversity Mech.**	**Only Stepsize Reduction**
g01	-15.000	$\mathbf{-15.000}$	-10.875	-12.585
g02	0.803619	**0.787626**	0.586408	0.499773
g03	1.000	0.294	**0.441**	0.021
g04	-30665.539	$\mathbf{-30649.424}$	-30447.381	-30582.023
g05	5126.498	**5246.968**	6018.426	6090.623
g06	-6961.814	-5218.657	-6618.615	$\mathbf{-6952.750}$
g07	24.306	**24.658**	38.710	31.982
g08	0.095825	**0.095825**	**0.095825**	**0.095825**
g09	680.63	**680.649**	681.752	683.611
g10	7049.25	**7548.530**	9089.470	8585.027
g11	0.75	0.785	0.824	**0.767**
g12	1.000	**1.000**	0.999	**1.000**
g13	0.053950	$1.0004*$	**0.06212**	$1.965371*$

Table 6.20: Worst solutions found by our SMES with its three mechanisms analyzed separately. "*" means infeasible. A result in **boldface** indicates either a better result by the corresponding approach or that the global optimum (or best known solution) was found.

Problem	Best solutions obtained by our SMES with two of its mechanisms combined.			
	Optimal	Combined Recombination & Diversity Mechanism	Combined Recombination & Stepsize Reduction	Stepsize Reduction & Diversity Mechanism
g01	-15.000	$\mathbf{-15.000}$	$\mathbf{-15.000}$	$\mathbf{-15.000}$
g02	0.803619	0.803549	**0.803592**	0.741027
g03	1.000	0.998	**1.000**	0.725
g04	-30665.539	$\mathbf{-30665.539}$	-30665.422	-30665.318
g05	5126.498	$5105.347*$	5126.988	**5126.534**
g06	-6961.814	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$
g07	24.306	24.353	**24.343**	24.478
g08	0.095825	**0.095825**	**0.095825**	**0.095825**
g09	680.63	680.633	**680.631**	680.671
g10	7049.25	7092.887	**7062.754**	7095.610
g11	0.75	**0.75**	**0.75**	**0.75**
g12	1.000	**1.000**	**1.000**	**1.000**
g13	0.053950	**0.055491**	0.058037	$0.033529*$

Table 6.21: Best solutions found by our SMES with all possible combinations of two of its (three) mechanisms. "*" means infeasible. A result in **boldface** indicates either a better result by the corresponding approach or that the global optimum (or best known solution) was found.

Problem	Mean solutions obtained by our SMES with two of its mechanisms combined.			
	Optimal	Combined Recombination & Diversity Mechanism	Combined Recombination & Stepsize Reduction	Stepsize Reduction & Diversity Mechanism
g01	-15.000	$\mathbf{-15.000}$	$\mathbf{-15.000}$	-14.125
g02	0.803619	0.775841	**0.798786**	0.609223
g03	1.000	0.808	**1.000**	0.315
g04	-30665.539	$\mathbf{-30665.539}$	-30661.106	-30637.253
g05	5126.498	$5249.087*$	**5158.739**	5303.175
g06	-6961.814	-6900.247	$\mathbf{-6961.814}$	$\mathbf{-6961.814}$
g07	24.306	24.559	**24.474**	26.327
g08	0.095825	**0.095825**	**0.095825**	**0.095825**
g09	680.63	680.643	**680.637**	681.040
g10	7049.25	7605.077	**7193.887**	7823.012
g11	0.75	0.754	**0.752**	0.757
g12	1.000	**1.000**	**1.000**	**1.000**
g13	0.053950	0.372581	**0.247404**	$0.108290*$

Table 6.22: Mean solutions found by our SMES with all possible combinations of two of its (three) mechanisms. A result in **boldface** indicates either a better result by the corresponding approach or that the global optimum (or best known solution) was found.

Problem	Worst solutions obtained by our SMES with two of its mechanisms combined.			
	Optimal	Combined Recombination & Diversity Mechanism	Combined Recombination & Stepsize Reduction	Stepsize Reduction & Diversity Mechanism
g01	-15.000	$\mathbf{-15.000}$	$\mathbf{-15.000}$	-11.694
g02	0.803619	0.647445	**0.785255**	0.446562
g03	1.000	0.243	**0.999**	0.088
g04	-30665.539	$\mathbf{-30665.539}$	-30647.484	-30523.984
g05	5126.498	$*5877.772$	**5201.935**	6005.305
g06	-6961.814	-6173.165	$\mathbf{-6961.814}$	-6961.808
g07	24.306	25.136	**24.789**	30.682
g08	0.095825	**0.095825**	**0.095825**	**0.095825**
g09	680.63	**680.664**	**680.664**	681.724
g10	7049.25	13883.840	**7368.333**	9099.229
g11	0.75	0.854	**0.767**	0.909
g12	1.000	**1.000**	**1.000**	**1.000**
g13	0.053950	$1.229679*$	**0.466266**	$0.469699*$

Table 6.23: Worst solutions found by our SMES with all possible combinations of two of its (three) mechanisms. A result in **boldface** indicates either a better result by the corresponding approach or that the global optimum (or best known solution) was found.

for problems g01, g06, g08 and g12. Finally, this version provided "better" worst results for problems g02, g03, g05, g06, g07, g10, g11 and g13, and "similar" worst results for problems g01, g08 and g12.

Based on the results obtained, we decided to compare the results provided by the two most competitive versions of our SMES (the version with only the combined recombination and the version with the combined recombination coupled with the stepsize reduction). The comparison of results is shown in the last three columns from Tables 6.14, 6.15, and 6.16. The results indicated that the version with both the recombination and the stepsize reduction provided "better" best results in seven problems (g02, g03, g05, g07, g09, g10 and g13) and "similar" best results in other five (g01, g06, g08, g11 and g12). This version with two mechanisms reached "better" mean results in three problems (g03, g10 and g13), and "similar" mean results in six functions (g01, g06, g08, g09, g11 and g12). Finally, this version provided "better" worst results in six problems (g03, g05, g06, g10, g11 and g13), and it provided "similar" worst results in three more (g01, g08 and g12). All these results suggest that the stepsize reduction, which provides finer mutation movements in the search space, help the combined recombination to sample the feasible region as to find competitive results.

The main question that arose at this point was: what is the role of the diversity mechanism in the success of our approach? In order to answer this question, we compared the results of the version with combined recombination and stepsize reduction against the version with the three mechanisms. The results can be seen in columns 9 and 6, respectively

from Tables 6.14, 6.15, and 6.16 The complete version provided "better" best results in six functions (g02, g04, g05, g07, g10 and g13), and "similar" best results in other six (g01, g03, g06, g08, g11 and 12). Moreover, the complete version provided "better" mean results for three problems (g04, g11 and g13), and "similar" mean results in other four (g01, g03, g08 and g12). Finally the complete version obtained "better" worst results in three problems (g03, g04 and g11), and it reached "similar" worst solutions for other three (g01, g08 and g12).

Thus, our approach provides results of a better quality when using the diversity mechanism. However, the price paid for this higher quality of results is a slight decrease in robustness. Also, the overall results (providing competitive results in all 13 test functions) are better when the diversity mechanism is incorporated into our SMES. It is also worth reminding that the goal of the diversity mechanism is to allow the search to generate solutions in the boundaries of the feasible region (which is something critical when dealing with constraints that are active in the global optimum). Hence, the use of such diversity mechanism seems a logical choice for dealing with active constraints.

To conclude, the combined recombination seems to be the dominant mechanism, which is assisted by the fine mutation movements provided by the reduction of the initial stepsize. Finally, the diversity mechanism helps to sample solutions located on the boundaries between the feasible and infeasible regions.

ES against GA

For the comparison of performance between a genetic algorithm and an evolution strategy, we used a real-coded GA with non-uniform mutation [128]. Such a GA used the same comparison mechanism (with the diversity mechanism) adopted by our SMES. It is important to note that we tested different mutation operators for real-coded GAs and non-uniform mutation provided the best results. Furthermore, we intended that the GA used the same features of the ES (except for the self-adaptive mutation which we hypothesized was the main strength of our ES-based approach). Finally, the same dynamic mechanism to handle the tolerance for equality constraints was employed.

The parameters used by our real-coded GA were the following:

- Population size: 200
- Maximum number of generations: 1200
- Crossover rate: 0.8
- Mutation rate: 0.6

- Number of objective function evaluations: $240,000$ (the same performed by our SMES).

We performed 30 runs for each test problem. The results obtained by the GA are presented in Tables 6.14, 6.15 and 6.16 in column 7 and they are compared against those provided by the SMES in column 6. As can be seen, both the quality and robustness of the results provided by the GA are significantly poorer than those obtained with the evolution strategy in all the test functions adopted. The exceptions are g08, g11 and g12, in which the GA was able to find competitive results. These results highlight the strong influence (positive in this case) of using a more adequate search engine, in our case an ES over a GA. Therefore, the results seem to confirm our initial hypothesis about the usefulness of an ES to sample constrained search spaces in a more appropriate way.

6.6 Some Remarks

Besides still being a very simple approach, it is worth reminding that the parameters of the algorithm do not require a fine-tuning like penalty factors. In contrast, the Homomorphous Maps require an additional parameter (called v) which has to be found empirically [110]. Stochastic ranking requires the definition of a parameter called P_f, whose value has an important impact on the performance of the approach [162]. ASCHEA also requires the definition of several extra parameters, and in its latest version, it uses niching, which is a process that also has at least one additional parameter [77].

The computational cost measured in terms of the number of fitness function evaluations (FFE) performed by any approach is lower for the $(\mu + \lambda)$-ES with respect to the others to which it was compared. This is an additional (and important) advantage, mainly if we wish to use this approach for solving real-world problems. The $(\mu + \lambda)$-ES performed $240,000$ FFE, the $(\mu + 1)$-ES required $350,000$ FFE, the$(1 + \lambda)$-ES $330,000$ FFE, the Stochastic Ranking performed $350,000$ FFE, the Homomorphous Maps performed $1,400,000$ FFE, and ASCHEA required $1,500,000$ FFE.

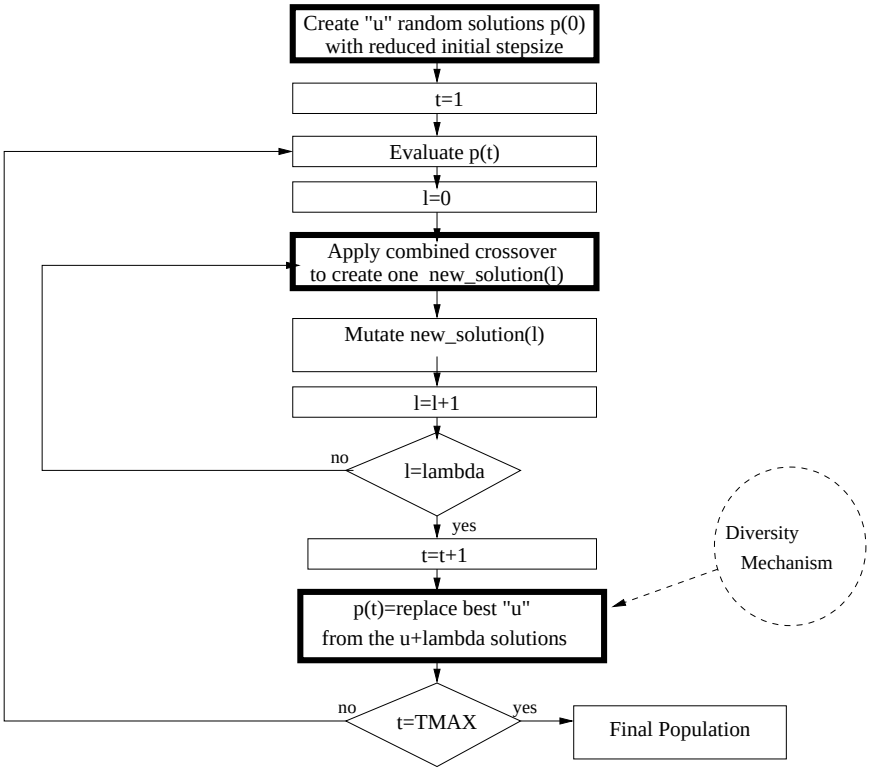


Figure 6.10: Algorithm of our $(\mu + \lambda)$-ES (SMES). The thick boxes indicate the three modifications made to the original ES

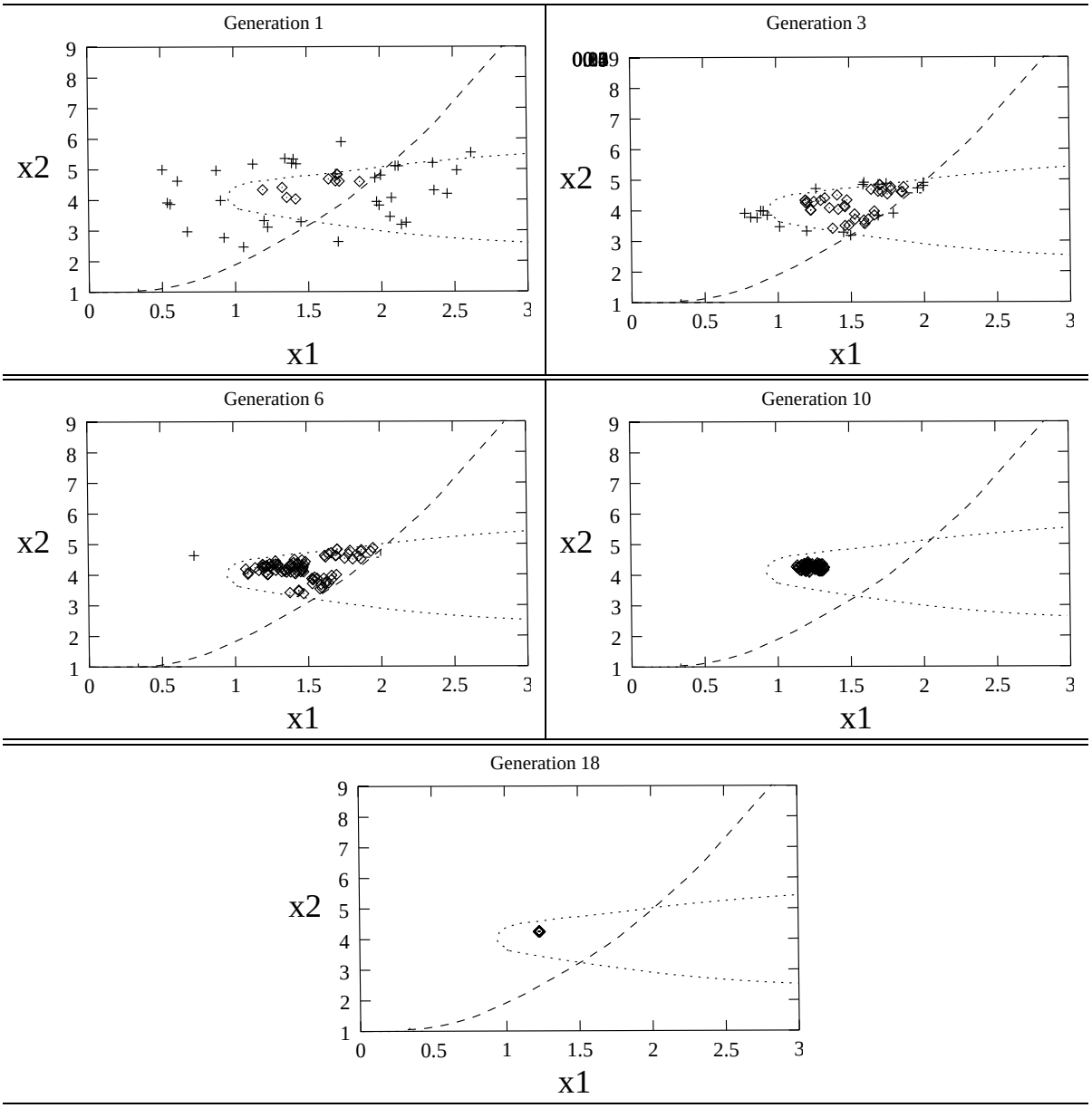


Figure 6.11: Graphs showing the population behavior using our proposed $(\mu + \lambda)$-ES. "$\diamond$" points are feasible solutions, "+" points are infeasible ones. The dashed line represents constraint $g_1(x)$ of the problem and the dotted line represents constraint $g_2(x)$.

Chapter 7

Performance Measures

After comparing the statistical results of SMES against state-of-the-art approaches, we decided to investigate four issues:

1. To analyze the effect of the diversity mechanism in order to maintain infeasible solutions close to the feasible region during the evolutionary process.

2. To analyze how fast the global optimum is reached in those functions where this solution is found.

3. To find out if SMES shows sensitivity to either any of its parameter or a combination of them. The aim would be to be able to recommend adequate values for these parameters.

4. To define or use performance measures in order to compare the behavior of SMES against the most competitive constraint-handling approach found in the literature.

The four sections of this chapter provide answers to these questions by using statistical tools like the analysis of variance, the Kolmogorov-Smirnov test and bootstrapping.

7.1 Maintaining Good Infeasible Solutions

One of the three mechanisms of SMES is the diversity mechanism, whose goal is to maintain a few infeasible solutions close to the feasible region to help the algorithm to sample its boundaries and locate the global optimum (let us keep in mind that in many complex constrained problems, the global optimum is located on the boundary between the feasible and infeasible regions). To study this issue, we monitored the percentage of feasible solutions

in the population of our SMES at every 200 generations (the total number of generations was fixed to 800). The results are presented in Figure 7.1(a). As it can be seen, for all the test problems our approach reaches the feasible region by generation 200. For problem g05, more than 20% of the population is feasible by generation 200 and for the remaining functions almost all the population is feasible by then. Based on the results found, we were interested in answering two questions:

1. What is the behavior before generation 200. In other words, how fast does the population become almost feasible (where "almost feasible" refers to have a population in which at least 50% of the individuals are feasible)?

2. How well is the diversity maintained at late stages of the evolutionary process?

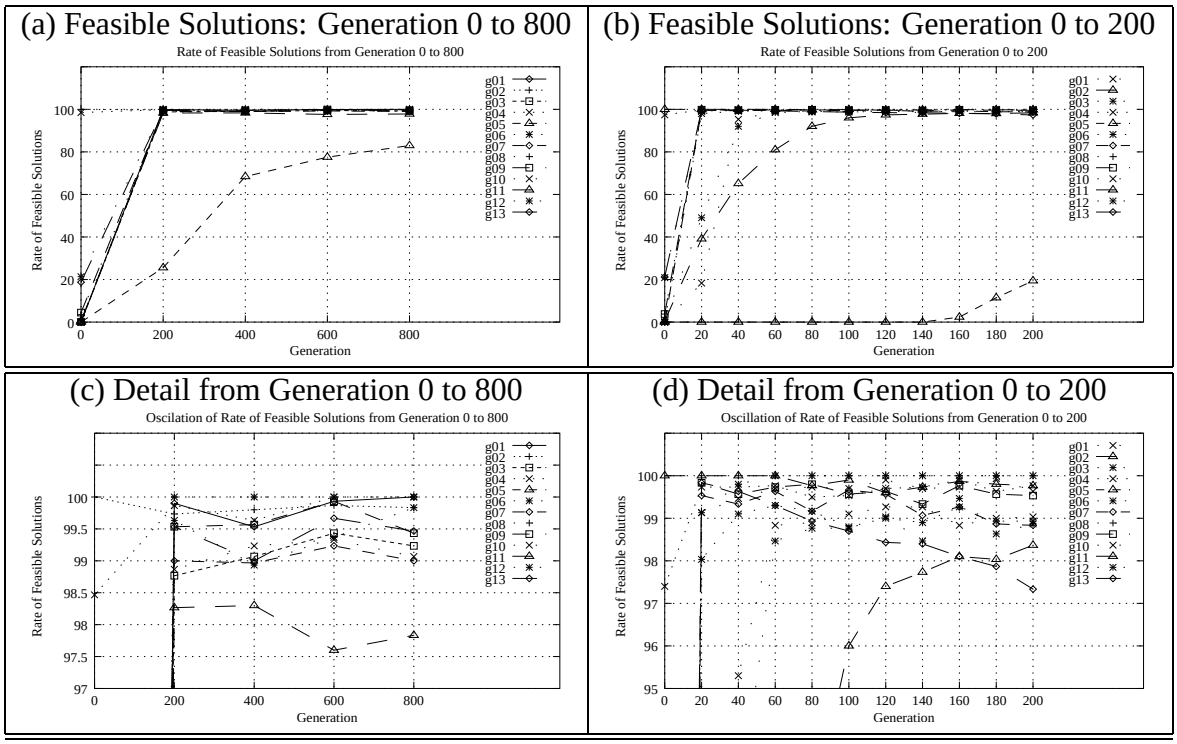


Figure 7.1: Percentage of feasible solutions (a) at every 200 generations (from 0 to 800), (b) at every 20 generations (from 0 to 200), (c) a detailed oscillation of feasible and infeasible solutions (from 0 to 800) and (d) a detailed oscillation of feasible and infeasible solutions (from 0 to 200)

	SMES after 20 generations.					
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	-15.000	-7.291	-5.686	-5.685	-3.820	9.05E-1
g02	0.803619	0.442641	0.372686	0.370697	0.317450	2.65E-2
g03	1.000	0.949	0.785	0.827	0.526	1.13E-1
g04	-30665.539	-30563.615	-30473.319	-30462.027	-30401.756	40.26E+0
g05	5126.498	$*5067.897$	5211.511	5199.974	$*5643.923$	103.26E+0
g06	-6961.814	-6890.164	-6235.589	-6188.203	-5552.386	371.89E+0
g07	24.306	62.136	135.969	121.868	682.452	107.72E+0
g08	0.095825	**0.095825**	**0.095825**	**0.095825**	0.095817	2.0E-6
g09	680.63	686.592	704.351	704.377	719.151	8.91E+0
g10	7049.25	12777.324	17407.559	17284.081	25774.398	2368.59E+0
g11	0.75	**0.750**	0.783	0.764	$*0.897$	4.07E-2
g12	1.000	0.999	0.999	0.999	0.999	7.0E-5
g13	0.053950	$*0.001348$	$*0.009035$	$*0.004388$	$*0.026345$	8.45E-3

Table 7.1: Statistical results of our approach after 20 generations. ("*" means infeasible). A value in **boldface** indicates that the global optimum (or best known solution) was reached.

The results obtained for those two questions are shown in Figure 7.1(b) and 7.1(d) for question 1 and in Figure 7.1(c) for question 2. Figure 7.1(b) shows that the feasible region is reached at generation 20 in most cases. This means that (except for g05 and g13) the approach only requires $6,100$ FFE to find feasible solutions. In Table 7.1, we show the statistical results obtained at this stage of the search. Note that although the results are still far from the optimum, with the exception of problems g05 and g13, most of the solutions are feasible. In Figure 7.1(d) we observe in a close-up of Figure 7.1(b) that the algorithm has the capability of maintaining some infeasible solutions despite the almost-feasible population (which, indeed, is the main motivation of using the diversity mechanism adopted). In addition, we show the statistical results obtained at generation 200 in Table 7.2. A substantial improvement of the quality and robustness of the results is shown at generation 200 where only $20,000$ FFE have been performed. Indeed, the results are close to the optimum in most of the problems (for problems g08 and g12 the algorithm has reached the global optimum). This means that the approach is about to converge in most cases. This highlights the importance of the diversity mechanism in order to avoid that the algorithm gets trapped in local optima and it can reach a better solution (even the global optimum).

On the other hand, Figure 7.1(c) shows a zooming of Figure 7.1(a), where it is possible to see again in detail the smooth oscillation on the percentage of feasible solutions during the evolutionary process after generation 200. This behavior suggests that the diversity mechanism still works well, maintaining near-feasible solutions with a good value of the objective function in the population (between 1 and 3 infeasible solutions are enough based on the previous results of the $(1+\lambda)$-ES approach [124], which is able to avoid local optima

	SMES after 200 generations.					
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	−15.000	−14.999	−14.960	−14.999	−13.828	2.10E-1
g02	0.803619	0.801158	0.777458	0.787125	0.678203	2.43E-2
g03	1.000	**1.000**	0.999	0.999	0.987	3.74E-3
g04	−30665.539	**−30665.539**	−30665.531	−30665.536	−30665.473	1.35E-2
g05	5126.498	5126.988	5179.163	5162.323	5379.227	63.5E+0
g06	−6961.814	−6961.808	−6959.910	−6961.624	−6938.690	4.39E+0
g07	24.306	24.473	24.734	24.711	25.401	2.15E-1
g08	0.095825	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0
g09	680.63	680.643	680.680	680.680	680.736	2.43E-2
g10	7049.25	7076.725	7330.398	7319.405	7816.830	153.72E+0
g11	0.75	**0.75**	**0.75**	**0.75**	0.76	3.07E-3
g12	1.000	**1.000**	**1.000**	**1.000**	**1.000**	0
g13	0.053950	∗0.041436	0.145069	∗0.045766	0.387152	1.53E-1

Table 7.2: Statistical results of our approach after 200 generations. ("*" means infeasible). A value in **boldface** indicates that the global optimum (or best known solution) was reached.

with only a few copies of the best infeasible solution).

The final results (on generation 800) provided in Table 6.13, compared with those on generation 200 (Table 7.2), suggest that our diversity mechanism does its job of avoiding premature convergence and, when coupled with the combination of discrete/intermediate recombination and the self-adaptation mechanism of the ES leads the evolutionary search towards the global optimum of a problem.

It is important to remark that the process of finding the global optimum takes almost $3/4$ of the evolutionary search and only $1/4$ (or less) is necessary to find the feasible region of the search space. This behavior is analyzed in detail later in this Chapter. The point we want to make here is that our approach is fast at reaching the feasible region while managing to avoid local attractors (thanks to its diversity mechanism) as to converge to the global optimum or its close vicinity.

7.2 Reaching the Global Optimum

In Table 7.3 we show in how many runs the global optimum was reached for the test functions where such a thing was possible. In addition, we show the lowest and the average generation at which the optimum was found. The results obtained suggest that for those problems where the global optimum is reached, the algorithm is able to find it using no more than 250 generations (about $75,000$ evaluations of the objective function), except for function g01 where the number of generations is 671.

Problem	Runs that find the optimum	Lowest generation	Average
g01	30	634	671
g03	30	41	184
g04	30	113	129
g06	15	47	249
g08	30	11	18
g11	30	28	88
g12	30	63	77

Table 7.3: Number of runs (out of 30) where the optimum is found. We also show the best and average generation number at which the optimum is found.

The results on Table 7.3 suggest that our approach is able to find the global optimum relatively fast for different types of problems. Coupled with its speed to find feasible region, SMES has the feature of providing reasonably good results with a low computational cost (measured by the number of fitness function evaluations).

7.3 Analysis of Variance

An Analysis of Variance (ANOVA) was performed to determine the sensitivity of our final approach to its parameters using a subset of the test functions indicated in Appendix A. The justification of using just a subset of the set of functions adopted in this dissertation is presented after the details of the experimental design shown below.

- The parameters (factors) analyzed were μ, λ, G_{max}, the reduction of the stepsize and S_r (the stepsize reduction and S_r remained fixed in the original SMES).
- The levels for each parameter were defined as follows:
  - μ: 50, 150
  - λ: 200, 400.
  - G_{max}: 600, 1000.
  - *Stepsize-reduction*: 0.1, 0.4, 0.9.
  - S_r: 0.2, 0.5, 0.97.
- 72 different combinations of initial parameters were obtained.

- 30 independent runs per combination per test function were performed.

- 28080 total independent runs were performed.

A previous ANOVA with just two levels of each parameter was performed in order to detect which parameters showed sensitivity. Therefore, the first three parameters have only two levels. Also, from this previous ANOVA and the statistical results, some test functions were eliminated owing to the fact that they also showed no variation in the results provided by SMES. The functions used in ANOVA were: g02, g05, g07, g10 and g13. The hypotheses used in this experiment were the following:

Null Hypothesis: There is no a significative difference among the averages of the obtained results and if there are differences, they are due to random effects.

Alternative Hypothesis: There is a combination of factors where the averages are different and they are not due to random effects.

The results obtained proved the Null Hypothesis for most of the combination of the parameters. Nonetheless, for some of them the Alternative Hypothesis was proved. These cases are shown in Figure 7.2 for function g02, in Figure 7.3 for function g05, in Figure 7.4 for function g07, in Figure 7.5 for function g10 and finally in Figure 7.6 for function g13. In those graphics, we show the error obtained in each independent run (i.e. the absolute value of the difference between the global optimum solution, or best known solution, and the best solution obtained in such run). Therefore, a small value of the error means a better result provided by the approach.

From the results that proved the Alternative Hypothesis, the following can be stated:

- For function **g02**: Better results are obtained with $\lambda = 400$, $Stepsize - reduction = 0.9$ and $S_r = 0.97$

- For function **g05**: Better results are obtained with $Stepsize - reduction = 0.1$ and $S_r = 0.97$.

- For function **g07**: Better results are obtained with $S_r = 0.97$.

- For function **g10**: Better results are obtained with $\lambda = 400$, $G_{max} = 1000$, $Stepsize - reduction = 0.1 - 0.4$ and $S_r = 0.97$.

- For function **g13**: Better results are obtained with $\lambda = 400$.

The results mentioned above confirm the empirical selection of parameters used in the experiments with SMES. However, it is important to notice that allowing the generation of more offspring at each generation ($\lambda = 400$) helps the approach to improve the results. Besides, SMES seems to be insensitive to the number of parents (μ) at each generation. Another important aspect is that when the G_{max} value is increased, the quality of results improves for function g10, where the search space is one of the largest (due to the intervals of the decision variables). For the $Stepsize - reduction$ value, the ANOVA also provided some interesting results: For function g02, its advisable value is 0.9, which means that SMES requires a large initial stepsize to move inside a very large feasible region (about 99% of the whole search space). On the other hand, for problem g10, whose feasible region is very small with respect to the whole search space, the adequate value for $Stepsize - reduction$ is either 0.1 or 0.4. This is due to the fact that when the feasible region is reached, the approach requires fine movements in order to sample it in a better way as to approximate the global optimum. Finally, we confirmed that $S_r = 0.97$ is an adequate value for all these functions. This highlights the idea that only a few infeasible solutions close to the boundary between the feasible and infeasible region are necessary to maintain diversity and to sample this region in a proper way. As a final conclusion of the performed ANOVA, we suggest the following parameters for the SMES:

- $\mu = 50$
- $\lambda = 300$ (400 to improve the quality of the results, at the expense of more evaluations of the fitness function).
- $G_{max} = 800$ (1000 to improve the quality of the results, at the expense of more evaluations of the fitness function).
- $Stepsize - reduction = 0.1$ for small feasible regions (0.9 for large feasible regions) both with respect to the whole search space.
- $S_r = 0.97$

7.4 Three Measures of Performance

Despite the extensive amount of research made in the area of constraint-handling mechanisms for evolutionary algorithms, there are very few measures of performance proposed to compare the performance of different approaches. In this section, we use three measures:

1. The first one was used by Lampinen [113] to measure how many evaluations are required to find the first feasible solution; we call it EVALS.

2. The second one was proposed by Bäck [6] to measure the progress ratio for unconstrained optimization. We adapt it to measure the progress ratio just inside the feasible region, because we have detected that for constraint handling algorithms solving constrained problems, is quite hard to improve solutions once inside the feasible region. The original expression taken from [6] is the following:

$$P = \ln \sqrt{\frac{f_{\min}(0)}{f_{\min}(T)}} \tag{7.1}$$

   where $f_{\min}(i)$ refers to the best objective function value occurring at generation i. T is the final generation of the process. However, we are interested in measuring the progress once the feasible region is reached. Therefore the modified expression that we use is the following:

$$P = \left| \ln \sqrt{\frac{f_{\min}(G_{\text{ff}})}{f_{\min}(T)}} \right| \tag{7.2}$$

   where $f_{\min}(G_{\text{ff}})$ refers to the objective function value of the first feasible solution found and $f_{\min}(T)$ refers to the objective function value of the best feasible solution found in the last generation.

3. The third measure counts the number of evaluations required by an algorithm to have its population with only feasible solutions. We call it ALL-FEASIBLE. This measure estimates the computational effort needed by the algorithm to have all its solutions inside the feasible region.

Using EVALS and PROGRESS-RATIO we can know how fast the approach reaches the feasible region (EVALS) and we can measure how much is the approach capable of improving the first feasible solution produced (PROGRESS-RATIO) i.e. how well is the feasible region being sampled. Finally, ALL-FEASIBLE helps us to validate the usefulness of the diversity mechanism, whose aim is to maintain infeasible solutions during all the evolutionary process.

The main difference between EVALS and ALL-FEASIBLE is that EVALS estimates the number of evaluations required by an algorithm to find the first feasible solution and ALL-FEASIBLE counts the number of evaluations required by an approach to have its population with only feasible solutions

Technique	SMES	SR
ES Type	$(100 + 300)$-ES	$(30, 200)$-ES
Generations	800	1200
Total Evaluations	240000	240000
P_f	–	0.45
Sweeps for ranking	–	200
S_r	0.97	–
Initial Stepsize	40%	–

Table 7.4: Summary of parameters for the SMES and the Stochastic Ranking (SR) used in the experiments for the 3 performance measures. "–" means "not applicable".

It is important to remark that both, the EVALS and ALL-FEASIBLE performance measures return an integer value. If a technique obtains a lower value than other one, it means that the first one finds the feasible region faster than the second one. On the other hand, the PROGRESS-RATIO performance measure returns a real value. If an approach gets a higher value for this performance measure, it means an improvement inside the feasible region.

7.4.1 Experiments

In order to perform a fair comparison using our SMES, we chose the most competitive approach out of the three that we are comparing, the Stochastic Ranking (SR) [162] and we implemented it. We maintained the same set of parameters for the SMES used in the comparison included in the previous chapter. The parameters adopted for Stochastic Ranking were chosen according to the authors' recommendations found in [162]. The total number of evaluations of the objective function was fixed to 240000 for both approaches. Also, because of the fact that the SMES uses a dynamic adjustment for the allowable tolerance of the equality constraints and SR does not, we decided for the SMES to consider a feasible solution only when the same value for tolerance used by SR ($\epsilon = 0.0001$) was reached. 30 independent runs were performed for each approach for each test function for each performance measure. In Table 7.4 there is a summary of the parameters adopted.

EVALS performance measure

The summary of results for the EVALS performance measure obtained by each approach for each test problem is presented in Table 7.5.

Statistics EVALS measure.										
	Min		Mean		Median		Max		Std. Dev.	
	SMES	SR	SMES	SR	SMES	SR	SMES	SR	SMES	SR
g01	**1315**	3585	**2455**	9146	**2523**	9139	**3444**	15352	**572.69**	2857.41
g02	1	1	1	1	1	1	1	41	0.00	0.00
g03	**868**	1823	**2745**	6674	**2341**	6622	**7554**	14026	**1697.76**	3253.81
g04	1	1	4	4	**2**	3	17	**9**	3.58	**2.01**
g05	51139	**24727**	124120	**27843**	84697	**28038**	240100	**31694**	69209.56	**1674.18**
g06	674	**262**	1552	**1181**	1547	**1245**	2395	**1711**	494.92	**371.18**
g07	**1211**	2087	**2024**	3093	**2064**	3150	**2772**	4049	**446.85**	480.56
g08	**1**	3	**91**	110	**76**	78	**292**	428	**73.79**	104.72
g09	**3**	6	132	**122**	120	**85**	**359**	375	**90.67**	109.14
g10	**1155**	7864	**4630**	77222	**4547**	27012	**6907**	240000	**1341.02**	92708.03
g11	**30**	503	**4290**	26463	**2799**	2847	**16204**	240000	**4311.14**	72406.28
g12	2	**1**	**21**	26	**15**	23	92	**91**	**22.52**	24.26
g13	209429	**16578**	212579	**18949**	212686	**18619**	214125	**21616**	**967.17**	1439.51

Table 7.5: Statistics obtained for the EVALS performance measure in 30 independent runs. A number in **boldface** indicates the best result found.

From these results we can comment the following: For functions g02 and g04 both approaches find feasible solutions from the first population randomly generated.

For problem g12 both algorithms exhibit a very similar behavior, although the SMES has better mean, median and standard deviation values and the SR has better min and max values. For problem g09 there is an inverse behavior, the SMES has better min and max values and the SR has better mean and median values.

The SMES consistently provided better results for six functions g01, g03, g07, g08, g10 and g11. The SR obtained better results for three problems g05, g06 and g13. It is important to remark that the SMES obtained generally smaller standard deviation values than the SR, which means more consistent results by the SMES.

The overall discussion of the results suggests that the SMES finds the feasible region faster than the SR, except in problems where there is more than one nonlinear equality constraint (g05 and g13); the exception is g06, which is a problem where the SR does not provide robust results [162]. On the other hand, both approaches reach the feasible region almost at the same time when the size of the feasible region is approximately more than 4% of the whole search space (g02, g04 and g12). The exception is problem g09 where the feasible region is about 0.51% of the whole search space (which is a large value when compared to other problems with a very small feasible region, of about 0.0003%).

PROGRESS-RATIO performance measure

After analyzing how fast the algorithms reach the feasible region, we now focus on the capabilities of both algorithms to improve the first feasible solution that they find. For this

sake, we used the progress ratio measure presented in Equation 7.2. The results obtained for the SMES and the SR are detailed in Table 7.6:

	Statistics PROGRESS-RATIO measure.									
	Min		Mean		Median		Max		Std. Dev.	
	SMES	SR	SMES	SR	SMES	SR	SMES	SR	SMES	SR
g01	**1.229**	1.166	**1.324**	1.286	**1.324**	1.284	1.399	**1.408**	**0.041**	0.054
g02	**0.251**	0.244	**0.262**	0.260	**0.264**	0.260	0.276	**0.281**	**0.008**	0.009
g03	0.092	($\frac{13}{30}$)**0.166**	0.241	($\frac{13}{30}$)**0.307**	0.253	($\frac{13}{30}$)**0.327**	0.338	($\frac{13}{30}$)**0.346**	0.064	($\frac{13}{30}$)**0.051**
g04	3.553	**3.588**	4.081	**4.182**	4.123	**4.194**	**4.394**	4.387	0.212	**0.180**
g05	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
g06	3.225	($\frac{17}{30}$)**3.749**	4.045	($\frac{17}{30}$)**4.124**	**4.129**	($\frac{17}{30}$)4.125	**4.320**	($\frac{17}{30}$)4.283	0.266	($\frac{17}{30}$)**0.159**
g07	**1.159**	($\frac{29}{30}$)0.873	**1.998**	($\frac{29}{30}$)1.400	**2.043**	($\frac{29}{30}$)1.406	**2.436**	($\frac{29}{30}$)2.038	0.299	($\frac{29}{30}$)**0.251**
g08	0.014	**0.015**	**0.045**	0.042	**0.046**	0.045	**0.087**	0.079	0.012	**0.011**
g09	0.212	**0.359**	**2.749**	2.387	**2.823**	2.244	**4.738**	4.564	1.425	**1.242**
g10	**0.325**	($\frac{25}{30}$)0.004	**0.497**	($\frac{25}{30}$)0.133	**0.514**	($\frac{25}{30}$)0.079	**0.625**	($\frac{25}{30}$)0.459	**0.073**	($\frac{25}{30}$)0.139
g11	**0.004**	($\frac{27}{30}$)0.000	**0.098**	($\frac{27}{30}$)0.054	**0.114**	($\frac{27}{30}$)0.027	**0.144**	($\frac{27}{30}$)0.142	**0.046**	($\frac{27}{30}$)0.052
g12	**0.032**	0.013	0.092	**0.093**	0.089	**0.094**	0.147	**0.161**	**0.033**	0.041
g13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.002**	0.000	0.000

Table 7.6: Statistics obtained for the PROGRESS-RATIO performance measure in 30 independent runs. A number in **boldface** indicates the result found. For the SR, the fraction indicates the number of independent runs (out of 30) where feasible solutions were found in the population of the last generation. In the remaining runs, no feasible solutions were found at the end of the process. Note that SMES obtained feasible solutions in every run for each problem.

From these results, we observe very similar values of improvement by both approaches in four problems (g05, g08, g12 and g13). It is important to assert that in functions g05 and g13 none of the algorithms was capable of improving the feasible solutions found. Both problems have three nonlinear equality constraints. For problems g08 and g12 both approaches could get a small improvement inside the feasible region. This is because for these two functions feasible solutions are found very quickly and after that the global optimum is found quickly, as well. This is because these two problems are "easy" to solve.

The SR presented some problems to maintain the feasible solutions previously found. For example, in function g03 only in 13 runs out of 30, we were able to find feasible solutions in the last generation. A similar behavior was found in problems g06 (feasible solutions were found at the end of the process only in 17 runs), g07 ($29/30$), g10 (25/30) and g11 (27/30). Nevertheless, SR obtained consistently better results in problem g04.

The results show that SMES was able to get a better improvement inside the feasible region in eight problems: g01, g02, g03, g06 g07, g09, g10 and g11. It is worth reminding that SMES did not presented any problem to maintain good feasible solutions until the end of a single run. Therefore we consider better the results provided by SMES in functions

g03 and g06 (where the SR had problems to keep feasible solutions). In problems g01, g02, g07, g09, g10 and g11 SMES was superior in all statistics except in the minimum ratio and its standard deviation for problem g09. Also, it was surpassed in the standard deviation value in problem g07.

Based on this discussion of results we can empirically suggest that the mechanism to maintain diversity and the combined recombination operator allow the SMES to sample the feasible region well enough as to maintain a considerable improvement inside the feasible region. On the other hand, the stochastic selection mechanism of the SR and its "," selection prevent the SR from keeping good feasible solutions during the process and, in consequence, affect its progress inside the feasible region.

ALL-FEASIBLE performance measure

Finally, we used the ALL-FEASIBLE performance measure in order to know the computational cost, measured by the number of objective function evaluations, required by each approach to get a complete feasible population. This measure is also useful to estimate the effectiveness of the diversity mechanisms of each approach: The SMES maintains a few (3% of the whole population) infeasible solutions with a good value of the objective function close to the boundaries of the feasible region and the SR allows a certain percentage (45% of the whole population) of solutions with a good value of the objective function, regardless of their feasibility. It is well known that both algorithms provide competitive results, but now we want to estimate how fast the approaches have all their individuals feasible; this can give us an idea of the time (after all the population is feasible) at which the diversity mechanism operates in order to generate infeasible solutions to sample the boundaries of the feasible region. The results are provided in Table 7.7.

The SMES required less evaluations to get a fully feasible population in eight functions (g01, g04, g06, g07, g08, g09, g10 and g12). This is because the approach focuses on finding the feasible region and after that, the diversity mechanism allows the approach to return to its boundaries. This was empirically shown in Section 7.1.

The SR provided better results in three problems (g05, g11 and g13). It is interesting to note that the mechanism to maintain solutions with a good value of the objective function regardless of feasibility is suitable to solve these problems with a very small feasible region and in presence of one up to three nonlinear equality constraints. This behavior may be attributed to the fitness landscape of the function (very rugged) defined by the nonlinear objective function in these three problems.

In function g02 both approaches got a fully feasible population in the first generation because almost all the search space (99%) is feasible. For problem g03, none of the approaches could get a fully feasible population. This problem has one nonlinear inequality

Statistics ALL-FEASIBLE measure.										
	Min		Mean		Median		Max		Std. Dev.	
	SMES	SR	SMES	SR	SMES	SR	SMES	SR	SMES	SR
g01	**4900**	13230	**6050**	19050	**5800**	18430	**9100**	25630	**822.84**	2936.27
g02	100	30	110	30	100	30	400	30	54.77	0.00
g03	240000	240000	240000	240000	240000	240000	240000	240000	0.00	0.00
g04	**400**	830	**540**	3877	**400**	3630	**700**	7030	**152.22**	1864.87
g05	240000	**30230**	240000	**40323**	240000	**38030**	240000	**58830**	0.00	7627.13
g06	4900	**4830**	**5850**	108344	**5800**	11630	**7600**	240000	**726.71**	117130.5
g07	**4300**	5430	**4950**	8350	**4900**	7530	**5800**	33230	**417.50**	4817.67
g08	**1600**	1830	**1950**	2383	**1900**	2430	**2800**	3230	344.16	**322.42**
g09	**1300**	2430	**1580**	3717	**1600**	3530	**2200**	5230	**248.30**	663.71
g10	**8500**	19430	**10440**	65706	**10150**	33530	**14800**	240000	**1500.48**	72642.61
g11	240000	**8830**	240000	**35940**	240000	**13130**	240000	240000	**0.00**	69292.97
g12	**1600**	2430	**2290**	3023	**2200**	3030	**3400**	3830	480.19	**308.43**
g13	215500	**22030**	215530	**29710**	215500	**27030**	215800	**57030**	**91.54**	8347.55

Table 7.7: Statistics obtained for the ALL-FEASIBLE performance measure in 30 independent runs. A number in **boldface** indicates the best result found.

constraint, 10 decision variables and a very small feasible region (only 0.0026% of the search space is feasible).

It is important to note that for the SMES, the diversity mechanism works during almost all the evolutionary process because the population becomes feasible very quickly. Besides, the importance of the diversity mechanism was highlighted in the experiments performed using the three mechanisms of the SMES (diversity mechanism, combined recombination and stepsize reduction) separately in Section 6.5.3. Finally, it is evident that the size of the parent population for the SMES was more than three times the corresponding parent population of the SR. This seemed to be a disadvantage for the SMES. However, the approach was able to generate feasible solutions faster than the SR as to get a fully feasible population. Nevertheless, this behavior seems to impact performance in functions like g05, g11 and g13 where the SR outperformed the SMES.

7.4.2 Confidence intervals

After discussing the results obtained for a sample of 30 runs, we performed a statistical test to estimate the confidence intervals for the mean statistics. We chose the mean because we wanted to compare the average performance of both approaches with respect to the three performance measures previously indicated.

First of all, we plotted the density graphs for each sample of 30 runs per each approach per function per performance measure in order to graphically find out if the distributions were close to a normal one. The density histograms are shown as follows: For the EVALS

measure, we show in Figures C-1, C-2 and C-3 the results provided by the SMES and we show in Figures C-4, C-5 and C-6 the results for the Stochastic Ranking. Test function g02 was omitted because both approaches found a feasible solution in the initial (randomly-generated) population. For the PROGRESS-RATIO measure the results of the SMES are presented in Figures C-7, C-8, C-9 and C-10 are shown in Figures C-11, C-12, C-13 and C-14 for the Stochastic Ranking. Finally, for the All-FEASIBLE measure the results for the SMES are provided in Figures C-15, C-16, C-17 and C-18, and are shown in Figures C-19, C-20, C-21 and C-22 for the Stochastic Ranking. All these graphics can be found in Appendix C.

From these graphics it is distinguishable that in all cases the distribution of data is skewed and far from adjusting to a normal distribution. However, we decided to perform a one-sample Kolmogorov-Smirnov test to validate it. The results from the Kolmogorov-Smirnov test confirmed in all cases that the distributions were not normal. Therefore, we decide to use a Bootstrapping test to obtain the confidence intervals. The advantage of using bootstrapping is that it does not require any assumption about the distribution of the sample. Based on 1000 resamples the distribution obtained for each sample and its corresponding normal quantile (to confirm that the bootstrap distribution is nearly normal in shape) are shown as follows: for the EVALS measure we show in Figures C-23, C-24, C-25, C-26, C-27 and C-28 the results provided by the SMES and we show in Figures C-29, C-30, C-31, C-32, C-33 and C-34 the results provided by the SR (reminding that the test function g02 is omitted). For the PROGRESS-RATIO the bootstrap distribution and the normal quantile graphics for the results provided by the SMES are presented in Figures C-35, C-36, C-37, C-38, C-39, C-40 and C-41 and we present in Figures C-42, C-43, C-44, C-45, C-46, C-47 and C-48 the results provided by the SR. Finally, for the ALL-FEASIBLE performance measure, the bootstrapping results for the SMES are presented in Figures C-49, C-50, C-51, C-52 and C-53, and in Figures C-54, C-55, C-56, C-57, C-58 and C-59 for the SR. For the SMES, test functions g03, g05 and g11 were omitted because the approach could not generate a fully feasible population in any single run. This is the same case for the SR in function g03. Function g02 was also omitted in both cases because both approaches generated a fully feasible population in the initial population.

For those skewed bootstrapping distribution we calculated the bootstrap bias-corrected accelerated (BCa) in order to obtain more accurate results. The confidence intervals were calculated with 95% confidence. A summary of them for the three performance measures is presented in Table 7.8 and it is discussed next.

First, we center the discussion on the EVALS confidence intervals. The estimated mean for the EVALS performance measure obtained by the SMES was lower than the corresponding mean provided by the SR in six problems (g01, g03, g07, g08, g10 and g11). The SR found a “better” mean in three problems (g05, g06 and g13). Also, both approaches pro-

	Evals		Progress		All-feasible	
	SMES	**SR**	**SMES**	**SR**	**SMES**	**SR**
g01	$[\mathbf{2249}, \mathbf{2642}]$	$[8103, 10189]$	$[\mathbf{1.308}, \mathbf{1.338}]$	$[1.270, 1.31]$	$[\mathbf{5804}, \mathbf{6400}]$	$[18043, 20169]$
g02	$[1, 1]$	$[1, 1]$	$[0.259, 0.265]$	$[0.260, 0.260]$	$[100, 130]$	$[30, 30]$
g03	$[\mathbf{2286}, \mathbf{3458}]$	$[5630, 7899]$	$[0.217, 0.265]$	$\frac{\mathbf{13}}{\mathbf{30}}[\mathbf{0.280}, \mathbf{0.340}]$	$[240000, 240000]$	$[240000, 240000]$
g04	$[3, 5]$	$[3, 4]$	$[4.004, 4.158]$	$[\mathbf{4.120}, \mathbf{4.250}]$	$[\mathbf{480}, \mathbf{590}]$	$[3225, 4528]$
g05	$[104203, 151126]$	$[\mathbf{27272}, \mathbf{28451}]$	$[0.000, 0.000]$	$[0.000, 0.000]$	$[240000, 240000]$	$[\mathbf{37521}, \mathbf{43125}]$
g06	$[1384, 1735]$	$[\mathbf{1038}, \mathbf{1315}]$	$[3.948, 4.142]$	$\frac{\mathbf{17}}{\mathbf{30}}[\mathbf{4.050}, \mathbf{4.200}]$	$[\mathbf{5590}, \mathbf{6100}]$	$[69251, 147501]$
g07	$[\mathbf{1874}, \mathbf{2185}]$	$[2920, 3261]$	$[\mathbf{1.891}, \mathbf{2.094}]$	$\frac{\mathbf{29}}{\mathbf{30}}[1.300, 1.500]$	$[\mathbf{4800}, \mathbf{5100}]$	$[7383, 12182]$
g08	$[\mathbf{65}, \mathbf{118}]$	$[80, 157]$	$[0.040, 0.049]$	$[0.040, 0.050]$	$[\mathbf{1840}, \mathbf{2070}]$	$[2262, 2504]$
g09	$[98, 165]$	$[83, 162]$	$[\mathbf{2.235}, \mathbf{3.263}]$	$[1.96, 2.84]$	$[\mathbf{1490}, \mathbf{1670}]$	$[3476, 3958]$
g10	$[\mathbf{4144}, \mathbf{5116}]$	$[49468, 113585]$	$[\mathbf{0.470}, \mathbf{0.524}]$	$\frac{\mathbf{25}}{\mathbf{30}}[0.080, 0.190]$	$[\mathbf{9970}, \mathbf{10990}]$	$[42366, 92455]$
g11	$[\mathbf{3062}, \mathbf{6123}]$	$[10462, 65829]$	$[\mathbf{0.081}, \mathbf{0.116}]$	$\frac{\mathbf{27}}{\mathbf{30}}[0.035, 0.074]$	$[240000, 240000]$	$[\mathbf{20262}, \mathbf{72970}]$
g12	$[15, 31]$	$[17, 35]$	$[0.079, 0.104]$	$[0.077, 0.107]$	$[\mathbf{2120}, \mathbf{2460}]$	$[2922, 3129]$
g13	$[212174, 212874]$	$[\mathbf{18414}, \mathbf{19484}]$	$[0.000, 0.000]$	$[0.000, 0.000]$	$[215500, 215560]$	$[\mathbf{27583}, \mathbf{34644}]$

Table 7.8: Confidence intervals for the mean statistics for the three measures (95% level of confidence). A number in **boldface** means a better result. The fraction in the PROGRESS-RATIO measure for the SR indicates the number of independent runs (out of 30) in which feasible solutions were found in the population of the last generation. In the remaining runs, no feasible solutions were found at the end of the process.

vided a "similar" interval for problem g09. Test functions g02, g04 and g12 are discarded for discussion because both approaches found a feasible solution in the initial population.

The estimated mean value for the PROGRESS-RATIO measure obtained by the SMES was higher (better) than the corresponding value obtained by the SR in seven test functions (g01, g03, g06, g07, g09, g10 and g11). The SR obtained better results in problem g04. There were similar results by both approaches in the remaining problems (g02, g05, g08, g12 and g13). Despite a better confidence interval for the progress inside the feasible region by the SR in problems g03 and g06, this approach presented problems to maintain the feasible solutions found up to the last generation of the evolutionary process. In contrast, for problems g05 and g13 none of the approaches could improve the first feasible solution found. This means that for these two problems, both approaches found a value close to the optimum, but could not improve it. For problems g08 and g12, both approaches provided similar estimated low mean values for their progress inside the feasible region. This is because it is easy to reach the global optimum in these problems. On the other hand, for problem g02, the progress is very low since most of the search space is feasible. This suggests that both approaches have problems to improve good solutions previously found. A high dimensionality (30 decision variables) combined with a nonlinear objective function generates a very rugged and complicated feasible region for both approaches.

Finally, we discuss the estimated mean for the ALL-FEASIBLE measure. In problem

g02, both approaches got a fully feasible population in the initial population. In contrast, for problem g03 none of them could have a feasible population in any generation. The estimated mean of evaluations to have a fully feasible population is lower by the SMES in eight test problems (g01, g04, g06, g07, g08, g09, g10 and g12). In the remaining three functions, the SR required less evaluations (on average) to have only feasible solutions (g05, g11 and g13).

Based on the results of the estimated mean for the three performance measures, we can conclude the following for these 13 test functions:

- SMES is able to generate a feasible solution faster than the SR except in problems with more than one nonlinear equality constraint.
- SMES seems to have a better capability to improve the results once the feasible region is reached.
- In presence of more than one nonlinear equality constraint both approaches are unable to improve significantly a feasible solution.
- For problems with a low dimensionality (between 2 and 3 decision variables) and a quadratic objective function, the progress provided by both approaches inside the feasible region is enough as to reach, reasonably fast, the global optimum.
- SMES provided a better improvement inside the feasible region for most of the functions. This is due to its deterministic selection compared with the stochastic selection used by the SR which causes a more emphasized oscillation between feasible and infeasible solutions.
- Due to its emphasis on reaching the feasible region, SMES requires less evaluations to generate a fully feasible population. Therefore, its diversity mechanism is effective when active during most of the evolutionary process, since such mechanism maintains a few recently-generated infeasible solutions close to the boundaries of the feasible region.
- SR tends to have almost always a considerable number of infeasible solutions with a good value of the objective function, regardless of how close they are from the feasible region. In fact, this mechanism seems to be more adequate when solving problems with nonlinear equality constraints where the SR outperforms SMES.

All these conclusions, based on a statistical test, give some insights about the behavior of these two approaches and not only compare them based on their final results.

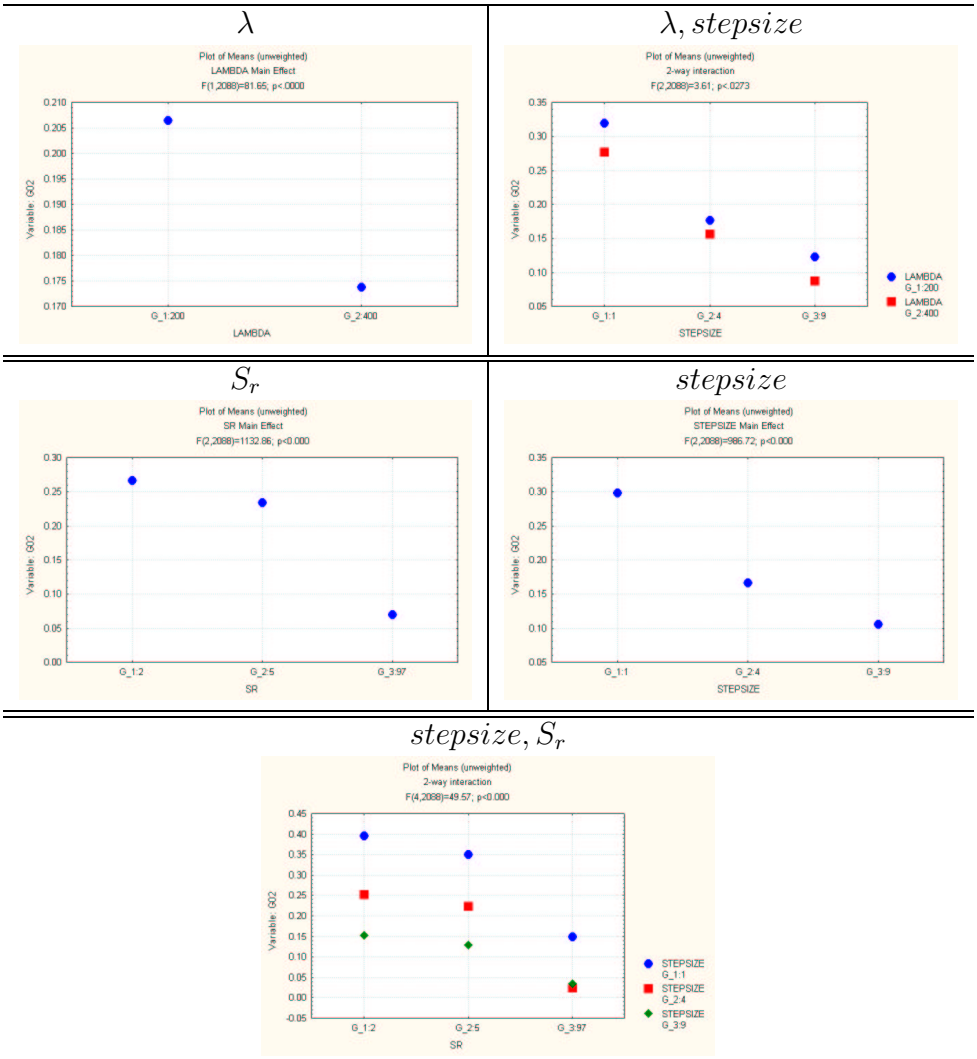


Figure 7.2: Error measured with different parameter values where the ANOVA found significance of the results (alternative hypothesis) for test function g02.

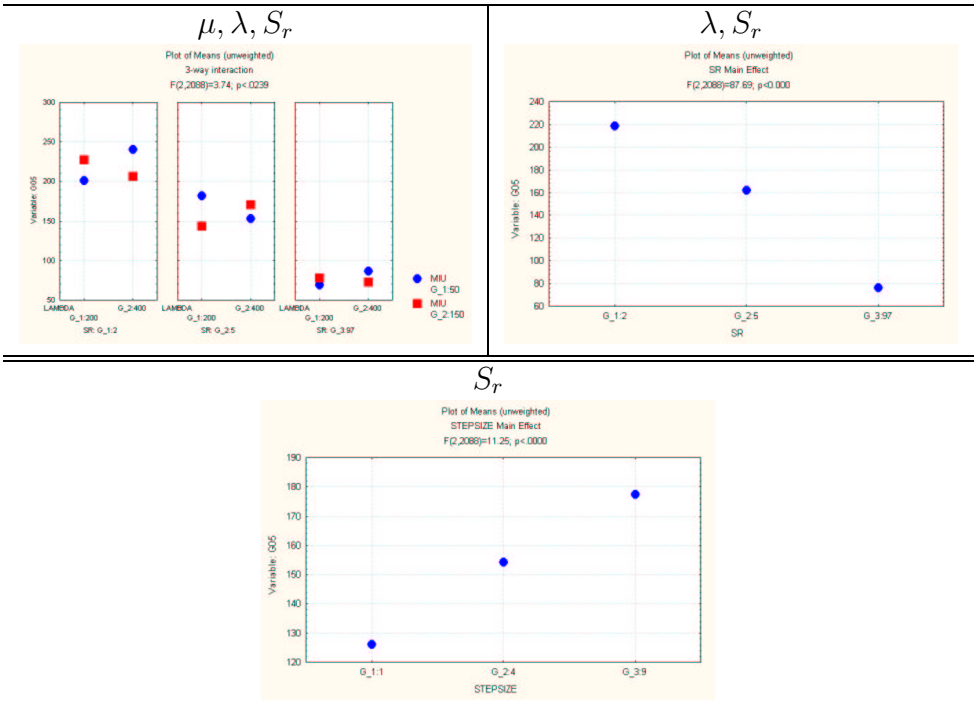


Figure 7.3: Error measured with different parameter values where the ANOVA found significance of the results (alternative hypothesis) for test function g05.

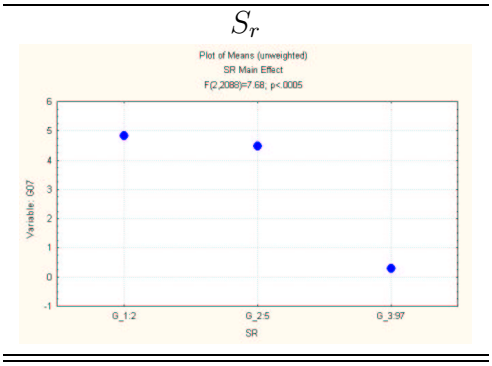


Figure 7.4: Error measured with different parameter values where the ANOVA found significance of the results (alternative hypothesis) for test function g07.

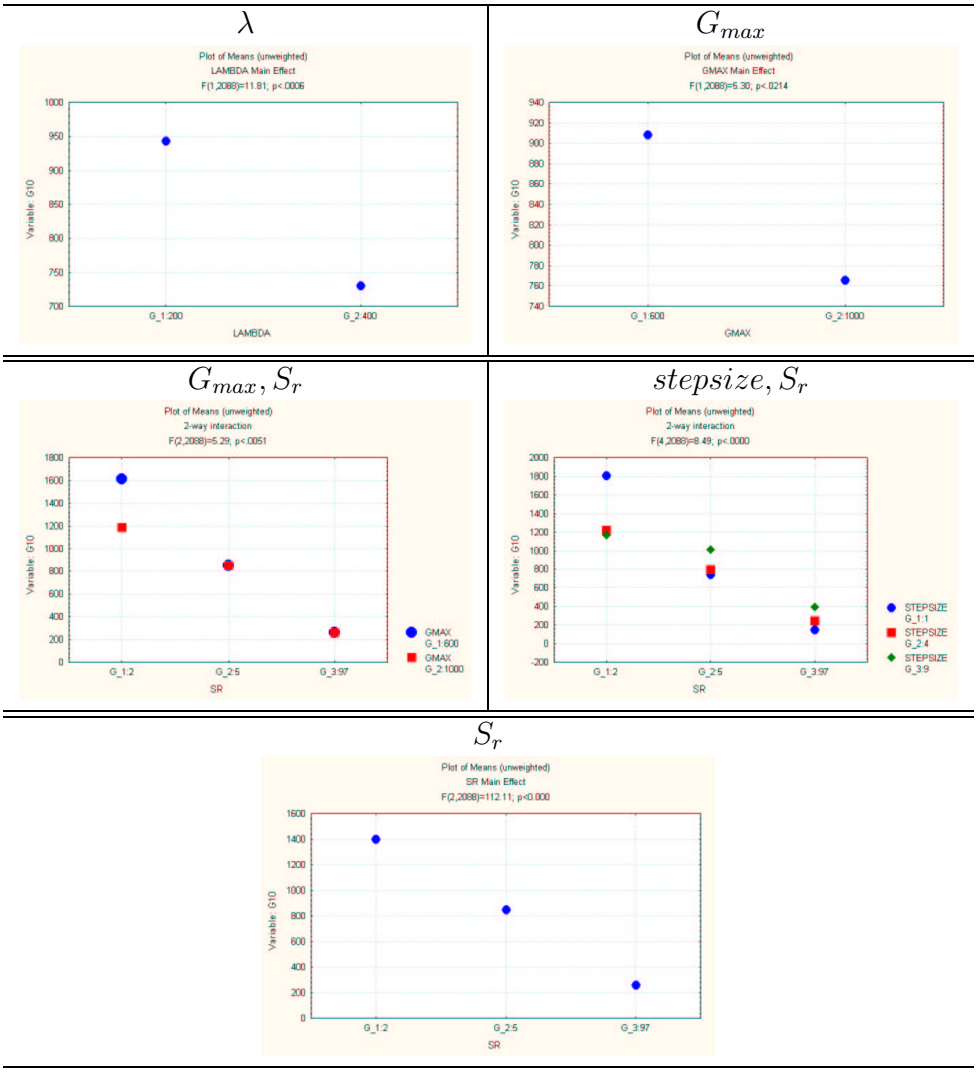


Figure 7.5: Error measured with different parameter values where the ANOVA found significance of the results (alternative hypothesis) for test function g10.

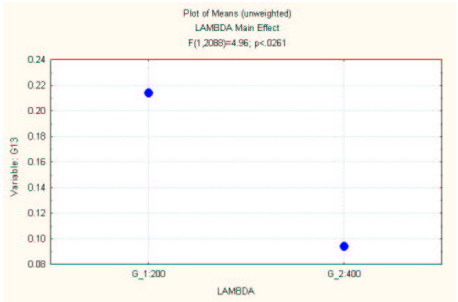
λ


Figure 7.6: Error measured with different parameter values where the ANOVA found significance of the results (alternative hypothesis) for test function g13.

Chapter 8

What Makes a Constrained Optimization Problem Difficult to Solve by The Proposed Approach?

This chapter proposes an approach to empirically find out what features of a problem (which are not fully covered in the most used benchmark to test constraint handling techniques in EAs) decrease the good performance of an evolutionary algorithm. Our study starts by using the $(\mu + \lambda)$-ES discussed in Chapter 6 which has been shown to provide a very competitive performance on the aforementioned benchmark. We introduce 11 new test functions that include characteristics that the current benchmark lacks, like nonlinear equality constraints and a higher dimensionality. The algorithm is tested on them and the results provided are analyzed and discussed.

8.1 Previous Work

The idea of having a set of constrained optimization problems with different characteristics to test evolutionary algorithms was initially proposed by Michalewicz [129, 126]. This work was summarized by Michalewicz & Schoenauer [133] to propose a set of constrained test functions. That set consisted of eleven problems with features such as different types of objective function (linear, quadratic, nonlinear), different types of constraints (linear, nonlinear, equality or inequality) and different dimensionality. Koziel & Michalewicz [110] added one function to the original benchmark. The main feature of this new function is its disjoint feasible region. Runnarson & Yao added another function to the benchmark [162]. This function has three equality constraints (two of them are nonlinear) and the objective

function is also nonlinear. Those two new functions [110, 162] addressed two features that the original benchmark lacked (disjoint feasible regions and a combination of linear and nonlinear equality constraints). The goal of this benchmark is to have a reliable mean to test the quality and robustness of constraint handling techniques used with evolutionary algorithms.

The idea of generating artificial constrained test functions has been almost explored. Michalewicz [130] proposed a test case generator for constrained parameter optimization techniques. This generator allows to generate test problems by varying several features such as: dimensionality, multimodality, number of constraints, connectivity of the feasible region, size of the feasible region with respect to the whole search space and ruggedness of the objective function. This first version had some problems because the generated functions were highly symmetric. Therefore, a new version called TCG-2 was later proposed [165] to solve this drawback and to improve its features. Both versions of the TCG were used to test a constraint-handling mechanism based on a static penalty function. They used a steady-state EA as a search engine. The results obtained suggest that the sources of difficulty for the penalty function approach were a high dimensionality and multimodality of the objective function. For the first TCG, having a disconnected feasible region also affected the performance of the approach. For the TCG-2, the width of peaks of the objective function also decreased the performance of the algorithm. The experiments also showed that for both, TCG and TCG-2, the size of the feasible region with respect to the whole search had no negative influence in the performance of the approach. In addition, for the TCG, the number of constraints and the ruggedness of the objective function did not affect the good quality of the results provided by the approach. Finally, for the TCG-2 the complexity of the function and the number of active constraints caused little impact in the performance of the approach.

8.2 Our Empirical Study

The main motivation of this work is to determine which characteristics of a global nonlinear optimization constrained problem increase the difficulty to be solved by an EA. Such knowledge can help researchers to develop more robust and more reliable EAs that can be used in real-world problems. We hypothesized that the current benchmark lacks of two main features: high dimensionality and a considerable (more than three) number of nonlinear equality constraints. As a second set of features we include the number of nonlinear inequality constraints (more than ten at least), nonlinear objective functions and a disjoint feasible region (only one function with this feature is included in the current benchmark). See Table 5.1 in Chapter 5 to review the details of the current benchmark.

Problem	n	Type of function	ρ	LI	NI	LE	NE
g14	10	nonlinear	0.00%	0	0	3	0
g15	3	quadratic	0.00%	0	0	1	1
g16	5	nonlinear	0.0204%	4	34	0	0
g17	6	nonlinear	0.00%	0	0	0	4
g18	9	quadratic	0.00%	0	12	0	0
g19	15	nonlinear	33.4761%	0	5	0	0
g20	24	linear	0.00%	0	6	2	12
g21	7	linear	0.00%	0	1	0	5
g22	22	linear	0.00%	0	1	8	11
g23	9	linear	0.00%	0	2	3	1
g24	2	linear	79.6556%	0	2	0	0

Table 8.1: Values of ρ for the new 11 test problems. n is the number of decision variables, LI is the number of linear inequality constraints, NI the number of nonlinear inequality constraints, LE is the number of linear equality constraints and NE is the number of nonlinear equality constraints.

Unlike the test case generators previously discussed, we do not intend to provide the user with the best possible EA to be used for a certain type of problem. Instead, we want to detect features that keep an EA from reaching the feasible region or the global optimum, so that we can gain a deeper understanding of the sort of features that make difficult a problem for the approach proposed in this dissertation.

Our experimental design was the following: First, we selected test functions (either artificial or from real world problems) that had at least one of the features mentioned before. We selected seven functions from Himmelblau's book [81] (g14, g15, g16, g17, g18, g19, g20), two are from heat exchanger network problems detailed in [53] and tested in [54] (g21, g22). One more was proposed by Xia [185] (g23) and the last one was taken from Floudas et al.'s Handbook [57] (g24). Selected problems with high dimensionality are: g19, g20 and g22. Test functions with more than three nonlinear equality constraints are: g17, g20, g21 and g22. For the secondary set of features, we chose problems g16 and g18,which have more than ten nonlinear inequality constraints. Problems having a nonlinear objective function are g14, g16, g17 and g19. Finally, a test function having a feasible region consisting on two disconnected sub-regions is g24. For completeness, we also included two functions that seemed to be easy to solve because they have only one nonlinear equality constraint (g15) and a quadratic and linear objective function (g23), respectively. The characteristics of each problem of our new proposed benchmark are summarized in

j	1	2	3	4	5
e_j	-15	-27	-36	-18	-12
c_{1j}	30	-20	-10	32	10
c_{2j}	-20	39	-6	-31	32
c_{3j}	-10	-6	10	-6	-10
c_{4j}	32	-31	-6	39	-20
c_{5j}	-10	32	-10	-20	30
d_j	4	8	10	6	2
a_{1j}	-16	2	0	1	0
a_{2j}	0	-2	0	4	2
a_{3j}	-3.5	0	2	0	0
a_{4j}	0	-2	0	-4	-1
a_{5j}	0	-9	-2	1	-2.8
a_{6j}	2	0	-4	0	0
a_{7j}	-1	-1	-1	-1	-1
a_{8j}	-1	-2	-3	-2	-1
a_{9j}	1	2	3	4	5
a_{10j}	1	1	1	1	1

Table 8.2: Data set for test problem g19

Table 8.1. We also calculated the "ρ" metric [133] for this set of new functions, in order to get an estimate of the ratio between the feasible region an the whole search space.

The details of each functions are presented below:

- **g14** [81]:

  Minimize: $f(\vec{x}) = \sum_{i=1}^{10} x_i \left(c_i - \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$

  subject to:

  $h_1(\vec{x}) = x_1 + 2x_2 + +2x_3 + x_6 + x_{10} - 2 = 0$

  $h_2(\vec{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$

  $h_3(\vec{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$

  where the bounds are $0 \leq x_i \leq 10$ ($i = 1, \ldots, 10$), and $c_1 = -6.089$, $c_2 = -17.164$, $c_3 = -34.054$, $c_4 = -5.914$, $c_5 = -24.721$, $c_6 = -14.986$, $c_7 = -24.1$, $c_8 = -10.708$, $c_9 = -26.662$, $c_{10} = -22.179$,. The best known solution is at $x^* = (0.0350, 0.1142, 0.8306, 0.0012, 0.4887, 0.0005, 0.0209, 0.0157, 0.0289, 0.0751)$ where

$f(x^*) = -47.751$.

- **g15** [81]:
  Minimize: $f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$

  subject to:
  $h_1(\vec{x}) = x_1^2 + x_2^2 + +x_3^2 - 25 = 0$
  $h_2(\vec{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$

  where the bounds are $0 \leq x_i \leq 10$ ($i = 1, \ldots, 10$). The best known solution is at $x^* = (3.512, 0.217, 3.552)$ where $f(x^*) = 961.715$.

- **g16** [81]:
  Maximize: $f(\vec{x}) = 0.0000005843y_{17} - 0.000117y_{14} - 0.1365 - 0.00002358y_{13}$
  $- 0.0000011502y_{16} - 0.0321y_{12} - 0.004324y_5 - 0.0001\frac{c_{15}}{c_{16}} - 37.48\frac{y_2}{c_{12}}$

  subject to:
  $g_1(\vec{x}) = y_4 - \frac{0.28}{0.72}y_5 \geq 0$
  $g_2(\vec{x}) = 1.5x_2 - x_3 \geq 0$
  $g_3(\vec{x}) = 21 - 3496\frac{y_2}{c_{12}} \geq 0$
  $g_4(\vec{x}) = \frac{62{,}212}{c_{17}} - 110.6 - y_1 \geq 0$
  $g_5(\vec{x}) = y_1 - 213.1 \geq 0$
  $g_6(\vec{x}) = 405.23 - y_1 \geq 0$
  $g_7(\vec{x}) = y_2 - 17.505 \geq 0$
  $g_8(\vec{x}) = 1053.6667 - y_2 \geq 0$
  $g_9(\vec{x}) = y_3 - 11.275 \geq 0$
  $g_{10}(\vec{x}) = 35.03 - y_3 \geq 0$
  $g_{11}(\vec{x}) = y_4 - 214.228 \geq 0$
  $g_{12}(\vec{x}) = 665.585 - y_4 \geq 0$
  $g_{13}(\vec{x}) = y_5 - 7.458 \geq 0$
  $g_{14}(\vec{x}) = 584.463 - y_5 \geq 0$
  $g_{15}(\vec{x}) = y_6 - 0.961 \geq 0$
  $g_{16}(\vec{x}) = 265.916 - y_6 \geq 0$
  $g_{17}(\vec{x}) = y_7 - 1.612 \geq 0$
  $g_{18}(\vec{x}) = 7.046 - y_7 \geq 0$
  $g_{19}(\vec{x}) = y_8 - 0.146 \geq 0$
  $g_{20}(\vec{x}) = 0.222 - y_8 \geq 0$

$g_{21}(\vec{x}) = y_9 - 107.99 \geq 0$
$g_{22}(\vec{x}) = 273.366 - y_9 \geq 0$
$g_{23}(\vec{x}) = y_{10} - 922.693 \geq 0$
$g_{24}(\vec{x}) = 1286.105 - y_{10} \geq 0$
$g_{25}(\vec{x}) = y_{11} - 926.832 \geq 0$
$g_{26}(\vec{x}) = 1444.046 - y_{11} \geq 0$
$g_{27}(\vec{x}) = y_{12} - 18.766 \geq 0$
$g_{28}(\vec{x}) = 537.141 - y_{12} \geq 0$
$g_{29}(\vec{x}) = y_{13} - 1072.163 \geq 0$
$g_{30}(\vec{x}) = 3247.039 - y_{13} \geq 0$
$g_{31}(\vec{x}) = y_{14} - 8961.448 \geq 0$
$g_{32}(\vec{x}) = 26844.086 - y_{14} \geq 0$
$g_{33}(\vec{x}) = y_{15} - 0.063 \geq 0$
$g_{34}(\vec{x}) = 0.386 - y_{15} \geq 0$
$g_{35}(\vec{x}) = y_{16} - 71084.33 \geq 0$
$g_{36}(\vec{x}) = 140000 - y_{16} \geq 0$
$g_{37}(\vec{x}) = y_{17} - 2802713 \geq 0$
$g_{38}(\vec{x}) = 12146108 - y_{17} \geq 0$

where:

$y_1 = x_2 + x_3 + 41.6$
$c_1 = 0.024x_4 - 4.62$
$y_2 = \frac{12.5}{c_1} + 12$
$c_2 = 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1$
$c_3 = 0.052x_1 + 78 + 0.002377y_2x_1$
$y_3 = \frac{c_2}{c_3}$
$y_4 = 19y_3$
$c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2}$
$c_5 = 100x_2$
$c_6 = x_1 - y_3 - y_4$
$c_7 = 0.950 - \frac{c_4}{c_5}$
$y_5 = c_6c_7$
$y_6 = x_1 - y_5 - y_4 - y_3$
$c_8 = (y_5 + y_4)0.995$
$y_7 = \frac{c_8}{y_1}$
$y_8 = \frac{c_8}{3798}$

$c_9 = y_7 - \frac{0.0663y_7}{y_8} - 0.3153$
$y_9 = \frac{96.82}{c_9} + 0.321y_1$
$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$
$y_{11} = 1.71x_1 - 0.452y_4 + 0.580y_3$
$c_{10} = \frac{12.3}{752.3}$
$c_{11} = (1.75y_2)(0.995x_1)$
$c_{12} = 0.995y_{10} + 1998$
$y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$
$y_{13} = c_{12} - 1.75y_2$
$y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \frac{146.312}{y_9+x_5}$
$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$
$y_{15} = \frac{y_{13}}{c_{13}}$
$y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13}$
$c_{14} = 2324y_{10} - 28740000y_2$
$y_{17} = 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}}$
$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$
$c_{16} = 1.104 - 0.72y_{15}$
$c_{17} = y_9 + x_5$

and where the bounds are $704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$ and $25 \leq x_5 \leq 84.1988$. The best known solution is at $x^* = (705.06, 68.6, 102.9, 282.341, 35.627)$ where $f(x^*) = 1.905$.

- **g17** [81]:
  Minimize: $f(\vec{x}) = f(x_1) + f(x_2)$

  subject to:
  $$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$
  $$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$$
  $h_1(\vec{x}) = x_1 = 300 - \frac{x_3x_4}{131.078}\cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\cos(1.47588)$
  $h_2(\vec{x}) = x_2 = -\frac{x_3x_4}{131.078}\cos((1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\cos(1.47588)$
  $h_3(\vec{x}) = x_5 = -\frac{x_3x_4}{131.078}\sin((1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\sin(1.47588)$
  $h_4(\vec{x}) = 200 - \frac{x_3x_4}{131.078}\sin((1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\sin(1.47588)$

where the bounds are $0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$ and $0 \leq x_6 \leq 0.5236$. The best known solution is at $x^* = (107.81, 196.32, 373.83, 420, 21.31, 0.153)$ where $f(x^*) = 8927.5888$.

- **g18** [81]:
  Maximize: $f(\vec{x}) = 0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$

  subject to:
  $g_1(\vec{x}) = 1 - x_3^2 - x_4^2 \geq 0$
  $g_2(\vec{x}) = 1 - x_9^2 \geq 0$
  $g_3(\vec{x}) = 1 - x_5^2 - x_6^2 \geq 0$
  $g_4(\vec{x}) = 1 - x_1^2 - (x_2 - x_9)^2 \geq 0$
  $g_5(\vec{x}) = 1 - (x_1 - x_5)^2 - (x_2 - x_6)^2 \geq 0$
  $g_6(\vec{x}) = 1 - (x_1 - x_7)^2 - (x_2 - x_8)^2 \geq 0$
  $g_7(\vec{x}) = 1 - (x_3 - x_5)^2 - (x_4 - x_6)^2 \geq 0$
  $g_8(\vec{x}) = 1 - (x_3 - x_7)^2 - (x_4 - x_8)^2 \geq 0$
  $g_9(\vec{x}) = 1 - x_7^2 - (x_8 - x_9)^2 \geq 0$
  $g_9(\vec{x}) = x_1x_4 - x_2x_3 \geq 0$
  $g_{10}(\vec{x}) = x_3x_9 \geq 0$
  $g_{11}(\vec{x}) = -x_5x_9 \geq 0$
  $g_{12}(\vec{x}) = x_5x_8 - x_6x_7 \geq 0$

  where the bounds are $-10 \leq x_i \leq 10$ ($i = 1, \ldots, 8$) and $0 \leq x_9 \leq 20$. The best known solution is at $x^* = (0.9971, -0.0758, 0.5530, 0.8331, 0.9981, -0.0623, 0.5642, 0.8256, 0.0000024)$ where $f(x^*) = 0.8660$.

- **g19** [81]:
  Maximize: $f(\vec{x}) = \sum_{i=1}^{10} b_i x_i - \sum_{j=1}^{5} \sum_{i=1}^{5} c_{ij} x_{(10+i)} x_{(10+j)} - 2 \sum_{j=1}^{5} d_j x_{(10+j)}^3$

  subject to:
  $g_j(\vec{x}) = 2 \sum_{i=1}^{5} c_{ij} x_{(10+i)} + 3 d_j x_{(10+j)}^2 + e_j - \sum_{i=1}^{10} a_{ij} x_i \geq 0 \quad j = 1, \ldots, 5$

  where $\vec{b} = [-40, -2, -.25, -4, -4, -1, -40, -60, 5, 1]$ and the remaining data is on Table 8.2. The bounds are $0 \leq x_i \leq 10$ ($i = 1, \ldots, 15$). The best known solution is at $x^* = (0, 0, 5.1740, 0, 3.0611, 11.8395, 0, 0, 0.1039, 0, 0.3, 0.3335, 0.4, 0.4283, 0.2240)$ where $f(x^*) = -32.386$.

- **g20** [81]:
  Minimize: $f(\vec{x}) = \sum_{i=1}^{24} a_i x_i$

  subject to:
  $h_i(\vec{x}) = \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j+13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0 \quad i = 1, \ldots, 12$
  $h_{13}(\vec{x}) = \sum_{i=1}^{24} x_i - 1 = 0$
  $h_{14}(\vec{x}) = \sum_{i=1}^{12} \frac{x_i}{d_i} + f \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0$
  $g_i(\vec{x}) = \frac{-(x_i + x_{(i+12)})}{\sum_{j=1}^{24} x_j + e_i} \geq 0 \qquad i = 1, 2, 3$
  $g_i(\vec{x}) = \frac{-(x_{(i+3)} + x_{(i+15)})}{\sum_{j=1}^{24} x_j + e_i} \geq 0 \qquad i = 4, 5, 6$

  where $f = (0.7302)(530)(\frac{14.7}{40})$ and the data set is detailed on Table 8.3. The bounds are $0 \leq x_i \leq 10$ ($i = 1, \ldots, 24$). The best known solution is at $x^* = (9.53E-7, 0, 4.21eE-3, 1.039E-4, 0, 0, 2.072E-1, 5.979E-1, 1.298E-1, 3.35E-2, 1.711E{-}2, 8.827E{-}3, 4.657E{-}10, 0, 0, 0, 0, 0, 2.868E{-}4, 1.193E{-}3, 8.332E{-}5, 1.239E-4, 2.07E-5, 1.829E-5)$ where $f(x^*) = 0.09670$.

- **g21** [53]:
  Minimize: $f(\vec{x}) = x_1$

  subject to:
  $g_1(\vec{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$
  $h_1(\vec{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$
  $h_2(\vec{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$
  $h_3(\vec{x}) = -x_5 + \ln(-x_4 + 900) = 0$
  $h_4(\vec{x}) = -x_6 + \ln(x_4 + 300) = 0$
  $h_5(\vec{x}) = -x_7 + \ln(-2x_4 + 700) = 0$

  where the bounds are $0 \leq x_1 \leq 1000$, $0 \leq x_2, x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$ and $4.5 \leq x_7 \leq 6.25$. The best known solution is at $x^* = (193.7783493, 0, 17.3272116, 100.0156586, 6.684592154, 5.991503693, 6.214545462)$ where $f(x^*) = 193.7783493$.

- **g22** [53]:
  Minimize: $f(\vec{x}) = x_1$

subject to:
$g_1(\vec{x}) = -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0$
$h_1(\vec{x}) = x_5 - 100000x_8 + 1 \times 10^7 = 0$
$h_2(\vec{x}) = x_6 + 100000x_8 - 100000x_9 = 0$
$h_3(\vec{x}) = x_7 + 100000x_9 - 5 \times 10^7 = 0$
$h_4(\vec{x}) = x_5 + 100000x_{10} - 3.3 \times 10^7 = 0$
$h_5(\vec{x}) = x_6 + 100000x_{11} - 4.4 \times 10^7 = 0$
$h_6(\vec{x}) = x_7 + 100000x_{12} - 6.6 \times 10^7 = 0$
$h_7(\vec{x}) = x_5 - 120x_2x_{13} = 0$
$h_8(\vec{x}) = x_6 - 80x_3x_{14} = 0$
$h_9(\vec{x}) = x_7 - 40x_4x_{15} = 0$
$h_{10}(\vec{x}) = x_8 - x_{11} + x_{16} = 0$
$h_{11}(\vec{x}) = x_9 - x_{12} + x_{17} = 0$
$h_{12}(\vec{x}) = -x_{18} + \ln(x_{10} - 100) = 0$
$h_{13}(\vec{x}) = -x_{19} + \ln(-x_8 + 300) = 0$
$h_{14}(\vec{x}) = -x_{20} + \ln(x_{16}) = 0$
$h_{15}(\vec{x}) = -x_{21} + \ln(-x_9 + 400) = 0$
$h_{16}(\vec{x}) = -x_{22} + \ln(x_{17}) = 0$
$h_{18}(\vec{x}) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$
$h_{19}(\vec{x}) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$
$h_{20}(\vec{x}) = x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0$

where the bounds are $0 \leq x_1 \leq 20000$, $0 \leq x_2, x_3, x_4 \leq 1 \times 10^6$, $0 \leq x_5, x_6, x_7 \leq 4 \times 10^7$, $100 \leq x_8 \leq 299.99$, $100 \leq x_9 \leq 399.99$, $100.01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_{13}, x_{14}, x_{15} \leq 500$, $0.01 \leq x_{16} \leq 300$, $0.01 \leq x_{17} \leq 400$, $-4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$. The best known solution is at $x^* = (12812.5, 722.1602494, 8628.371755, 2193.749851, 9951396.436, 18846563.16, 11202040.4, 199.5139644, 387.979596, 114.8336587, 27.30318607, 127.6585887, 52.020404, 160, 4.871266214, 4.610018769, 3.951636026, 2.486605539, 5.075173815)$ where $f(x^*) = 12812.5$.

- **g23** [185]:
Minimize: $f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$

subject to:
$h_1(\vec{x}) = x_1 + x_2 - x_3 - x_4 = 0$

$h_2(\vec{x}) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$
$h_3(\vec{x}) = x_3 + x_6 - x_5 = 0$
$h_4(\vec{x}) = x_4 + x_7 - x_8 = 0$
$g_1(\vec{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$
$g_2(\vec{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$

where the bounds are $0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ and $0.01 \leq x_9 \leq 0.03$. The best known solution has a objective function value of $f(x^*) = -400.0551$

- **g24** [57]:
  Minimize: $f(\vec{x}) = -x_1 - x_2$

  subject to:
  $g_1(\vec{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 \leq 0$
  $g_2(\vec{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$

  where the bounds are $0 \leq x_1 \leq 3$ and $0 \leq x_2 \leq 4$. The feasible global minimum is at $x^* = (2.3295, 3.17846)$ where $f(x^*) = -5.50796$.

Our next step was to solve the new set of 11 problems using our Simple Multimembered Evolution Strategy (SMES) described in Chapter 6, adopting the exact same parameters previously defined to solve the 13 test functions taken from [162].

Most of the previous work on constraint-handling techniques relates to the benchmark proposed in [162]. However, we know (from the No Free Lunch Theorems for search [183]) that using such a limited set of functions does not guarantee, in any way, that an algorithm that performs well on them will necessarily be competitive in a different set of problems. This motivated us to identify new test functions to validate our algorithm. What we expected to find was functions in which our approach (which was found to be highly competitive in the traditional benchmark from [162]) did not exhibit a good performance. We expected to identify in such functions some features that could be associated with sources of difficulty for the constraint-handling mechanism adopted in our approach.

We performed 30 independent runs for each test function. The learning rates values were calculated using the formulas proposed by Schwefel [170] (where n is the number of decision variables of the problem): $\tau = (\sqrt{2\sqrt{n}})^{-1}$ and $\tau' = (\sqrt{2n})^{-1}$. In the experimentation process we used the same parameters reported in Chapter 6 which are the following:

i	a_i	b_i	c_i	d_i	e_i
1	0.0693	44.094	123.7	31.244	0.1
2	0.0577	58.12	31.7	36.12	0.3
3	0.05	58.12	45.7	34.784	0.4
4	0.2	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.1	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	
10	0.18	82.507	17.7	64.517	
11	0.1	46.07	0.85	49.4	
12	0.9	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.2	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.1	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.1	46.07			
24	0.09	60.097			

Table 8.3: Data set for test problem g20

Problem	Statistical Results of SMES for the new 11 Problems					
	Optimal	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g14	-47.656000	-47.534851	-47.367386	-47.385674	-47.053207	0.133386
g15	961.715000	**961.698120**	963.921753	964.058350	967.787354	1.791314
g16	1.905000	**1.905155**	**1.905155**	**1.905155**	**1.905155**	0.000000
g17	8927.588800	$*$8890.182617	$*$8954.136458	$*$8948.685547	$*$9163.676758	40.826101
g18	0.866000	**0.866002**	0.715698	0.673722	0.647570	0.081901
g19	-32.386000	-34.222656	-37.208255	-36.429800	-41.251328	2.102102
g20	0.096700	$*$0.211364	$*$0.251130	$*$0.252439	$*$0.304414	0.023365
g21	193.778349	$*$347.980927	$*$678.392445	$*$711.847260	$*$985.782166	158.493960
g22	12812.500000	$*$2340.616699	$*$9438.254972	$*$9968.156250	$*$17671.535156	4360.887012
g23	-400.0551	$*-1470.152588$	$*-363.508270$	$*-333.251541$	$*$177.252640	316.115639
g24	-5.507960	$\mathbf{-5.508013}$	$\mathbf{-5.508011}$	$\mathbf{-5.508013}$	$\mathbf{-5.507959}$	0.000010

Table 8.4: Statistical results for SMES (the $(\mu + \lambda)$-ES from Chapter 6) with the 11 new test functions. "*" means infeasible. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

$(100 + 300)$-ES, number of generations = 800, number of objective function evaluations = $240,000$. To deal with equality constraints, the dynamic mechanism described in Chapter 6 was adopted. The initial ϵ_0 was set to 0.001.

8.3 Results and Discussion

The statistical results of our SMES for the new set of 11 functions are presented in Table 8.4.

We will now proceed to discuss the results obtained. SMES had no difficulties to solve problem g16 despite its low value of ρ. g16 involves a considerable number of nonlinear inequalities (34) combined with 4 linear inequality constraints and a nonlinear objective function. The problem has a low dimensionality (5 decision variables). SMES also solved quite well problems g14 and g18. In both problems the algorithm found the optimum reported in Himmelblau's book. Problem g14 has a nonlinear objective function and 3 linear equality constraints. Problem g18 has a quadratic objective function and 12 nonlinear inequality constraints. Both problems have a value of $\rho = 0\%$ and a relatively high dimensionality (10 and 9 decision variables, respectively).

A value close to the optimum and a low value of the standard deviation were found by SMES for problem g19. The algorithm was less robust in this problem with a nonlinear objective function and 5 nonlinear inequality constraints. It is interesting to note that, despite its ρ value of 33.4761% (which means a large feasible region), a low number of constraints (5) and no equality constraints, SMES could not find the best solution reported.

Note however that this problem has 15 decision variables.

For problem g15, the best objective function value found by SMES is better than the solution reported by Himmelblau, but it is slightly infeasible. Also, in about 35% of the 30 runs, SMES could not find feasible solutions. This problem has one linear and one nonlinear equality constraints. The objective function is quadratic and the ρ value is 0% (it is very hard to generate a feasible solution) . Note that this problem has only 3 decision variables.

Problems g17, g20, g21 and g22 have one common aspect: they have more nonlinear equality constraints than any other problem (4, 12, 5 and 11 linear equality constraints, respectively). In those problems, SMES could not find feasible solutions in any single run (we present the statistics of the 30 independent runs despite no feasible solutions were found in any of them in order to show the behavior of SMES in those functions). The dimensionality is different for each of these four problems (6, 24, 7 and 22, respectively). For three problems, the objective function is linear (g20, g21 and g22). Only g17 has a nonlinear objective function. All that suggests that the difficulty comes from the number of nonlinear equality constraints. It is worth reminding that none of the 13 original test functions has more than 3 nonlinear equality constraints. Furthermore, no problem with equality constraints has more than 5 decision variables.

The obtained results suggest that the combination of an increasing dimensionality and a high number of nonlinear equality constraints make a problem more difficult to solve by SMES. In fact, just one feature is enough to give some problems like in function g19 which does not have equality constraints, but has 15 decision variables. Another example is problem g17, which has a low dimensionality (6 decision variables) but 4 nonlinear equality constraints. The performance degrades the most when a problem combines nonlinear equality constraints and a high dimensionality, as in problems g20 and g22.

It is important to mention that the sum of constraint violation of the final results for problems g17, g20, g21 and g23 is not high. For problem g23 the best result was far from the feasible region.

There are two test problems that only have one nonlinear equality constraint: g15 and g23 which have a quadratic and linear objective function, respectively. The dimensionality is different (3 and 9 decision variables). Both of them have a very small feasible region compared with the whole search space. Furthermore, both have linear equality constraints (1 for g15 and 3 for g23 which has 2 nonlinear inequality constraints). For g23, the dimensionality coupled with the combination of linear and nonlinear equality constraints and nonlinear inequality constraints should influence SMES to keep it from reaching the feasible region. The results obtained with g15 seem to suggest that dimensionality also plays a crucial role on the performance of the algorithm.

Finally, Problem g24 with a disjoint and a very large feasible region but with a low

dimensionality of 2 represented no problem for SMES.

To summarize, the overall results suggest that the two main factors that affect the performance of our EA are the dimensionality (like Michalewicz & Schmidt concluded for the static penalty function approach [130, 165]) and the increasing number of nonlinear equality constraints. The factors that do not seem to decrease the performance of our EA are a high number of inequality constraints (even nonlinear), and, remarkably, the type of objective function. For some problems, including a linear objective function, the problems resulted difficult to solve (even reaching the feasible region). Finally, disjoint feasible regions with a considerable large size with respect to the search space and a low dimensionality do not seem to be difficult to reach for our EA. We need to test other functions with disjoint feasible regions but with a higher dimensionality and with nonlinear constraints to get more insights about this issue.

As a final discussion, we wanted to determine why the SMES failed to provide a competitive performance, but in this case, regarding SMES's own mechanisms. In the experiments in the current Chapter, SMES presented the poorest results in presence of equality constraints (mainly nonlinear). Also, in the experiments presented in Chapter 6, SMES required different initial values for the dynamic mechanism to deal with the small tolerance for equality constraints. Furthermore, in Chapter 7, the Stochastic Ranking outperformed SMES in problems with equality constraints. Based on those results, we argue that this mechanism to dynamically decrease the tolerance for equality constraints must be improved in order to avoid the user to define an initial tolerance value. Instead, SMES must define an initial value and also the algorithm must be capable of self-adapting it. In this way, the performance of SMES when dealing with equality constraints should be improved.

This small study is far from being conclusive, but it gives some insights about the factors that keep our EA from finding good results when solving constrained optimization problems. We hope that this information may help to develop more robust and general EAs.

Chapter 9

Global Convergence Properties of The Proposed Approach

The foundations of the theory of evolution strategies were laid by Rechenberg in [155]. Theory on evolution strategies has focused on two aspects [17]: (1) convergence velocity and (2) convergence reliability. Convergence velocity refers to the speed of the algorithm to approach a local optimum, and therefore it provides insight into the local behavior of an evolution strategy. In contrast, convergence reliability concentrates of proving that the algorithm is capable of finding the global optimum. The convergence velocity analysis requires strong simplifications related to the form of the objective function to be optimized (i.e. it requires convex functions). On the other hand, the analysis of global convergence with probability one yields a result for $t \rightarrow \infty$ (i.e. an asymptotical result as the running time of the algorithm goes to infinity) independently of the form of the objective function. The work in this Chapter focuses on the last approach.

Rudolph [161], proposed the conditions for an algorithm to converge to the global optimum, assuming infinite time and finite population size. The aim of this work is to prove that our SMES approach fits in Rudolph's scheme [161] and, therefore, SMES can converge to the feasible global optimum.

9.1 Basic Definitions

As detailed in Chapter 3, a global minimum is defined as follows:

Definition 11 (Global minimum): *Let $f(\vec{x}) \subset \mathbb{R}$ a function defined on a set $S \subset \mathbb{R}^l$ and let $F \subset S$ the feasible region of S. This function attains its feasible global minimum at a*

point $\vec{x}^* \in F$ if and only if: $f^* = f(\vec{x}^*) \leq f(\vec{x})$ for all $\vec{x} \in F$. □

Definition 12 (Indicator function):

$$1_A(x) = \begin{cases} 0 & \text{if } x \notin A \\ 1 & \text{if } x \in A \end{cases}$$

□

Definition 13 ($\sigma - algebra$): *A set $\mathcal{A} \neq \emptyset$, whose elements are subsets of a set Ω, is called a $\sigma - algebra$ if:*

(i) $A \in \mathcal{A} \Rightarrow A^c \in \mathcal{A}$, where $A^c = \Omega - A$ denotes the complement of set A, and

(ii) $\bigcup_{n=1}^{\infty} A_n \in \mathcal{A}$ for each sequence $(A_n)_{n \in \mathbb{N}} \subset \mathcal{A}$.

□

Definition 14 (Measurable space): *The pair $(\Omega, \mathcal{A})$, where $\mathcal{A}$ is a $\sigma - algebra$ of subsets of Ω, is called* **measurable space**. *A function ν that assigns a number $\nu(A) \in [0, \infty]$ to each subset $A \in \mathcal{A}$, is called a* **measure** *on $(\Omega, \mathcal{A})$ if $\nu(\emptyset) = 0$ and $\nu(\cup_{n=1}^{\infty} A_n) = \sum_{n=1}^{\infty} \nu(A_n)$ for each sequence $(A_i)_{i \in \mathbb{N}}$ of pairwise disjoint sets of $\mathcal{A}$. The triplet $(\Omega, \mathcal{A}, \nu)$ is called a* **measure space**. □

Definition 15 (Probability measure, event, sample space, probability of an event, probability space, event with probability 1): *Let $(\Omega, \mathcal{A})$ be a measurable space and P be a measure on $(\Omega, \mathcal{A})$ with $\mathsf{P}(\Omega) = 1$. Then P is termed a* **probability measure**, *the sets $A \in \mathcal{A}$ are called* **events**, *Ω is the* **sample space**, *the number $\mathsf{P}(A)$ with $A \in \mathcal{A}$ is called the* **probability of an event A**, *and the triplet $(\Omega, \mathcal{A}, \mathsf{P})$ is termed a* **probability space**. *If $\mathsf{P}(A) = 1$ for some event A, then the event is said to occur* **with probability 1**. □

Definition 16 (Measurable function): *Let $(\Omega, \mathcal{A})$ and $(\Omega', \mathcal{A}')$ be measurable spaces. The function $X : \Omega \to \Omega'$ is called a* **measurable function** *if $X^{-1}(A') \in \mathcal{A}$ for all $A' \in \mathcal{A}'$.* □

Definition 17 (Random variable, probability distribution): *Let $(\Omega, \mathcal{A}, \mathsf{P})$ a probability space and $(\Omega', \mathcal{A}')$ be a measurable space. A measurable function $X : \Omega \to \Omega'$ is called a* **random variable**. *The function $\mathsf{P}_X : \mathcal{A}' \to [0, 1] \subset \mathbb{R}$ with $\mathsf{P}_X(A') := \mathsf{P}(X^{-1}(A'))$ for all $A' \in \mathcal{A}'$ is called the* **probability distribution** *of X.* □

If a random variable only takes integer values, either from a finite or from an infinite set of numerable values we say it is a random discrete variable. On the other hand, if theoretically, the variable can take all values within an interval in $\mathbb{R}$, we say that this random variable is continuous. In this Chapter, we only refer to continuous random variables.

Definition 18 (Markov chain): *If $\mathcal{S} \neq \emptyset$ is a finite set and $\{X_t : t \in \mathbb{N}\}$ is a sequence of random variables with values in $\mathcal{S}$ with the feature that:*

$$\begin{gathered}\mathsf{P}\{X_{t+1} = j | X_t = i, X_{t-1} = i_{t-1}, ..., X_0 = i_0\} = \\ \mathsf{P}\{X_{t+1} = j | X_t = i\} =: p_{ij}\end{gathered}$$

for all $t \geq 0$ and for all $i, j \in \mathcal{S}$, then, the sequence $\{X_t : t \in \mathbb{N}\}$ is called a **finite Markov chain** *with state space $\mathcal{S}$.*

The number p_{ij} is called a **transition probability** *from state i to state j in one step. If we are supposing that these probabilities are independent of $t \in \mathbb{N}$, we state that the chain is* **homogeneous**. □

Definition 19 (Markovian kernel): *A Markovian kernel of a Markov chain $\{X_t : t \in \mathbb{N}\}$ with values in the set S, denoted as $K_X(x, y)$, is defined as $K_X(x, y) := p_{xy}$ for all $x, y \in S$. Let $A \subset S$, if the random variable X is discrete then: $K(x, A) = \sum_{y \in A} K(x, y)$. If the variable X is continuous , then: $K(x, A) = \int_{y \in A} K(x, dy)$.* □

Definition 20 (Level set and lower level set): *Let $f : \mathbb{R}^l \to \mathbb{R}$. The sets $L_a = \{x \in \mathbb{R}^l : f(x) = a\}$ and $H_a = \{x \in \mathbb{R}^l : f(x) \leq a\}$ are termed the* **level set** *and the* **lower level set** *of f at level $a \in \mathbb{R}$, respectively. The* **success set** *$G(x) = H_{f(x)}$ of $x \in \mathbb{R}^l$ is equivalent to the lower level set of f at level $f(x)$.* □

Definition 21 (Convergence): *Let X be a random variable and (X_n) a sequence of random variables defined on a probability space $(\Omega, \mathcal{A}, \mathsf{P})$. Then (X_n) is said*

- *to converge completely to X, denoted as $X_n \xrightarrow{c} X$, if for any $\epsilon > 0$*

$$\lim_{n \to \infty} \sum_{i=1}^{n} \mathsf{P}\{|X_i - X| > \epsilon\} < \infty;$$

- *to converge almost surely to X, denoted as $X_n \xrightarrow{a.s.} X$, if*

$$\mathsf{P}\left\{\lim_{n \to \infty} |X_n - X| = 0\right\} = 1;$$

- *to converge in probability to X, denoted as $X_n \xrightarrow{P} X$, if for any $\epsilon > 0$*

$$\lim_{n \to \infty} P\{|X_n - X| < \epsilon\} = 1.$$

□

9.2 Global Convergence of SMES

The main processes of SMES, which are relevant for this study, are shown in Figure 9.1 (the details of the approach were given in Chapter 6).

```
Begin
    Create M_0 random solutions
    t=0
    While t < T_max Do
        L_t = combined_recombination(M_t)
        L_t = mutate(L_t)
        M_{t+1} = "+"_selection_from(M_t + L_t)
        t=t+1
    End While
End
```

Figure 9.1: Pseudocode of SMES. M is the population of μ parents, L is the population of λ offspring and t is the iteration counter.

As can be noted, our SMES is really a typical evolution strategy. Its only meaningful difference, with respect to a general evolution strategy, is the combination of two well-known recombination operators (note that we use Gaussian mutation and the "+" replacement operators usually adopted in an ES). This difference does not keep SMES from being a standard ES. What we want to do is to demonstrate how our approach satisfies the theorem proposed by Rudolph [161], which proves that an EA with characteristics similar to SMES is able to converge to the feasible global optimum of a given arbitrary constrained objective function.

The following ideas were proposed by Rudolph [161].

The motivation to model an evolutionary algorithm using a Markov chain is because of the fact that a new population (at time t) only depends of the state of the precedent population (at time $t-1$).

Evolutionary algorithms can be modeled as homogeneous Markov chains. Then, an appropriate state space E and the probabilistic behavior of the evolutionary operators must be expressed in terms of transition probabilities over E.

In this way, if I is the space representing admissible instances of a solution, we define the state space $E = I^{\mu}$, where μ is the number of individuals in the population. Besides, it will be assumed that $I = \mathbb{R}^l$, then $E = (\mathbb{R}^l)^{\mu} = \mathbb{R}^{l \cdot \mu}$.

Let $b : E \rightarrow \mathbb{R}^*$, where $\mathbb{R}^* = \mathbb{R} \bigcup \{-\infty, \infty\}$, be the map that extracts the best objective function value of a feasible individual within a population. A populations x with no feasible solutions are assigned a value of $b(x) = \infty$ Then $B(x) = \{y \in E : b(y) \leq b(x)\}$ denotes the set of populations (states) that are as good as (or better than) population $x \in E$.

Let K_{cms} denotes the product kernel describing the crossover, mutation and selection operations of our SMES. The Markovian kernel of the entire SMES is:

$$\mathrm{K}_{cms}(x, A) = \int_A \mathrm{K}_{cms}(x, dy), \quad A \subset E \tag{9.1}$$

As E is a continuous space, we have to use the $\int$ operator. However, we need to restrict the Markovian kernel in Equation 9.1 for two reasons: (1) because we want to prove that our SMES converges to the global optimum of a given problem, and (2) because of the limitation of using an approach which runs on a computer. Therefore, we restrict the Markovian kernel to the set $A_\epsilon = \{x \in E : b(x) \leq f^* + \epsilon\}$ instead of the general set $A \in E$. Besides, $A_\epsilon \subset B(x)$ then $x \notin A_\epsilon$, $A_\epsilon \cap B(x) = A_\epsilon$.

First of all, we need to state that the EA is able to generate the global optimum with a positive probability. Furthermore, if at some generation, the corresponding population indeed contains the global optimum, it will remain in the population forever (elitism):

LEMMA 8.1

Let $A_\epsilon = \{x \in E : b(x) < f^* + \epsilon\}$ with $\epsilon > 0$. If $\mathrm{K}_{cms}(x, A_\epsilon) \geq \delta > 0$ for all $x \in A_\epsilon^c = E \setminus A_\epsilon$ and $\mathrm{K}_{cms}(x, A_\epsilon) = 1$ for $x \in A_\epsilon$ then $\mathrm{K}_{cms}^{(t)}(x, A_\epsilon) \geq 1 - (1-\delta)^t$ for $t \geq 1$.

Proof by induction:

Let $t = 1$. Then $\mathrm{K}_{cms}^{(1)}(x, A_\epsilon) = \mathrm{K}_{cms}(x, A_\epsilon) \geq 1 - (1-\delta)^1 = \delta$ and the hypothesis is true for $t = 1$. Now, assume that the hypothesis is true for $t > 1$. First, note that $\mathrm{K}_{cms}^{(t)}(x, A_\epsilon) = 1$ for all $t \geq 1$ if $x \in A_\epsilon$. This will be used to obtain (A). By choosing an arbitrary $x \in E$ one obtains:

$$
\begin{aligned}
\mathrm{K}_{cms}^{(t+1)}(x, A_\epsilon) &= \int_E \mathrm{K}_{cms}^{(t)}(y, A_\epsilon)\mathrm{K}_{cms}(x, dy) \\
&= \int_{A_\epsilon} \mathrm{K}_{cms}^{(t)}(y, A_\epsilon)\mathrm{K}_{cms}(x, dy) + \int_{A_\epsilon^c} \mathrm{K}_{cms}^{(t)}(y, A_\epsilon)\mathrm{K}_{cms}(x, dy) \\
&= \int_{A_\epsilon} \mathrm{K}_{cms}(x, dy) + \int_{A_\epsilon^c} \mathrm{K}_{cms}^{(t)}(y, A_\epsilon)\mathrm{K}_{cms}(x, dy) \qquad \text{(A)} \\
&= \mathrm{K}_{cms}(x, A_\epsilon) + \int_{A_\epsilon^c} \mathrm{K}_{cms}^{(t)}(y, A_\epsilon)\mathrm{K}_{cms}(x, dy) \\
&\geq \mathrm{K}_{cms}(x, A_\epsilon) + [1 - (1 - \delta)^t] \int_{A_\epsilon^c} \mathrm{K}_{cms}(x, dy) \qquad \text{(by hypothesis)} \\
&= \mathrm{K}_{cms}(x, A_\epsilon) + [1 - (1 - \delta)^t]\mathrm{K}_{cms}(x, A_\epsilon^c) \\
&= \mathrm{K}_{cms}(x, A_\epsilon) + \mathrm{K}_{cms}(x, A_\epsilon^c) - (1 - \delta)^t\mathrm{K}_{cms}(x, A_\epsilon^c) \\
&= 1 - (1 - \delta)^t\mathrm{K}_{cms}(x, A_\epsilon^c) \\
&= 1 - (1 - \delta)^t(1 - \mathrm{K}_{cms}(x, A_\epsilon)) \\
&\geq 1 - (1 - \delta)^t(1 - \delta) \qquad \text{(by assumption)} \\
&= 1 - (1 - \delta)^{t+1}
\end{aligned}
$$

Consequently, the hypothesis is true ■

Now, we can associate the behavior of an EA, modeled as a Markovian chain, with the preconditions of Lemma 8.1 as follows:

THEOREM 8.1

A $(\mu + \lambda)$-EA whose Markovian kernel satisfies the conditions $\mathrm{K}_{cms}(x, A_\epsilon) \geq \delta > 0$ for all $x \in A_\epsilon^c = E \setminus A_\epsilon$ and $\mathrm{K}_{cms}(x, A_\epsilon) = 1$ for all $x \in A_\epsilon$ will *converge completely* to the feasible global minimum of a real-valued function f regardless of the initial distribution.

Proof:

Let $p(\cdot)$ denote the arbitrary initial distribution of an EA. Then

$$
\begin{aligned}
\mathsf{P}\{X_t \in A_\epsilon\} &= \int_E \mathrm{K}_{cms}^{(t-1)}(x, A_\epsilon)p(dx) \\
&\geq [1 - (1 - \delta)^{t-1}] \int_E p(dx) \qquad \text{(by Lemma 8.1)} \\
&= 1 - (1 - \delta)^{t-1}
\end{aligned}
\tag{9.2}
$$

leading to

$$\lim_{t\to\infty} \mathsf{P}\{b(X_t) - f^* \leq \epsilon\} = \lim_{t\to\infty} \mathsf{P}\{X_t \in A_\epsilon\} \geq \lim_{t\to\infty} 1 - (1-\delta)^{t-1} = 1 \tag{9.3}$$

and hence $b(X_t) \xrightarrow{\mathsf{P}} f^*$. Thus, under the precondition of Lemma 8.1 it is guaranteed that a $(\mu + \lambda)$-EA will *converge almost surely* to the global optimum ([161] p. 163). But the mode of convergence can be shifted from *almost surely* to *complete convergence*, since

$$\sum_{t=0}^{\infty} \mathsf{P}\{b(x_t) - f^* > \epsilon\} \leq \mathsf{P}\{X_0 \notin A_\epsilon\} + \sum_{t=1}^{\infty}(1-\delta)^t \leq 1 + (1/\delta - 1) = 1/\delta < \infty \tag{9.4}$$

it follows that $b(X_t) \xrightarrow{c} f^*$. ■

Now, we will prove that our SMES satisfies the preconditions of Theorem 8.1. As it was mentioned, the condition $\mathrm{K}_{cms}(x, A_\epsilon) \geq \delta > 0$ for every $x \in A_\epsilon$ is sufficient because elitism is employed, and thus $\mathrm{K}_{cms}(x, A_\epsilon) = 1$ for $x \in A_\epsilon$. Elitism is applied in the selection process of SMES and its aim is to allow the best feasible solution found so far to remain in the population for the next generation. If there are no feasible solutions, the solution with the lowest sum of constraint violation will remain in the population.

THEOREM 8.2

Let X_0 the initial population of some elitist EA and let K_c, K_m and K_s, denote the stochastic kernels of the crossover, mutation and selector operators, respectively. If the conditions:

$$\begin{aligned} &\text{(a)} \quad \exists\delta_c > 0 : \forall x \in E : \mathrm{K}_c(x, B(x)) \geq \delta_c \\ &\text{(b)} \quad \exists\delta_m > 0 : \forall x \in B(X_0) : \mathrm{K}_m(x, A_\epsilon) \geq \delta_m \\ &\text{(c)} \quad \exists\delta_s > 0 : \forall x \in E : \mathrm{K}_s(x, B(x)) \geq \delta_s \end{aligned} \tag{9.5}$$

hold simultaneously, then for every $\epsilon > 0$ there exists a $\delta > 0$ such that $\mathrm{K}_{cms}(x, A_\epsilon) \geq \delta > 0$ for every $x \in B(X_0)$.

Proof:

Since for an EA which adopts the selection operator "+" (with explicit elitism), it is guaranteed that the sequence $(X_t : t \geq 0)$ of populations satisfies the relation $b(X_{t+1}) \leq b(X_t)$ for all $t \geq 0$ and hence

$$B(X_t) \subseteq B(X_{t-1}) \subseteq \ldots \subseteq B(X_1) \subseteq B(X_0) \tag{9.6}$$

for $t \geq 0$. Therefore, as soon as SMES is initialized, the set of possible populations that can occur during the process is restricted to the set $B(x_0) \subseteq E$. Consider the product kernel of crossover and mutation: Let $x \in B(X_0) \setminus A_\epsilon \neq \emptyset$. Then

$$\begin{aligned} \mathrm{K}_{cm}(x, A_\epsilon) &= \int_E \mathrm{K}_c(x, dy)\mathrm{K}_m(y, A_\epsilon) \qquad (9.7) \\ &\geq \int_{B(x)} \mathrm{K}_c(x, dy)\mathrm{K}_m(y, A_\epsilon) \\ &\geq \int_{B(x)} \mathrm{K}_c(x, dy)\delta_m \qquad \text{by condition (b)} \\ &= \mathrm{K}_c(x, B(x))\delta_m \\ &\geq \delta_c\delta_m > 0 \qquad \text{by condition (a)} \end{aligned}$$

(9.8)

Suppose that $y \in A_\epsilon$. Then $B(y) \subseteq A_\epsilon$ and $\mathrm{K}_s(y, A_\epsilon) \geq \mathrm{K}_s(y, B(y)) \geq \delta_s$. This fact will be used to bound the product kernel of crossover, mutation and selection:

$$\begin{aligned} \mathrm{K}_{cms}(x, A_\epsilon) &= \int_E \mathrm{K}_{cm}(x, dy)\mathrm{K}_s(y, A_\epsilon) \qquad (9.9) \\ &\geq \int_{A_\epsilon} \mathrm{K}_{cm}(x, dy)\mathrm{K}_s(y, A_\epsilon) \\ &\geq \int_{A_\epsilon} \mathrm{K}_{cm}(x, dy)\delta_s \qquad \text{by condition (c)} \\ &= \mathrm{K}_{cm}(x, A_\epsilon)\delta_s \\ &\geq \delta_c\delta_m\delta_s =: \delta > 0 \end{aligned}$$

(9.10)

using the bound 9.8 in the last step. ■

In the following, we will prove the three conditions specifically for our SMES in order to prove its global convergence.

- **Condition (a)**: It is fulfilled by the recombination mechanism used by SMES. The combined recombination chooses with replacement (one parent can be chosen more than one time) $l + 1$ parents from the available μ with probability $1/\mu$. Let x_b be one of the best individuals in some population $x \in E$. The probability to choose a parent x_b exactly $(l + 1)$ times is $1/\mu^{(l+1)} > 0$ and, regardless of the recombination operator used (discrete or intermediate), parent x_b may remain in the population with probability $1 - (1 - (1/\mu^{(l+1)}))^\lambda > 0$. The λ exponent is due to the fact that the recombination process is performed λ times to generate λ offspring. If this event occurs, then the offspring population $\hat{x}$ (say) contains x_b and the relation $b(\hat{x}) \leq f(x_b) = b(x)$ is valid which implies $\hat{x} \in B(x)$. Consequently, one obtains $\mathrm{K}_c(x, B(x)) \geq 1 - (1 - (1/\mu^{(l+1)}))^\lambda =: \delta_c > 0$ for every $x \in E$.

- **Condition (b)**: Suppose that the lower level set $\mathrm{H}_{b(X_0)}$ is bounded regardless of the actual choice of the initial population $X_0 \in E$ and the mutation is realized by adding a random vector $z \sim N(0, \sigma)$ where σ can vary in a fixed compact (i.e. closed and bounded) subset of $\{\sigma \in \mathbb{R} : \sigma > 0\}$. Then, there exists an $\eta > 0$ such that $\mathrm{P}\{v + z \in H_{f^*+\epsilon}\} \geq \eta > 0$ for all $v \in \mathrm{H}_{b(X_0)} \setminus \mathrm{H}_{f^*+\epsilon} \subset R^l$ ([161] p. 204). Let $x \in B(X_0)$ be the population before mutation and x_b be one of the best individuals. Then, the probability that x_b will enter the set $\mathrm{H}_{f^*+\epsilon}$ by mutation, i.e. $x_b + z \in \mathrm{H}_{f^*+\epsilon}$ is at least $\eta > 0$. But if x_b enters $\mathrm{H}_{f^*+\epsilon}$ then the population x will enter the set A_ϵ at least with the same probability. Thus, condition (b) is fulfilled with $\delta_m = \eta > 0$.

- **Condition (c)**: As the "+" operator used by our SMES has explicit elitism, the argument is quite simple. One may set $\mathrm{K}_s(x, A) = 1_A(x)$. In this case we obtain $\mathrm{K}_s(x, B(x)) = 1_{B(x)}(x) = 1$ for all $x \in E$.

Thus. based on the previous mathematical analysis, we conclude that the SMES proposed in this dissertation converges to the feasible global optimum, assuming infinite time and finite population size.

Chapter 10

Final Remarks

10.1 Summary

We have presented a study about constraint-handling in evolutionary algorithms. After performing an empirical comparison of some approaches to handle constraints based on multiobjective concepts, their limitations were identified. The results of this study suggested the use of an approach based on a separation of objective function and constraints, but avoiding the use of multiobjective concepts. Therefore, we decided to use a selection process based on feasibility and a simple diversity mechanism that consists on maintaining infeasible solutions close to the boundaries of the feasible region of the search space and with a good value of the objective function.

The proposed approach uses an evolution strategy as its search engine. To improve the exploration capabilities of the approach, we used a combination of two recombination operators. Also, a reduction of the initial stepsize of the evolution strategy was used in order to favor fine movements on the search space. The approach was tested on 13 well known test functions and compared against three state-of-the-art algorithms providing competitive results with a low computational cost (measured by the number of fitness function evaluations). Also, our approach did not require the definition of extra parameters to be fine-tuned by the user (the parameters adopted can remain fixed). Based on several tests, we found that the recombination operator is the most important mechanism that helps the proposed approach to obtain high quality results consistently.

It was empirically shown that an evolution strategy provides better results than a genetic algorithm, even when both have used the same constraint handling and diversity mechanism.

Besides its good performance, the approach showed to be fast at reaching the feasible

region of the search space. Furthermore, for problems where the global optimum was reached, the approach required only about 25% of the total number of generations (200 out of a total of 800 generations). In addition, three performance measures were used to analyze the behavior of our algorithm compared with respect to the most competitive approach found in the literature: stochastic ranking [162]. From those results we concluded that our approach is able to reach the feasible region faster than stochastic ranking and we found that the progress inside the feasible region is also higher for our approach. Besides, we found that, despite generating a feasible population of solutions early in the evolutionary process, the diversity mechanism of the proposed approach is able to generate useful infeasible solutions as to reach the global optimum solution. Finally, we performed an analysis of variance in order to suggest values for the parameters of the approach and to detect any possible sensitivity of our algorithm to its parameters. The analysis did not identify a significant sensitivity of the approach to any of its parameters.

Based on the fact that the approach performed well in the benchmark proposed in the literature, we proposed the use of 11 new test functions in order to empirically detect the features of a constrained problem that make it difficult to solve by our proposed approach. The results suggest that a high dimensionality and a high number of nonlinear equality constraints decrease the performance of the proposed technique.

Finally, we formally proved the global convergence of our algorithm.

10.2 Conclusions

Based on the overall results and observations we provide the following conclusions:

- Although in this work we empirically showed that the use of multiobjective concepts to deal with global optimization problems with constraints is not adequate, the idea seems to work well on problems where there are no equality constraints.

- The usefulness of competitive diversity mechanisms incorporated to constraint-handling mechanisms was highlighted in this work.

- It was also remarked in this work that the selection of the search engine is very important (an ES over a GA in our case), even more important than the design of the constraint-handling mechanism, when solving constrained optimization problems.

- In order to provide a more in-depth comparison of evolutionary approaches when solving constrained problems, it is very important to use performance measures. In this way, more insights about the behavior of the approaches can be obtained.

- The use of statistical tests from the sample of independent runs improves the quality of the discussions of results and allows more in-depth conclusions.

- The use of an analysis of variance helps practitioners to have a more detailed knowledge about the evolutionary algorithm used and its parameters. In this way, the calibration of them can be statistically-based.

- Despite the ANOVA performed showed no sensitivity of SMES to its parameters, an interesting relationship was found between the approximated size of the feasible region and the initial mutation stepsize. For large feasible regions large initial stepsizes are adequate and for small feasible regions small initial stepsizes are more convenient.

- SMES is an approach able to reach the feasible region reasonably fast, and also has an effective diversity mechanism to maintain good infeasible solutions during the evolutionary process. However, SMES presented problems when dealing with a considerable high number of nonlinear equality constraints (five or more). In these kind of problems, approaches like SR provided better results.

- The new benchmark proposed in Chapter 7 coupled with the original thirteen test problems provides a more complete set of problems in order to promote the design of more competitive EAs to deal with constrained search spaces.

- Theoretical work about EAs in constrained search spaces is almost unexplored. The mathematical proof provided in this work was only an adaptation of a global convergence found in the literature, which is based on the capabilities of the mutation operator of sampling any point in the search space, the capability of the recombination operator of not destroying a solution in the neighborhood of the optimum and the ability of the selection mechanism to keep always the best solutions found so far. Constraints are not taken into account in an explicit way in our proof. Therefore, more work in this area is required.

- The main contributions of this work to the area are:

  - The idea that the selection of an adequate search engine will lead to a not necessarily complex constraint-handling mechanism.
  - The use of performance measures to analyze the progress inside the feasible region and to analyze the usefulness of diversity mechanisms.
  - The use of new test problems in order to identify sources of difficulty.

10.3 Future Work

Part of our future work includes the following:

- To test the constraint handling mechanism, as well as the diversity mechanism over other evolution-based heuristics like Particle Swarm Optimization [101] and Differential Evolution [148].
- To incorporate variations of the original ES mutation operator, like the derandomized self-adaptation proposed by Ostermeier et al. [139] and by Hansen and Ostermeier [137].
- Due to the relevance shown by the combined recombination operator, we are interested in testing other types of recombination operators like generalized panmictic intermediate recombination.
- We want to apply the SMES to the solution of engineering optimization problems solved with a previous version of the algorithm.

Appendix A: Test functions

1. **g01**:
   Minimize: $f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$ subject to:

$$\begin{aligned} g_1(\vec{x}) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\ g_2(\vec{x}) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\ g_3(\vec{x}) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\ g_4(\vec{x}) &= -8x_1 + x_{10} \leq 0 \\ g_5(\vec{x}) &= -8x_2 + x_{11} \leq 0 \\ g_6(\vec{x}) &= -8x_3 + x_{12} \leq 0 \\ g_7(\vec{x}) &= -2x_4 - x_5 + x_{10} \leq 0 \\ g_8(\vec{x}) &= -2x_6 - x_7 + x_{11} \leq 0 \\ g_9(\vec{x}) &= -2x_8 - x_9 + x_{12} \leq 0 \end{aligned}$$

   where the bounds are $0 \leq x_i \leq 1$ ($i = 1, \ldots, 9$), $0 \leq x_i \leq 100$ ($i = 10, 11, 12$) and $0 \leq x_{13} \leq 1$. The global optimum is located at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ where $f(x^*) = -15$. Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

2. **g02**:
   Maximize: $f(\vec{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$
   subject to:

$$\begin{aligned} g_1(\vec{x}) &= 0.75 - \prod_{i=1}^{n} x_i \leq 0 \\ g_2(\vec{x}) &= \sum_{i=1}^{n} x_i - 7.5n \leq 0 \end{aligned}$$

   where $n = 20$ and $0 \leq x_i \leq 10$ $(i = 1, \ldots, n)$. The global maximum is unknown; the best reported solution is [162]: $f(x^*) = 0.803619$. Constraint g_1 is close to being active ($g_1 = -10^{-8}$).

3. **g03**:
   Maximize: $f(\vec{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$

   subject to:

   $h(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0$

   where $n = 10$ and $0 \leq x_i \leq 1$ $(i = 1, \ldots, n)$. The global maximum is located at $x_i^* = 1/\sqrt{n}$ $(i = 1, \ldots, n)$ where $f(x^*) = 1$.

4. **g04**:
   Minimize: $f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$
   subject to:
   $g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$
   $g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$
   $g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$
   $g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$
   $g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$
   $g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$

   where: $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$ $(i = 3, 4, 5)$. The global optimum is located at $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ where $f(x^*) = -30665.539$. Constraints g_1 and g_6 are active.

5. **g05**

   Minimize:$f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$

   subject to:
   $g_1(\vec{x}) = -x_4 + x_3 - 0.55 \leq 0$
   $g_2(\vec{x}) = -x_3 + x_4 - 0.55 \leq 0$
   $h_3(\vec{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$
   $h_4(\vec{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$
   $h_5(\vec{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$

   where $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$, and $-0.55 \leq x_4 \leq 0.55$. The best known solution is $x^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$ where $f(x^*) = 5126.4981$.

6. **g06**
   Minimize: $f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$
   subject to:
   $g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$
   $g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$

   where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. The global optimum is located at $x^* = (14.095, 0.84296)$ where $f(x^*) = -6961.81388$. Both constraints are active.

7. **g07**
   Minimize: $f(\vec{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$
   subject to:
   $g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$
   $g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$
   $g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$
   $g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$
   $g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$
   $g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1 x_2 + 14x_5 - 6x_6 \leq 0$
   $g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$
   $g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$

   where $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 10)$. The global optimum is located at $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ where $f(x^*) = 24.3062091$. Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

8. **g08**
   Maximize: $f(\vec{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$
   subject to:
   $g_1(\vec{x}) = x_1^2 - x_2 + 1 \leq 0$
   $g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$

   where $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$. The global optimum is located at $x^* = (1.2279713, 4.2453733)$ where $f(x^*) = 0.095825$.

9. **g09**
   Minimize: $f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6 x_7 - 10x_6 - 8x_7$

subject to:

$$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$
$$g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$
$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$$
$$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

where $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 7)$. The global optimum is located at $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$ where $f(x^*) = 680.6300573$. Two constraints are active (g_1 and g_4).

10. **g10**
Minimize: $f(\vec{x}) = x_1 + x_2 + x_3$
subject to: $g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leq 0$
$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$
$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leq 0$
$g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$
$g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$
$g_6(\vec{x}) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$

where $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$, $(i = 2, 3)$, $10 \leq x_i \leq 1000$, $(i = 4, \ldots, 8)$. The global optimum is located at $x^* = (579.19, 1360.13, 5109.92, 182.0174, 295.5985, 217.9799, 286.40, 395.5979)$, where $f(x^*) = 7049.25$. g_1, g_2 and g_3 are active.

11. **g11**
Minimize: $f(\vec{x}) = x_1^2 + (x_2 - 1)^2$
subject to:
$h(\vec{x}) = x_2 - x_1^2 = 0$

where: $-1 \leq x_1 \leq 1$, $-1 \leq x_2 \leq 1$. The global optimum is located at $x^* = (\pm 1/\sqrt{2}, 1/2)$ where $f(x^*) = 0.75$.

12. **g12**
Maximize: $f(\vec{x}) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100}$
subject to:
$g_1(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$

where $0 \leq x_i \leq 10$ $(i = 1, 2, 3)$ and $p, q, r = 1, 2, \ldots, 9$. The feasible region of

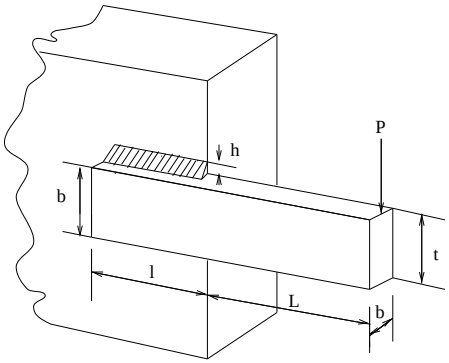


Figure A-1: The welded beam used for problem 14 .

the search space consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such the above inequality (12) holds. The global optimum is located at $x^* = (5, 5, 5)$ where $f(x^*) = 1$.

13. **g13**

Minimize: $f(\vec{x}) = e^{x_1 x_2 x_3 x_4 x_5}$
subject to:

$g_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$
$g_2(\vec{x}) = x_2 x_3 - 5x_4 x_5 = 0$
$g_3(\vec{x}) = x_1^3 + x_2^3 + 1 = 0$

where $-2.3 \leq x_i \leq 2.3$ $(i = 1, 2)$ and $-3.2 \leq x_i \leq 3.2$ $(i = 3, 4, 5)$. The global optimum is located at $x^* = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$ where $f(x^*) = 0.0539498$.

14. **Design of a Welded Beam**

A welded beam is designed for minimum cost subject to constraints on shear stress (τ), bending stress in the beam (σ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints [150]. There are four design variables as shown in Figure A-1 [150]: h (x_1), l (x_2), t (x_3) and b (x_4).

The problem can be stated as follows:

Minimize:

$$f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$$

Subject to:

$$\begin{aligned} g_1(\vec{x}) &= \tau(\vec{x}) - \tau_{max} \leq 0 \\ g_2(\vec{x}) &= \sigma(\vec{x}) - \sigma_{max} \leq 0 \\ g_3(\vec{x}) &= x_1 - x_4 \leq 0 \\ g_4(\vec{x}) &= 0.10471 x_1^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0 \leq 0 \\ g_5(\vec{x}) &= 0.125 - x_1 \leq 0 \\ g_6(\vec{x}) &= \delta(\vec{x}) - \delta_{max} \leq 0 \\ g_7(\vec{x}) &= P - P_c(\vec{x}) \leq 0 \end{aligned}$$

where

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$$

$$\tau' = \frac{P}{\sqrt{2}x_1 x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{\sqrt{2}x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma(\vec{x}) = \frac{6PL}{x_4 x_3^2}, \delta(\mathbf{X}) = \frac{4PL^3}{E x_3^3 x_4}$$

$$P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$$P = 6000\ lb,\ \ L = 14\ in,\ \ E = 30 \times 10^6\ psi,\ \ G = 12 \times 10^6\ psi$$

$$\tau_{max} = 13,600\ psi,\ \ \sigma_{max} = 30,000\ psi,\ \ \delta_{max} = 0.25\ in$$

where $0.1 \leq x_1 \leq 2.0$, $0.1 \leq x_2 \leq 10.0$, $0.1 \leq x_3 \leq 10.0$ y $0.1 \leq x_4 \leq 2.0$.

15. **Design of a Pressure Vessel**

A cylindrical vessel is capped at both ends by hemispherical heads as shown in Figure A-2. The objective is to minimize the total cost, including the cost of the material, forming and welding. There are four design variables: T_s (thickness of the shell), T_h (thickness of the head), R (inner radius) and L (length of the cylindrical section of

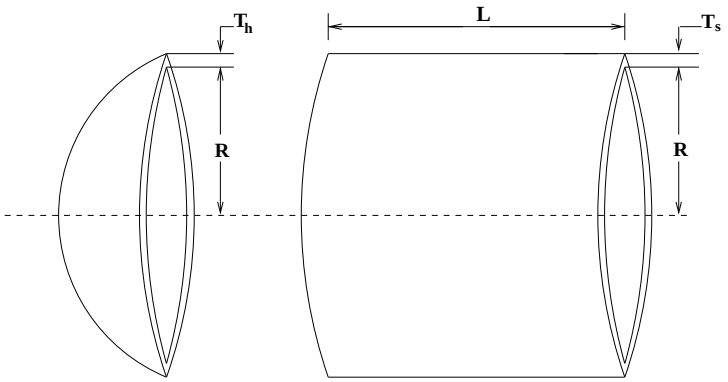


Figure A-2: Center and end section of the pressure vessel used for problem 15.

the vessel, not including the head). T_s and T_h are integer multiples of 0.0625 inch, which are the available thicknesses of rolled steel plates, and R and L are continuous. Using the same notation given by Kannan and Kramer [98], the problem can be stated as follows:

Minimize :

$$f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

Subject to :

$$\begin{aligned} g_1(\vec{x}) &= -x_1 + 0.0193x_3 \leq 0 \\ g_2(\vec{x}) &= -x_2 + 0.00954x_3 \leq 0 \\ g_3(\vec{x}) &= -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0 \\ g_4(\vec{x}) &= x_4 - 240 \leq 0 \end{aligned}$$

where $1 \leq x_1 \leq 99$, $1 \leq x_2 \leq 99$, $10 \leq x_3 \leq 200$ y $10 \leq x_4 \leq 200$.

16. **Minimization of the Weight of a Tension/Compression Spring**

This problem was described by Arora [5] and Belegundu [15], and it consists of minimizing the weight of a tension/compression spring (see Figure A-3) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the mean coil diameter D (x_2), the wire diameter d (x_1) and the number of active coils N (x_3).

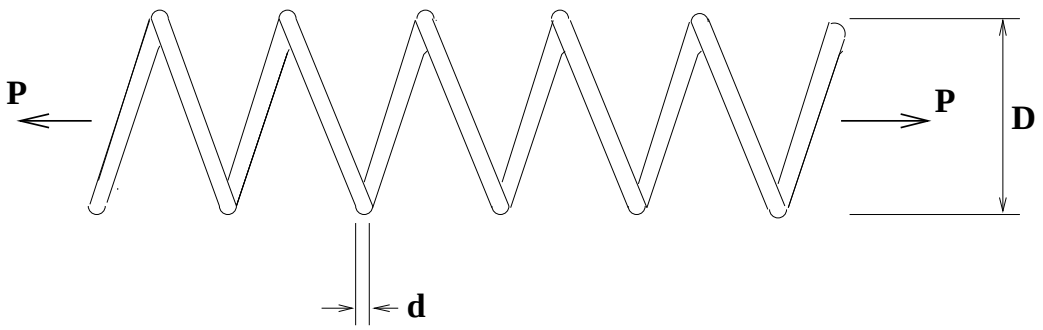


Figure A-3: Tension/compression spring used for problem 16.

Formally, the problem can be expressed as:

Minimize:

$$(N+2)Dd^2$$

Subject to:

$$\begin{aligned} g_1(\vec{x}) &= 1 - \frac{D^3 N}{71785 d^4} \leq 0 \\ g_2(\vec{x}) &= \frac{4D^2 - dD}{12566(Dd^3 - d^4)} + \frac{1}{5108 d^2} - 1 \leq 0 \\ g_3(\vec{x}) &= 1 - \frac{140.45 d}{D^2 N} \leq 0 \\ g_4(\vec{x}) &= \frac{D+d}{1.5} - 1 \leq 0 \end{aligned}$$

where $0.05 \leq x_1 \leq 2$, $0.25 \leq x_2 \leq 1.3$ y $2 \leq x_3 \leq 15$.

17. **Design of a 10-bar Plane Truss**

Consider the 10-bar plane truss shown in Figure A-4 [15]. The problem is to find the moment of inertia of each member of this truss, such that we minimize its weight, subject to stress and displacement constraints. The weight of the truss is given by:

$$f(x) = \sum_{j=1}^{10} \rho \, A_j \, L_j$$

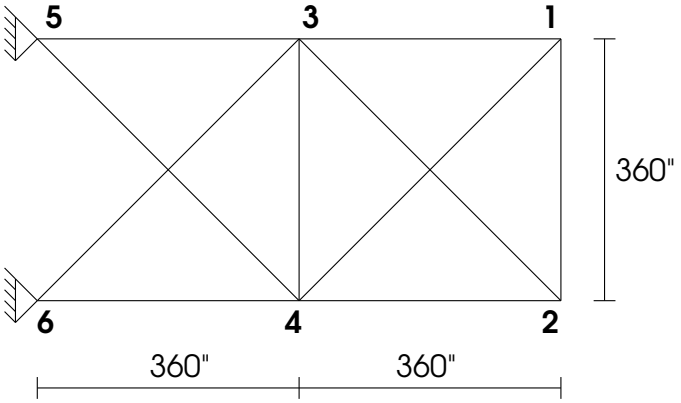


Figure A-4: 10-bar plane truss used for problem 17.

where x is the candidate solution, A_j is the cross-sectional area of the jth member, L_j is the length of the jth member, and ρ is the weight density of the material.

The assumed data are: modulus of elasticity, $E = 1.0 \times 10^4$ ksi 68965.5 MPa), $\rho = 0.10$ lb/in^3 (2768.096 kg/m^3), and a load of 100 kips (45351.47 Kg) in the negative y-direction is applied at nodes 2 and 4. The maximum allowable stress of each member is called σ_a, and it is assumed to be ± 25 ksi (172.41 MPa). The maximum allowable displacement of each node (horizontal and vertical) is represented by u_a, and is assumed to be 2 inches (5.08 cm).

There are 10 stress constraints, and 12 displacement constraints. The moment of inertia of each element can be different, thus the problem has 10 design variables.

18. **Minimization of the Weight of a Speed Reducer**

The weight of the speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surfaces stress, transverse deflections of the shafts and stresses in the shafts. The variables $x_1, x_2, \cdots, x_7$ are the face width, module of teeth, number of teeth in the pinion, length of the first shaft between bearings, length of the second shaft between bearings and the diameter of the first and second shafts. The third variable is integer, the rest of them are continuous.

Minimize : $f(\vec{x}) = 0.7854 x_1 x_2^2 (3.3333 x_3^2 + 14.9334 x_3 - 43.0934) - 1.508 x_1 (x_6^2 + x_7^2) + 7.4777 (x_6^3 + x_7^3) + 0.7854 (x_4 x_6^2 + x_5 x_7^2)$

Subject to :

$$
\begin{aligned}
g_1(\vec{x}) &= \frac{27}{x_1 x_2^2 x_3} - 1 \leq 0 \\
g_2(\vec{x}) &= \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \leq 0 \\
g_3(\vec{x}) &= \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \leq 0 \\
g_4(\vec{x}) &= \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \leq 0 \\
g_5(\vec{x}) &= \frac{\left(\left(\frac{745 x_4}{x_2 x_3}\right)^2 + 16.9 \times 10^6\right)^{1/2}}{110.0 x_6^3} - 1 \leq 0r \\
g_6(\vec{x}) &= \frac{\left(\left(\frac{745 x_5}{x_2 x_3}\right)^2 + 157.5 \times 10^6\right)^{1/2}}{85.0 x_7^3} - 1 \leq 0 \\
g_7(\vec{x}) &= \frac{x_2 x_3}{40} - 1 \leq 0 \\
g_8(\vec{x}) &= \frac{5 x_2}{x_1} - 1 \leq 0 \\
g_9(\vec{x}) &= \frac{x_1}{12 x_2} - 1 \leq 0 \\
g_{10}(\vec{x}) &= \frac{1.5 x_6 + 1.9}{x_4} - 1 \leq 0 \\
g_{11}(\vec{x}) &= \frac{1.1 x_7 + 1.9}{x_5} - 1 \leq 0
\end{aligned}
$$

where $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4 \leq 8.3$, $7.8 \leq x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$ and $5.0 \leq x_7 \leq 5.5$.

Problem 18 was only used to test one previous version of the SMES (the $(1+\lambda)$-ES detailed in Section 6.4).

Appendix B: Graphics of statistical tests of the SMES

All graphics in this appendix were generated using the software S-PLUS 6.2 Student Edition. A Continuous line in the histogram was drawn to visualize the histogram's distribution tendency.

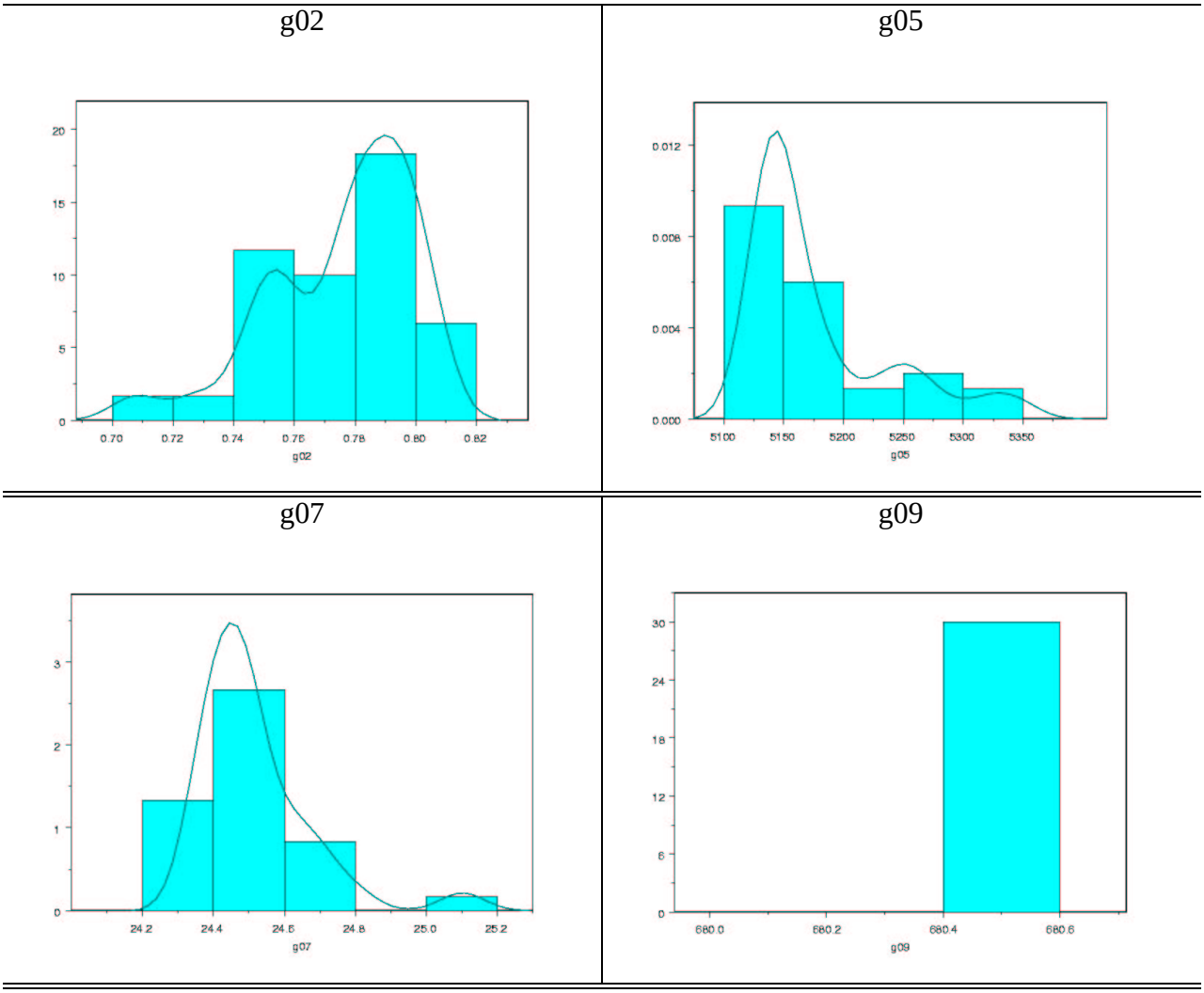


Figure B-1: Histogram and density line of the distribution of results obtained by the SMES in 30 independent runs. The omitted functions reached the global optimum in all runs.

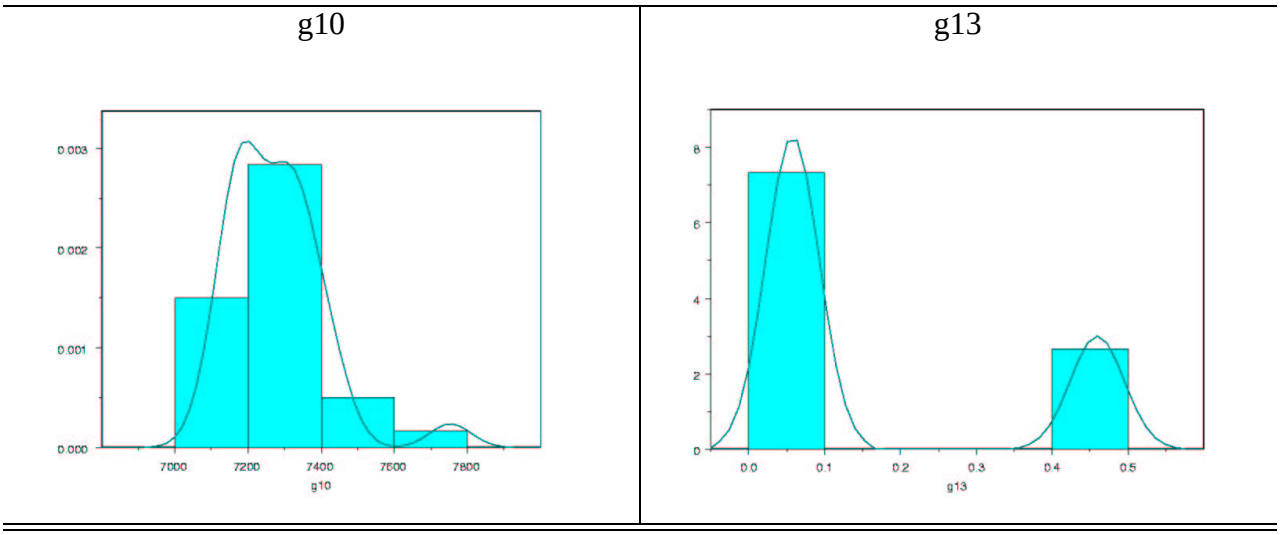


Figure B-2: Histogram and density line of the distribution of results obtained by the SMES in 30 independent runs. The omitted functions reached the global optimum in all runs.

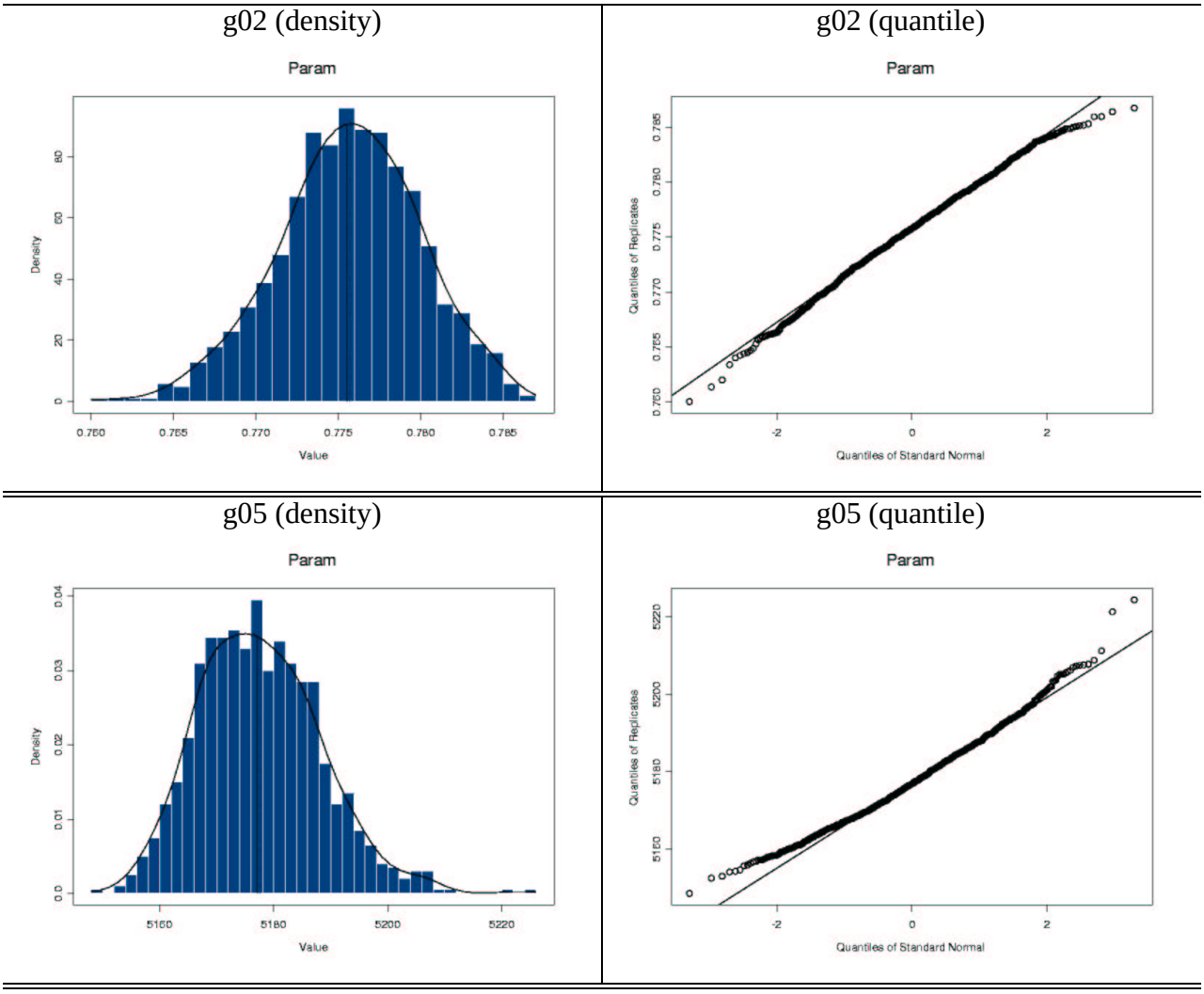


Figure B-3: Histogram of the sample mean values of results obtained by the SMES, generated by a bootstrapping process. Also shown is the normal quantile graph.

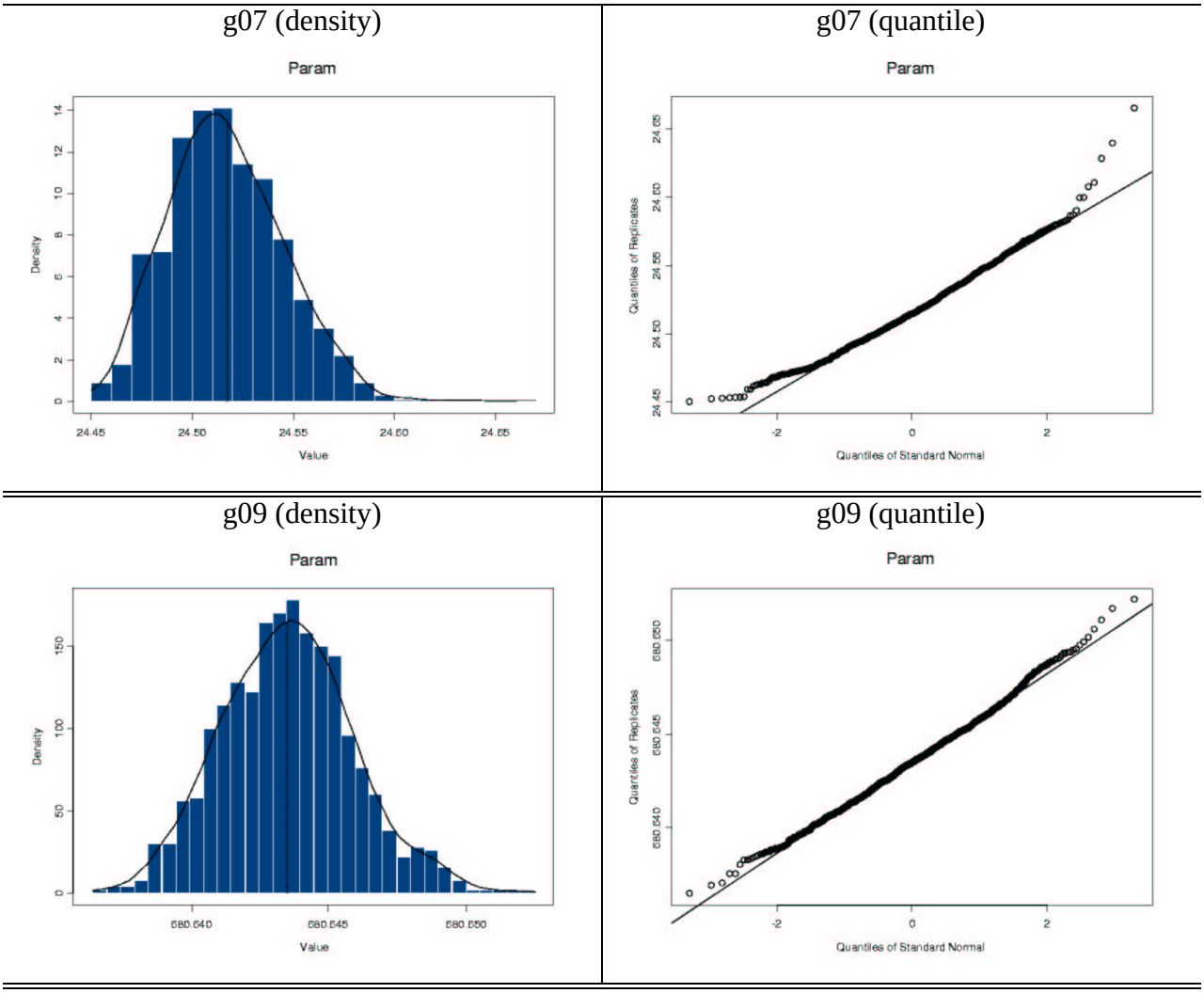


Figure B-4: Histogram of the sample mean values of results obtained by the SMES, generated by a bootstrapping process. Also shown is the normal quantile graph.

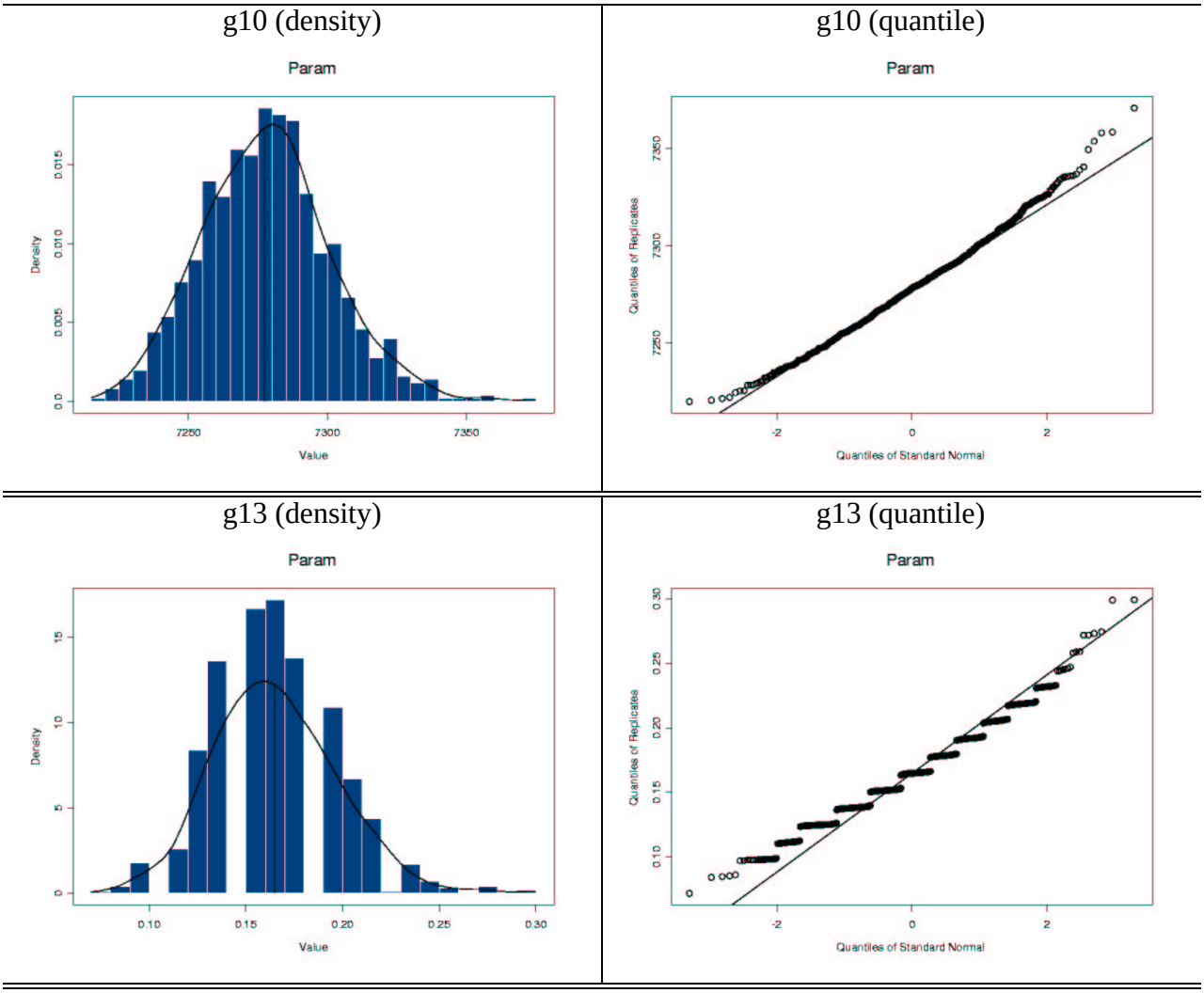


Figure B-5: Histogram of the sample mean values of results obtained by the SMES, generated by a bootstrapping process. Also shown is the normal quantile graph.

Appendix C: Graphics of statistical tests for the three performance measures

All graphics in this appendix were generated using the software S-PLUS 6.2 Student Edition. A Continuous line in the histogram was drawn to visualize the histogram's distribution tendency.

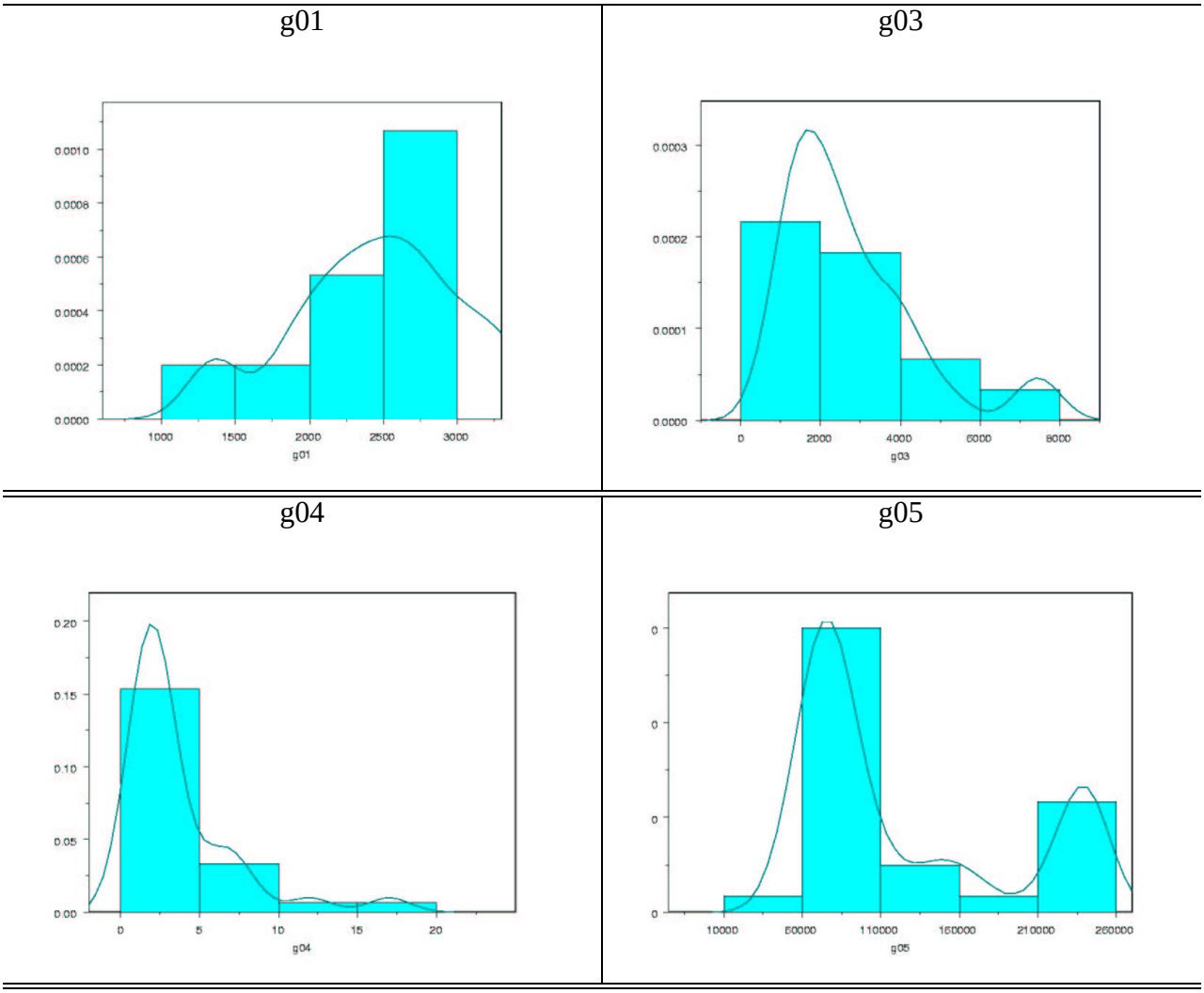


Figure C-1: Histogram and density line of the distribution of results for the EVALS performance measure obtained by the SMES in 30 independent runs. In the omitted function g02, a feasible solution is found in the first generation for all runs.

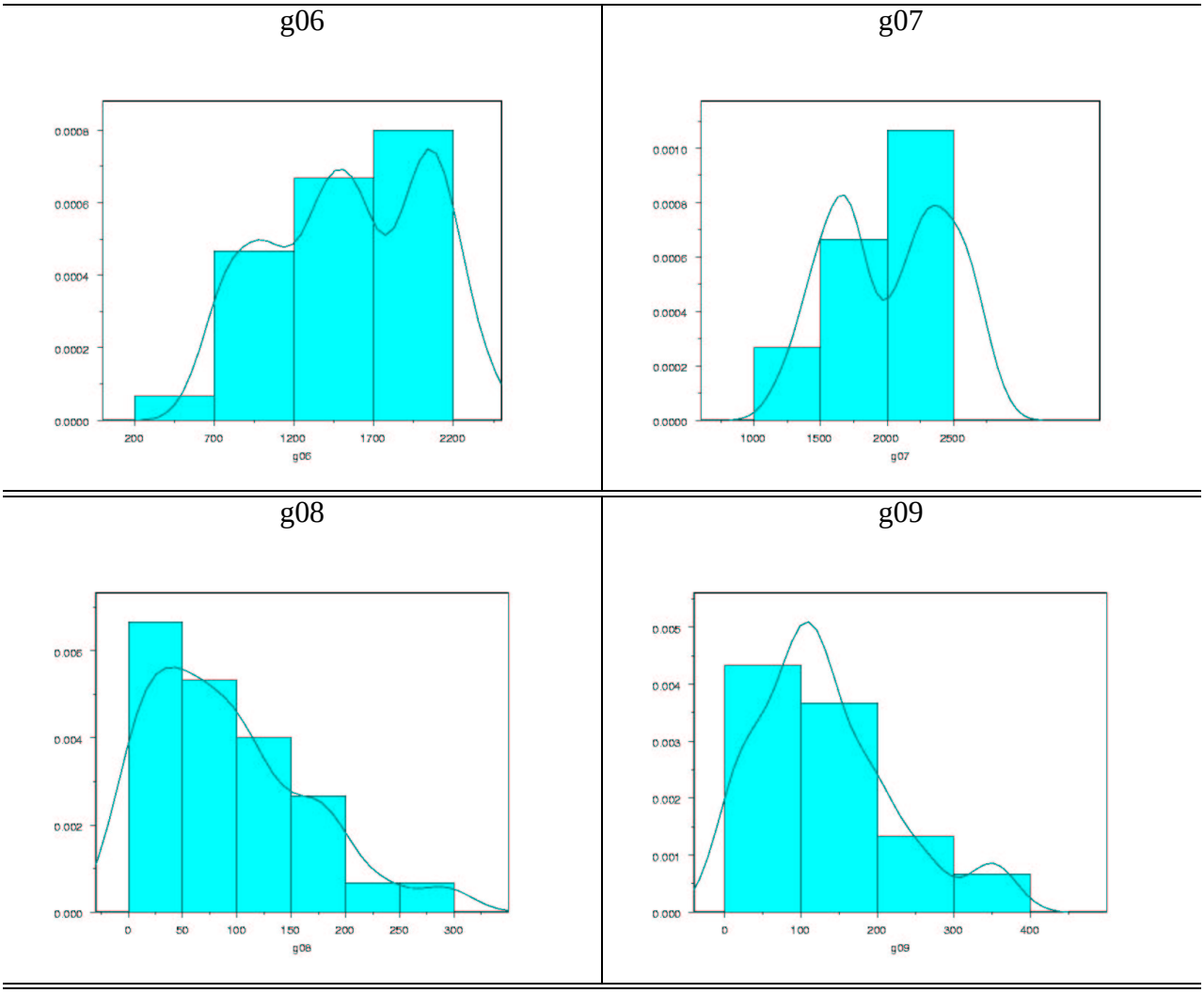


Figure C-2: Histogram and density line of the distribution of results for the EVALS performance measure obtained by the SMES in 30 independent runs. In the omitted function g02, a feasible solution is found in the first generation for all runs.

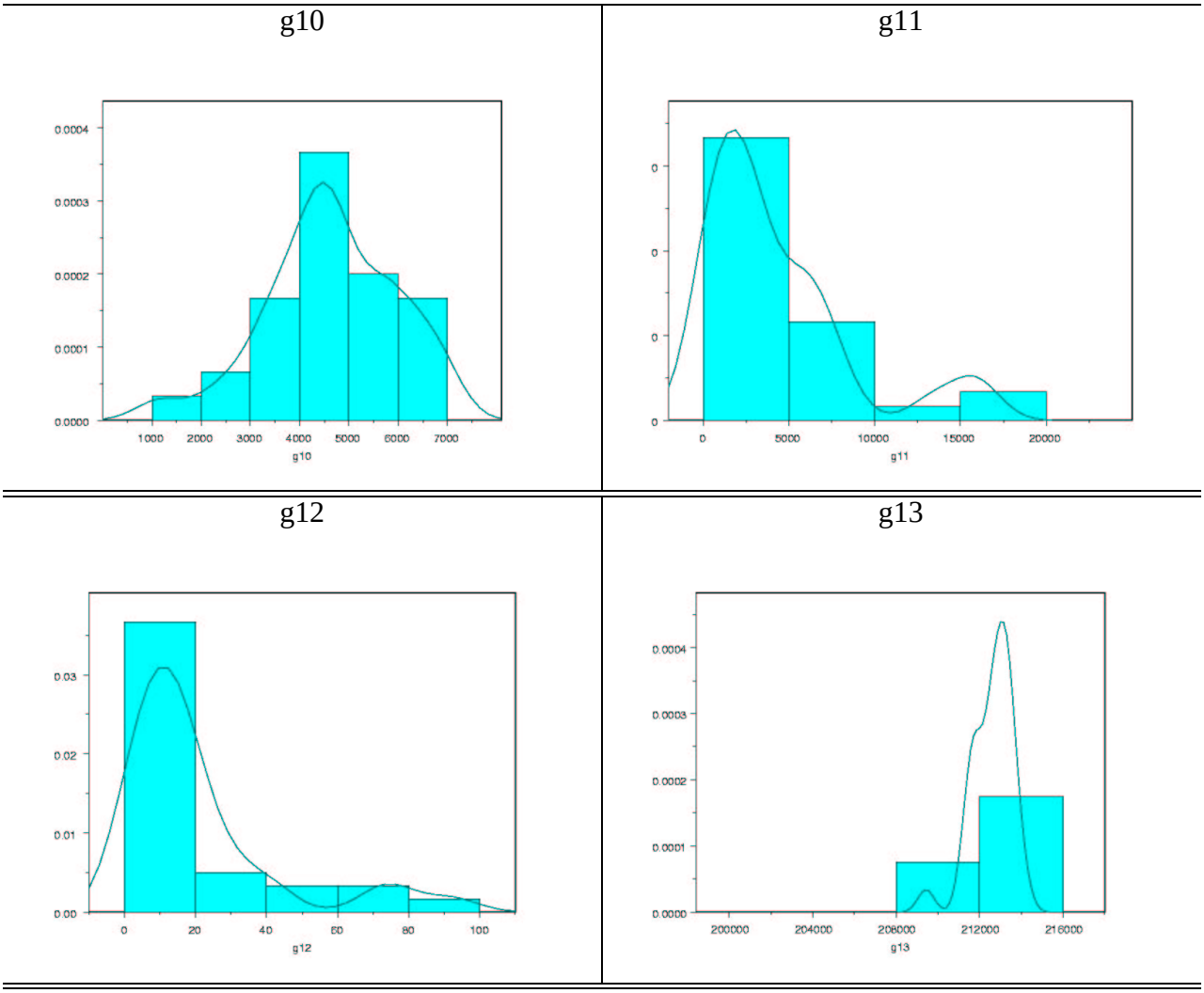


Figure C-3: Histogram and density line of the distribution of results for the EVALS performance measure obtained by the SMES in 30 independent runs. In the omitted function g02, a feasible solution is found in the first generation for all runs.

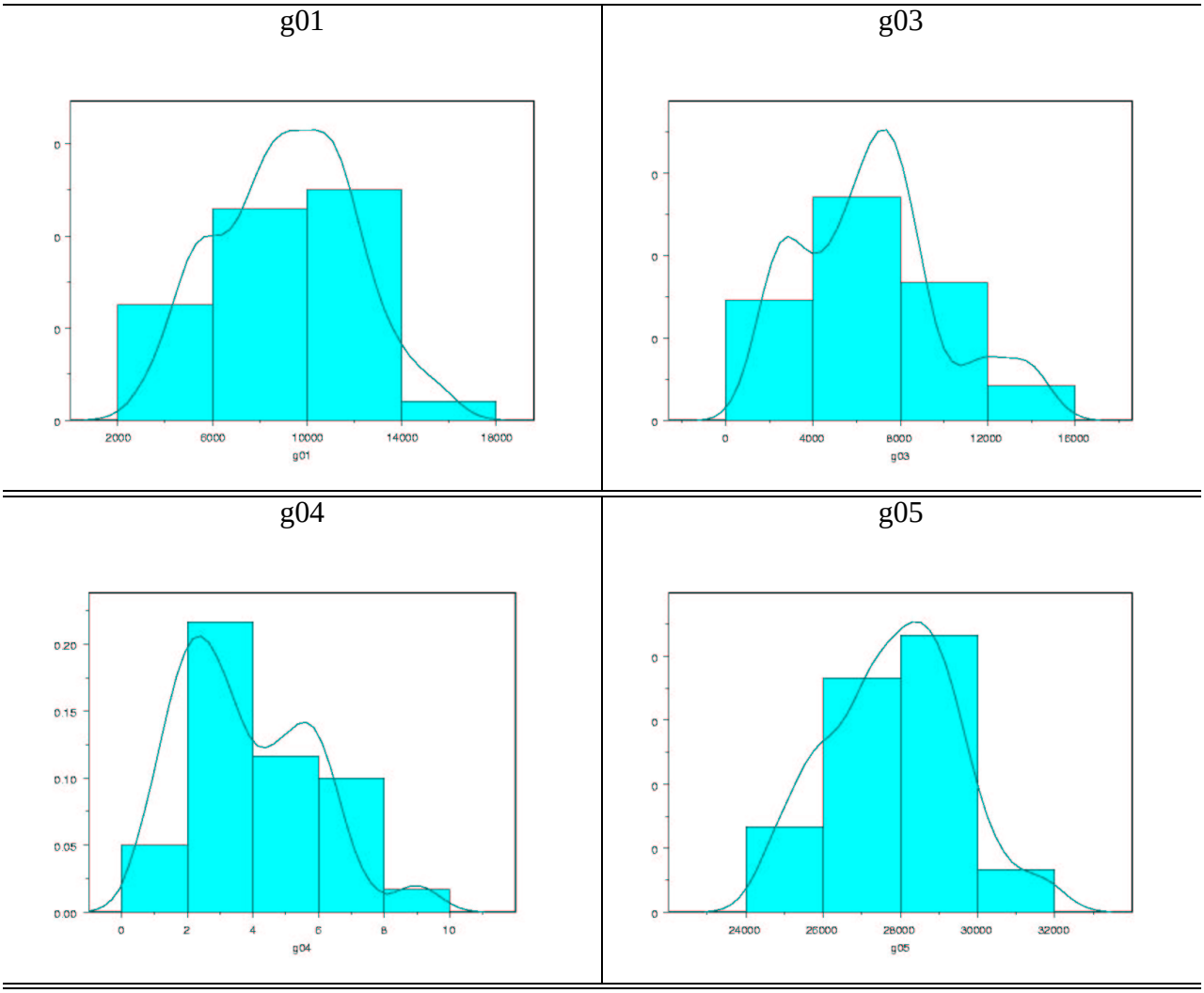


Figure C-4: Histogram and density line of the distribution of results for the EVALS performance measure obtained by the Stochastic Ranking in 30 independent runs. In the omitted function g02, a feasible solution is found in the first generation for all runs.

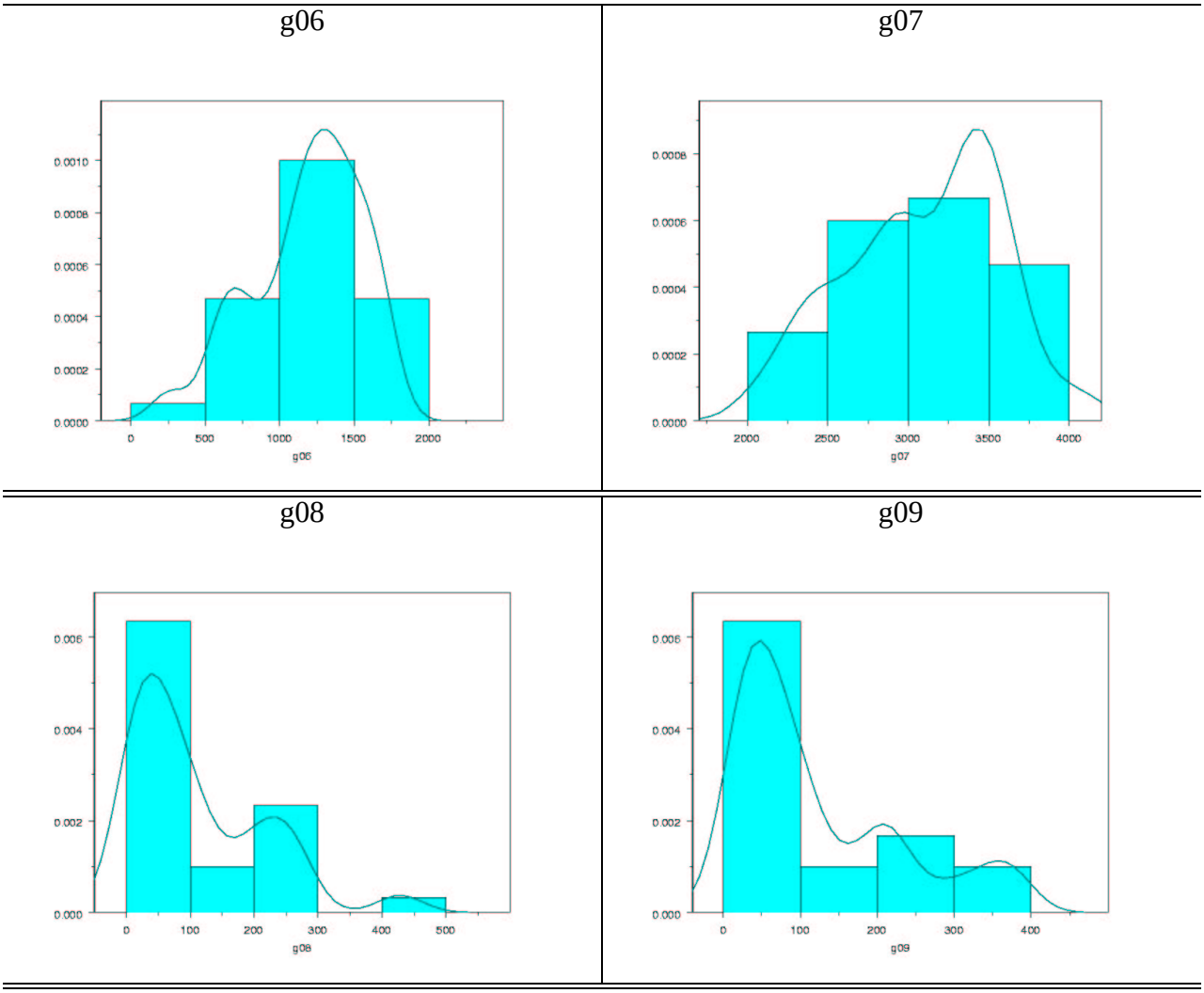


Figure C-5: Histogram and density line of the distribution of results for the EVALS performance measure obtained by the Stochastic Ranking in 30 independent runs. In the omitted function g02, a feasible solution is found in the first generation for all runs.

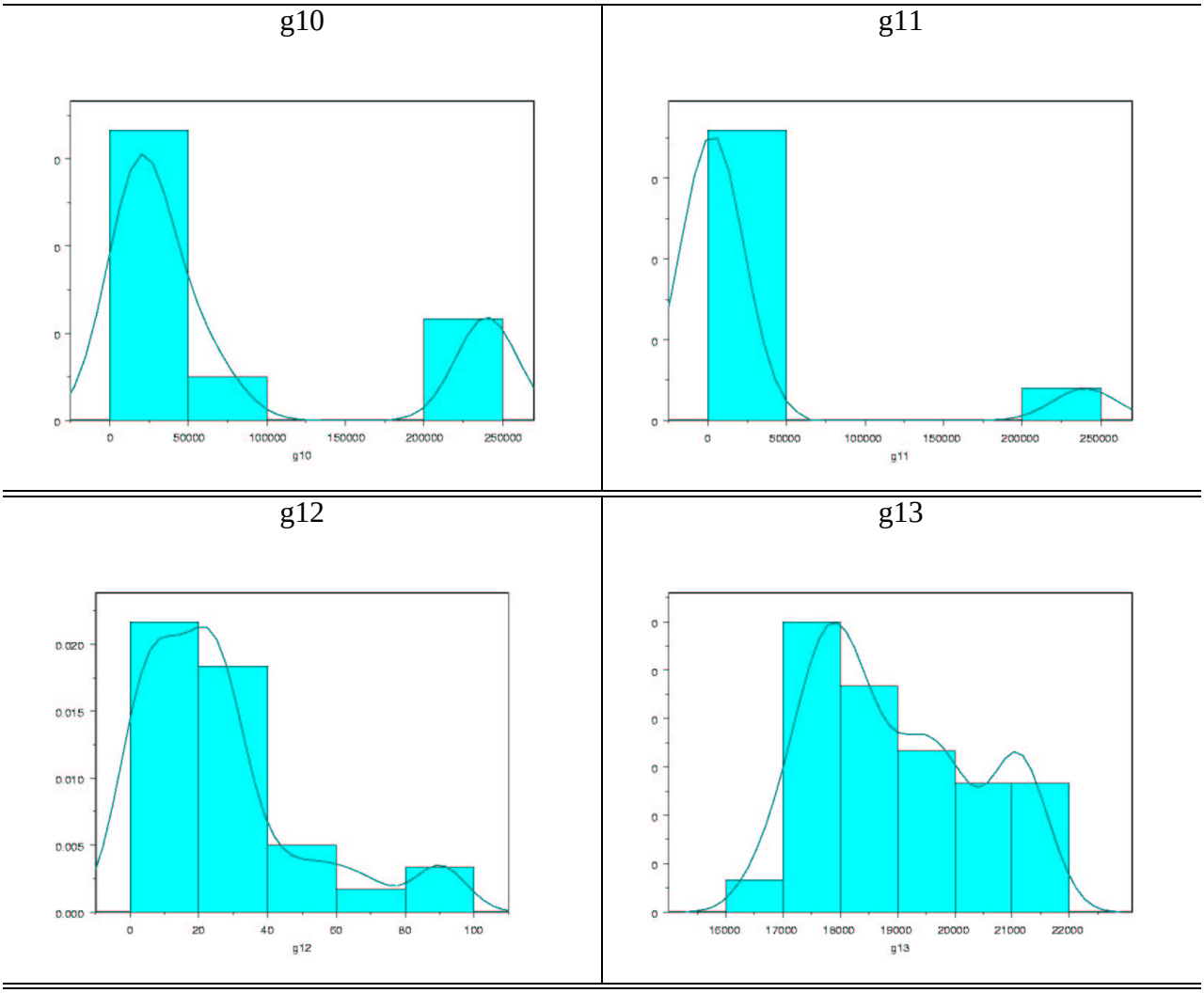


Figure C-6: Histogram and density line of the distribution of results for the EVALS performance measure obtained by the Stochastic Ranking in 30 independent runs. In the omitted function g02, a feasible solution is found in the first generation for all runs.

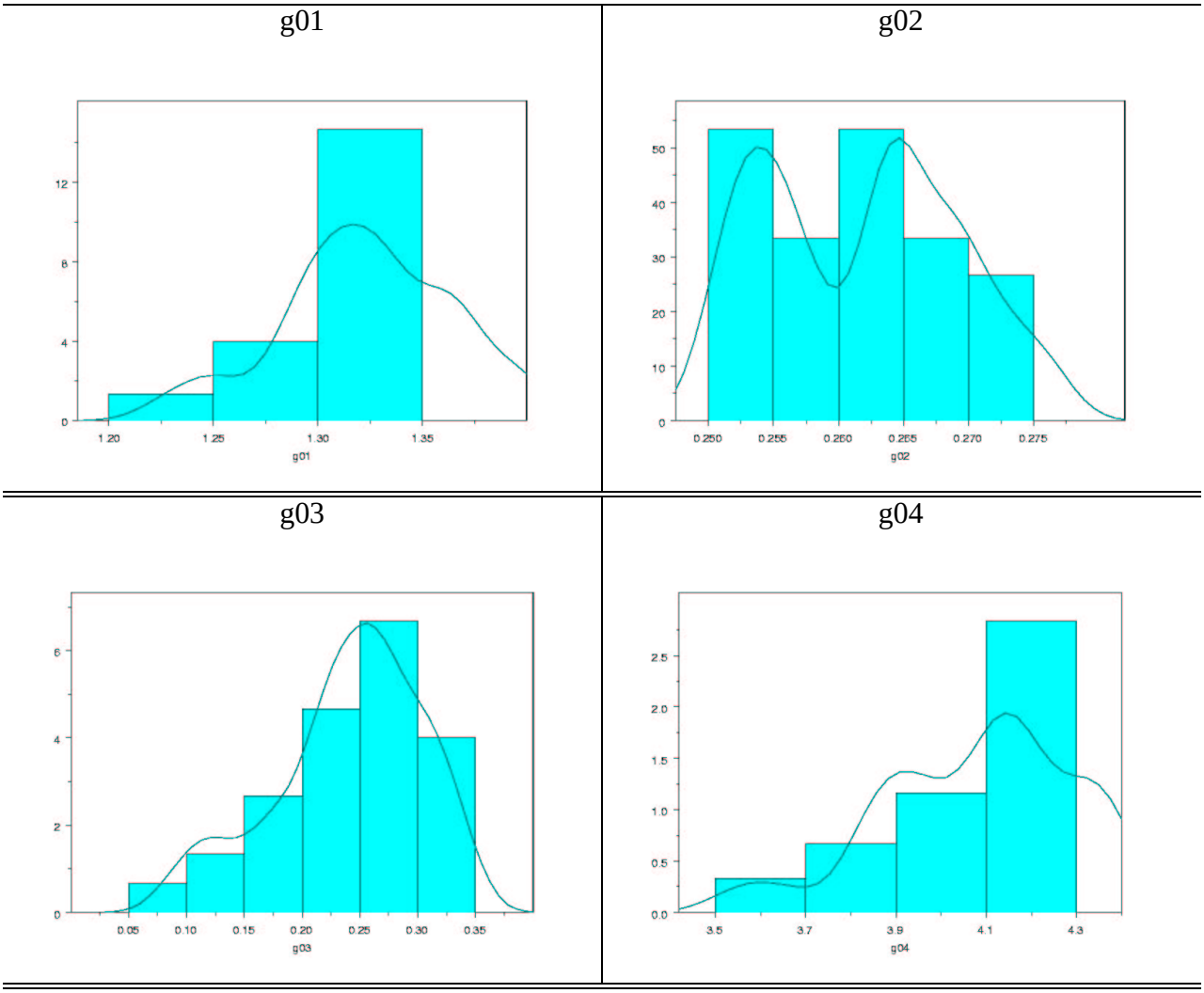


Figure C-7: Histogram and density line of the distribution of results for the PROGRESS-RATIO performance measure obtained by the SMES in 30 independent runs.

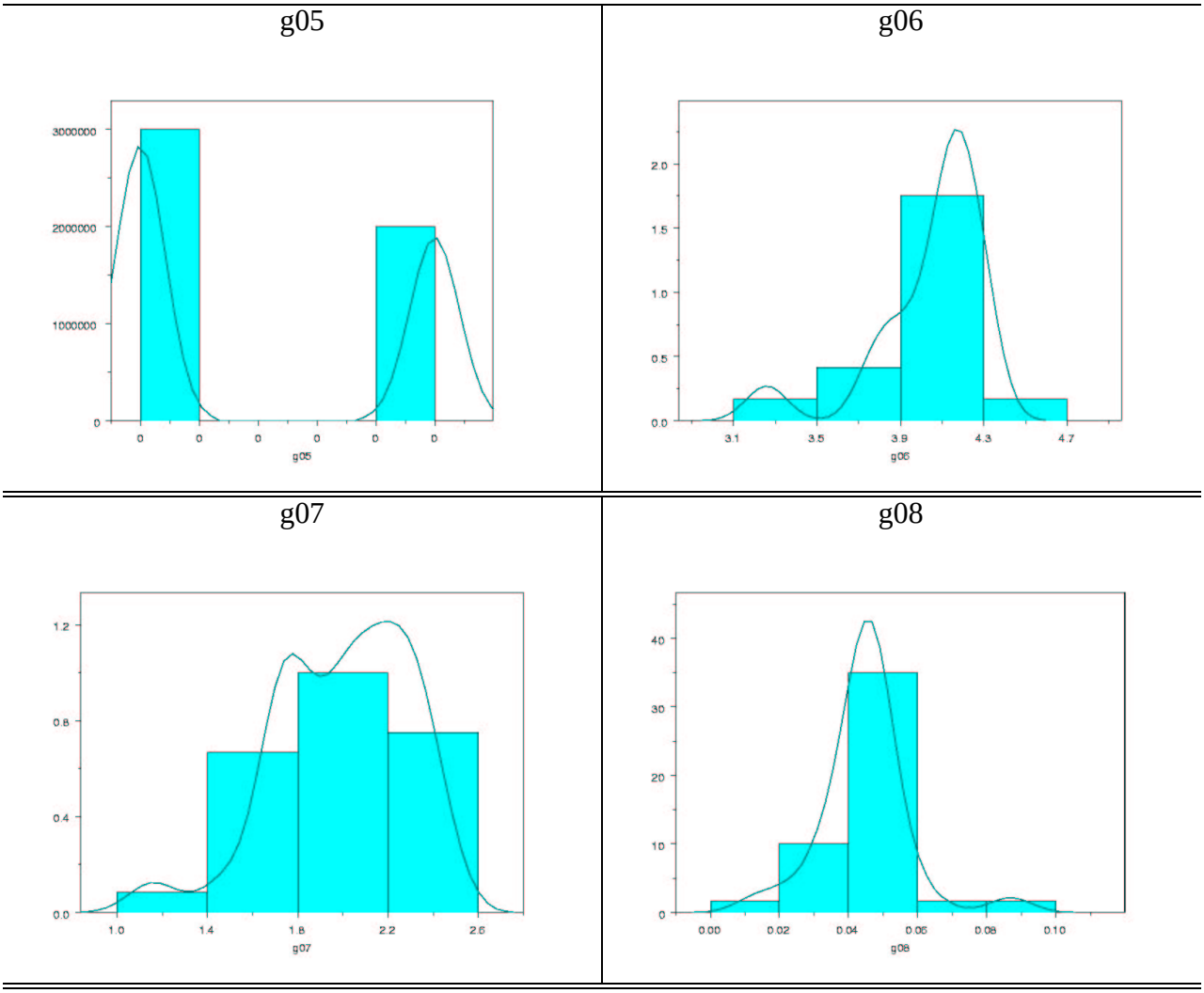


Figure C-8: Histogram and density line of the distribution of results for the PROGRESS-RATIO performance measure obtained by the SMES in 30 independent runs.

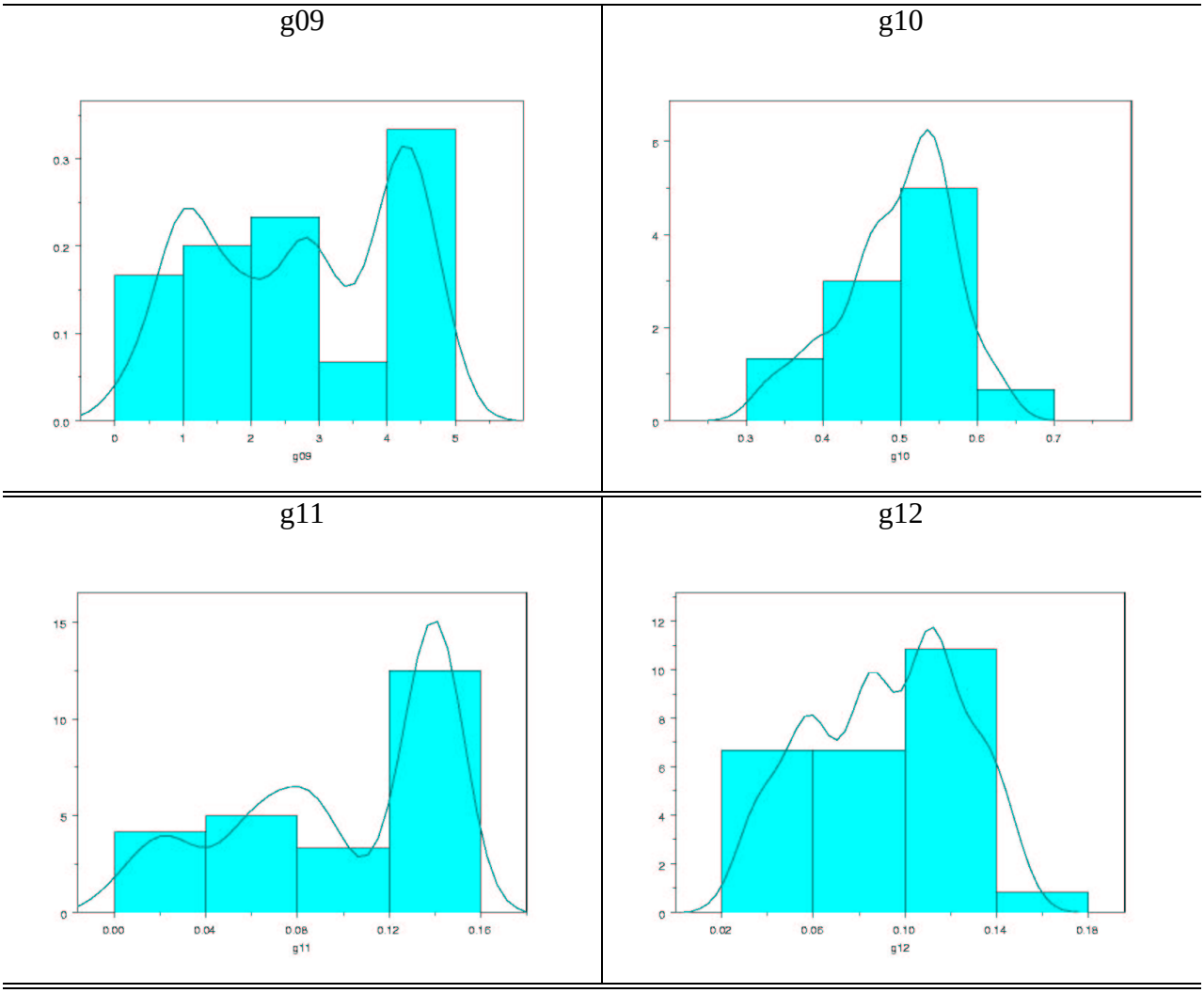


Figure C-9: Histogram and density line of the distribution of results for the PROGRESS RATIO performance measure obtained by the SMES in 30 independent runs.

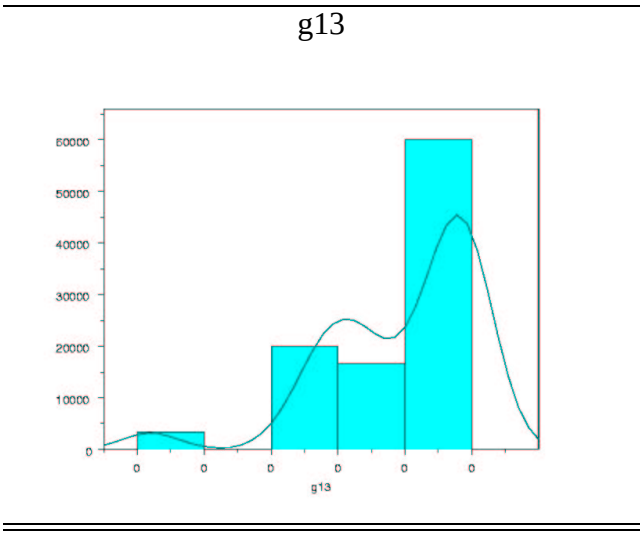


Figure C-10: Histogram and density line of the distribution of results for the PROGRESS RATIO performance measure obtained by the SMES in 30 independent runs.

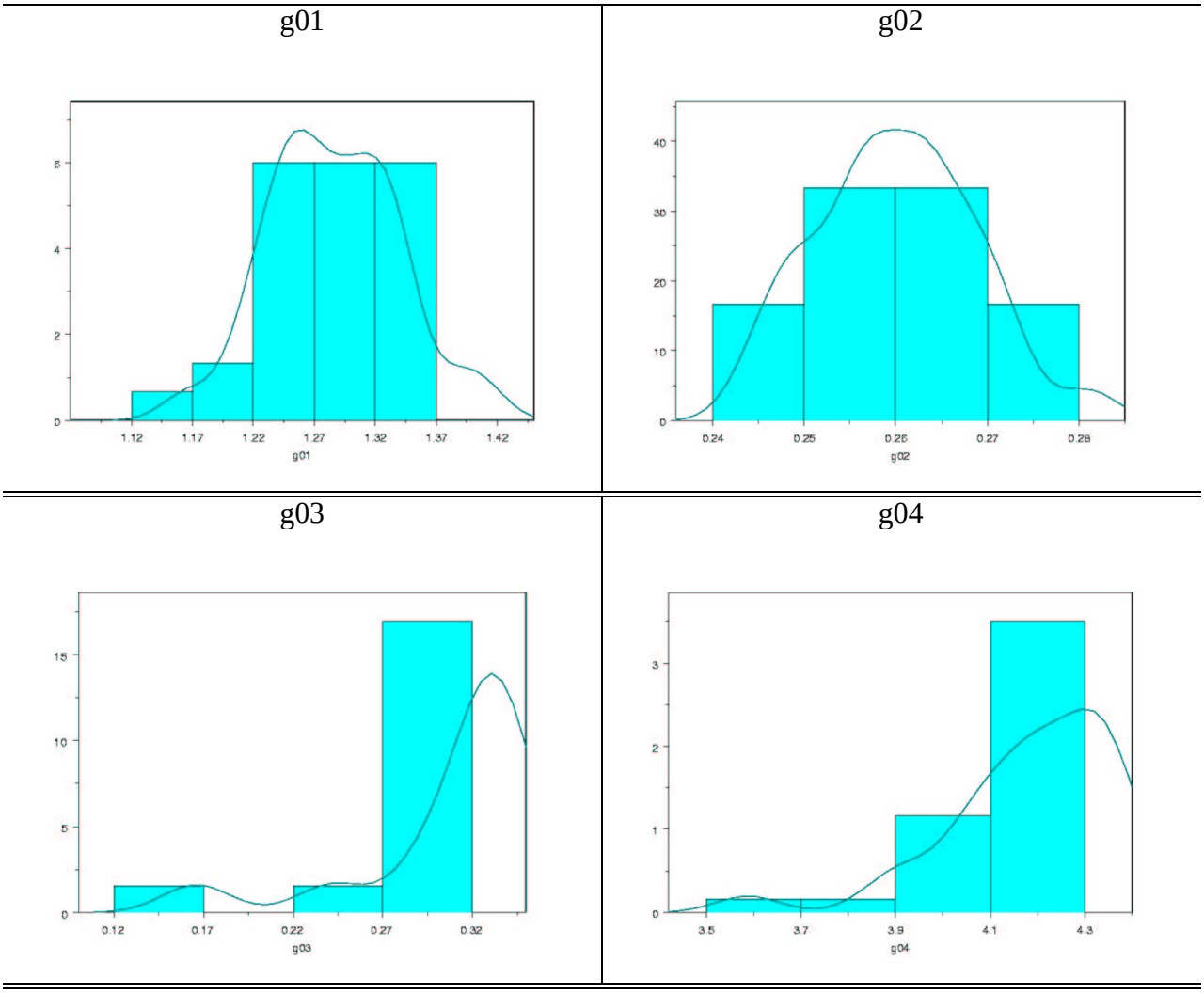


Figure C-11: Histogram and density line of the distribution of results for the PROGRESS-RATIO performance measure obtained by the Stochastic Ranking in 30 independent runs.

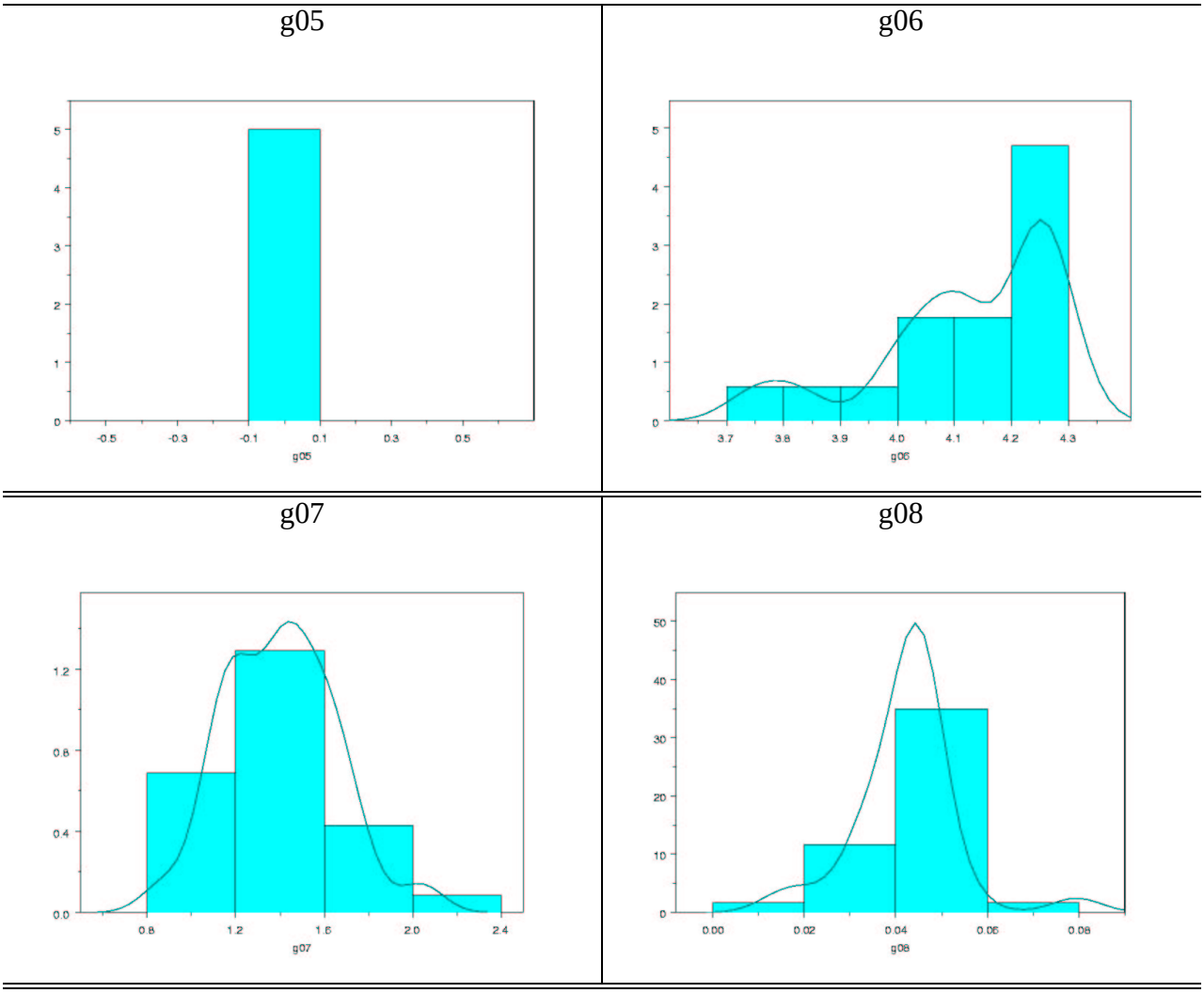


Figure C-12: Histogram and density line of the distribution of results for the PROGRESS-RATIO performance measure obtained by the Stochastic Ranking in 30 independent runs.

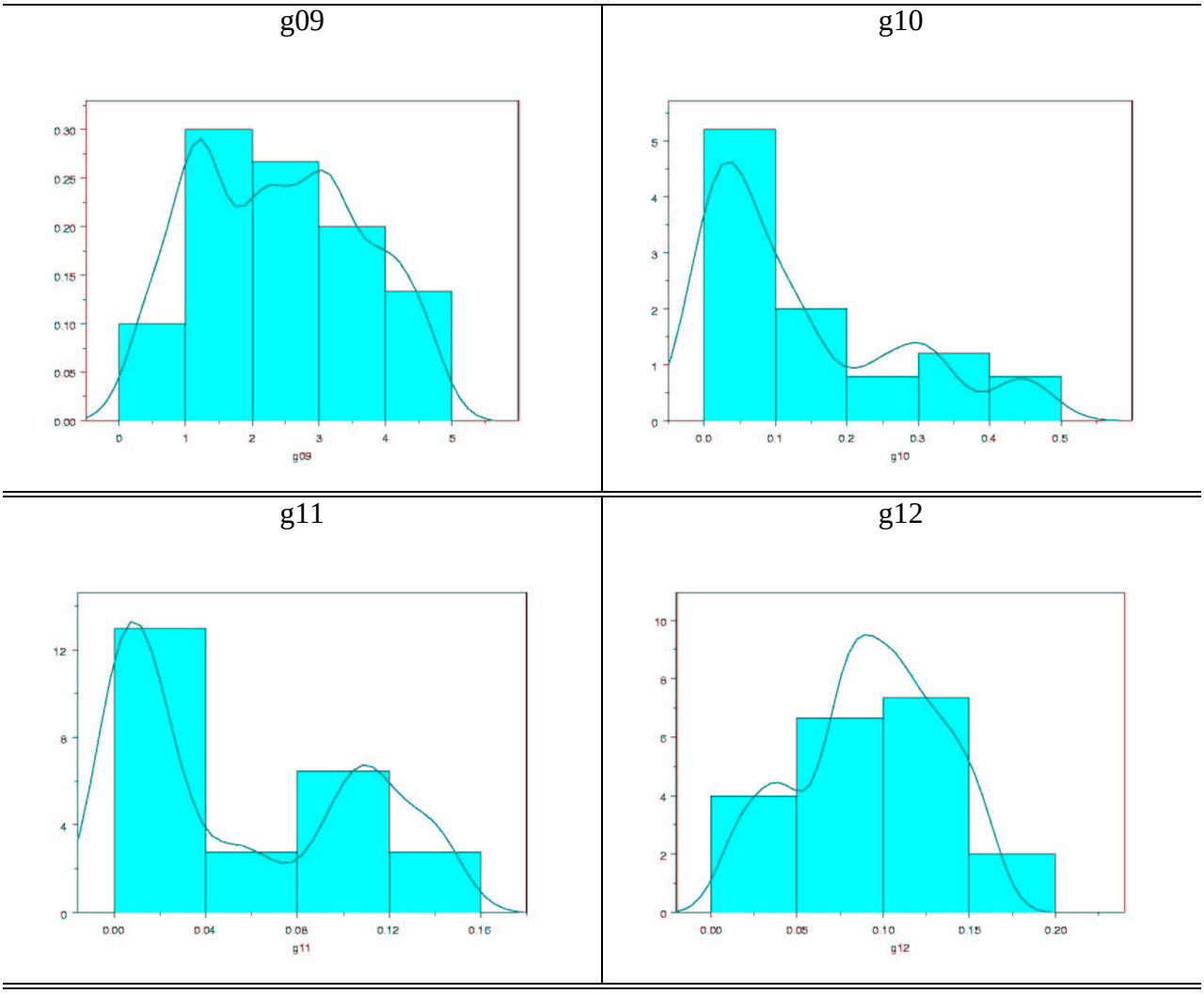


Figure C-13: Histogram and density line of the distribution of results for the PROGRESS RATIO performance measure obtained by the Stochastic Ranking in 30 independent runs.

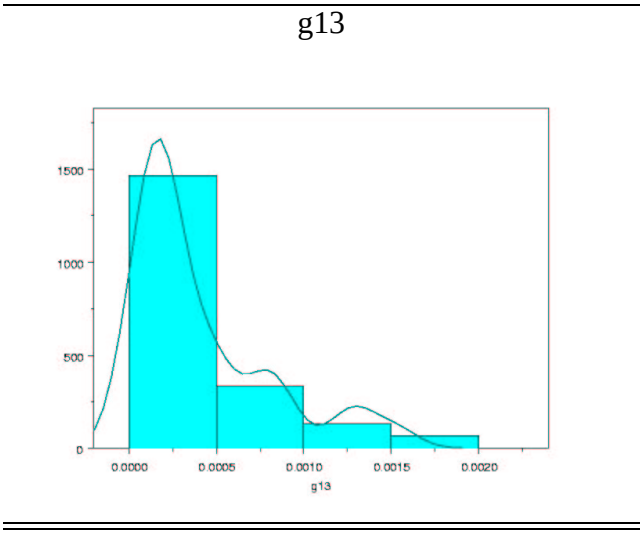


Figure C-14: Histogram and density line of the distribution of results for the PROGRESS RATIO performance measure obtained by the Stochastic Ranking in 30 independent runs.

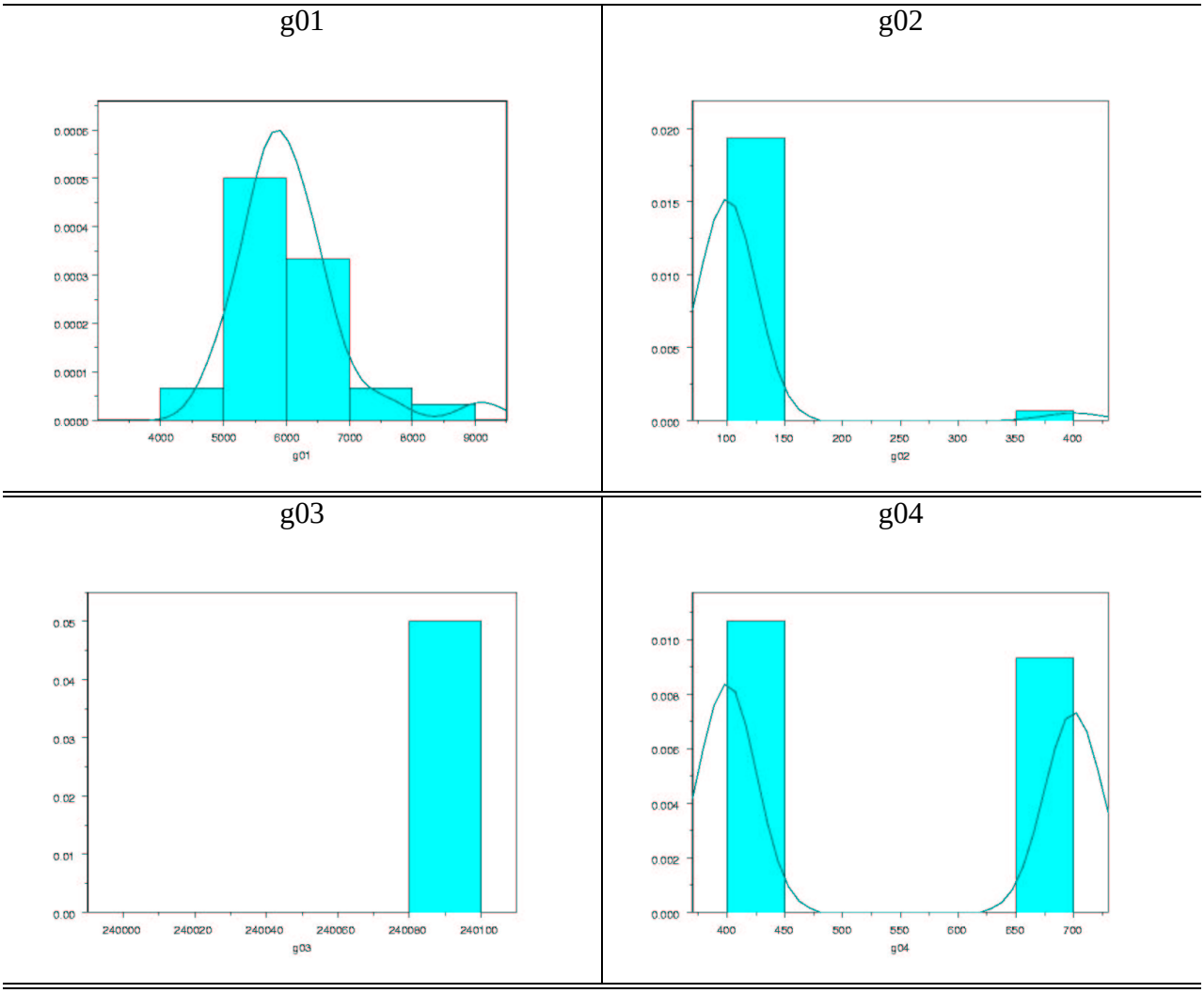


Figure C-15: Histogram and density line of the distribution of results for the ALL-FEASIBLE performance measure obtained by the SMES in 30 independent runs.

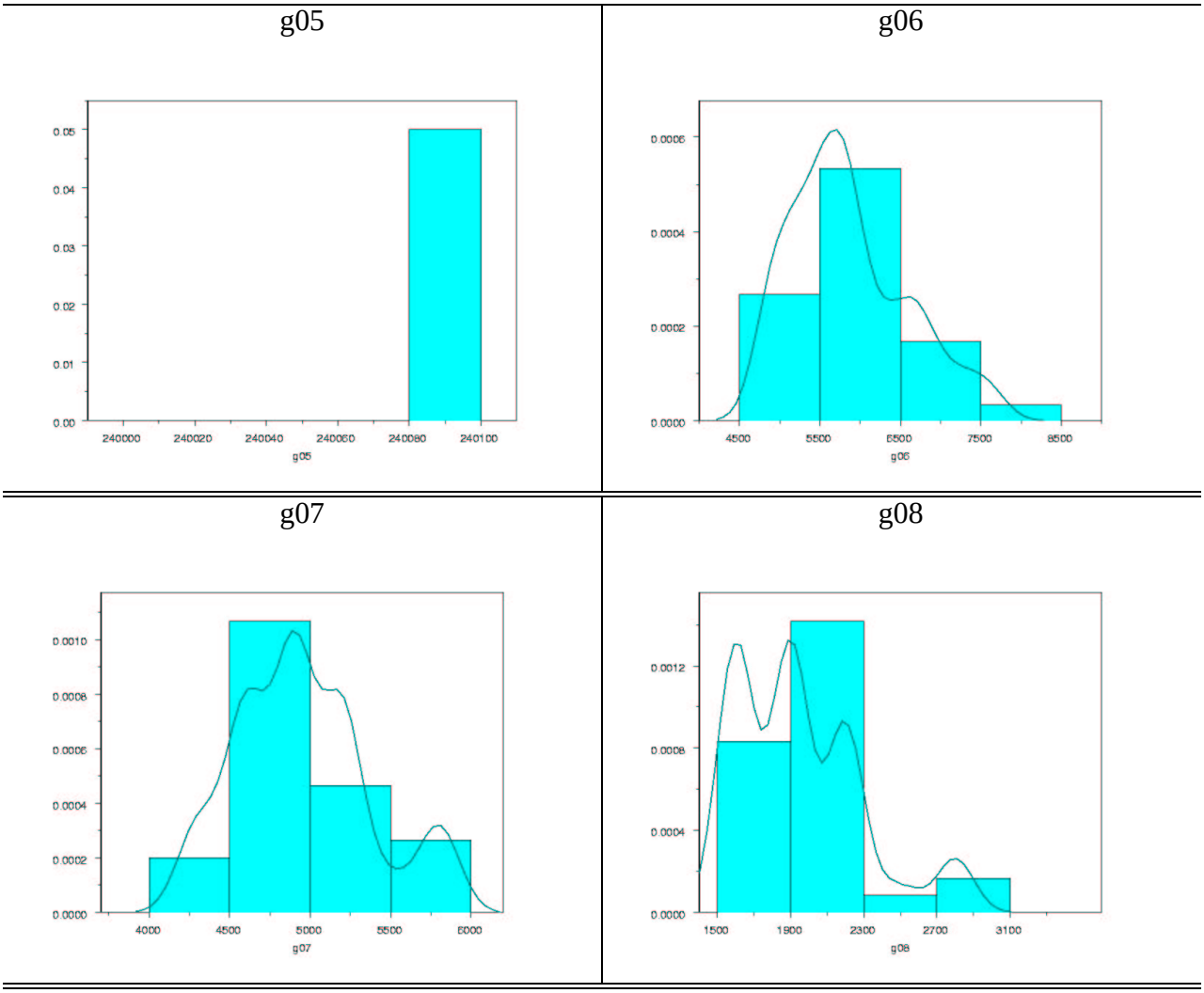


Figure C-16: Histogram and density line of the distribution of results for the ALL-FEASIBLE performance measure obtained by the SMES in 30 independent runs.

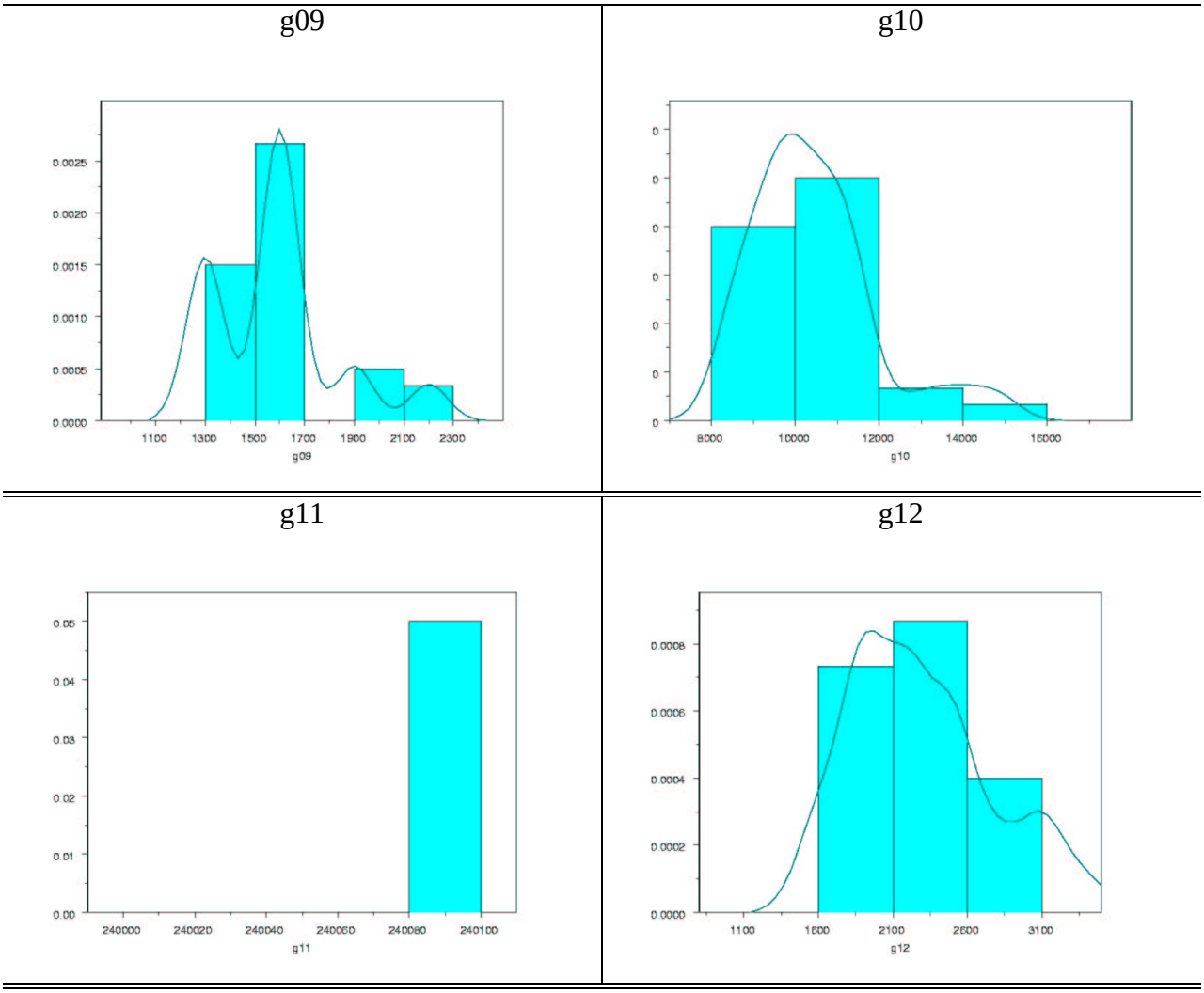


Figure C-17: Histogram and density line of the distribution of results for the ALL-FEASIBLE performance measure obtained by the SMES in 30 independent runs.

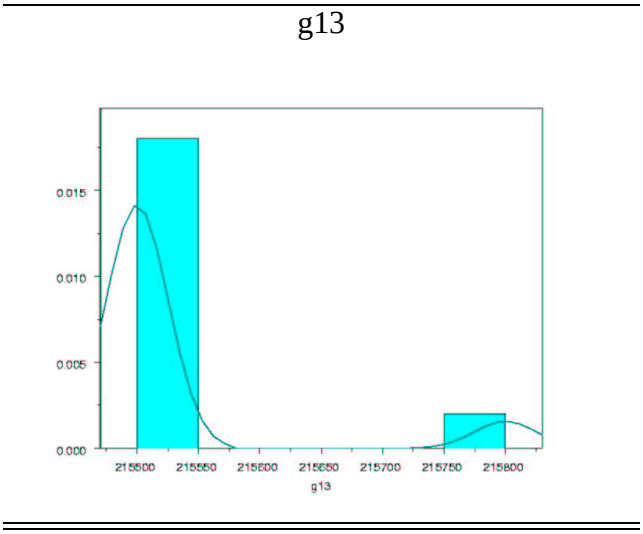


Figure C-18: Histogram and density line of the distribution of results for the ALL-FEASIBLE performance measure obtained by the SMES in 30 independent runs.

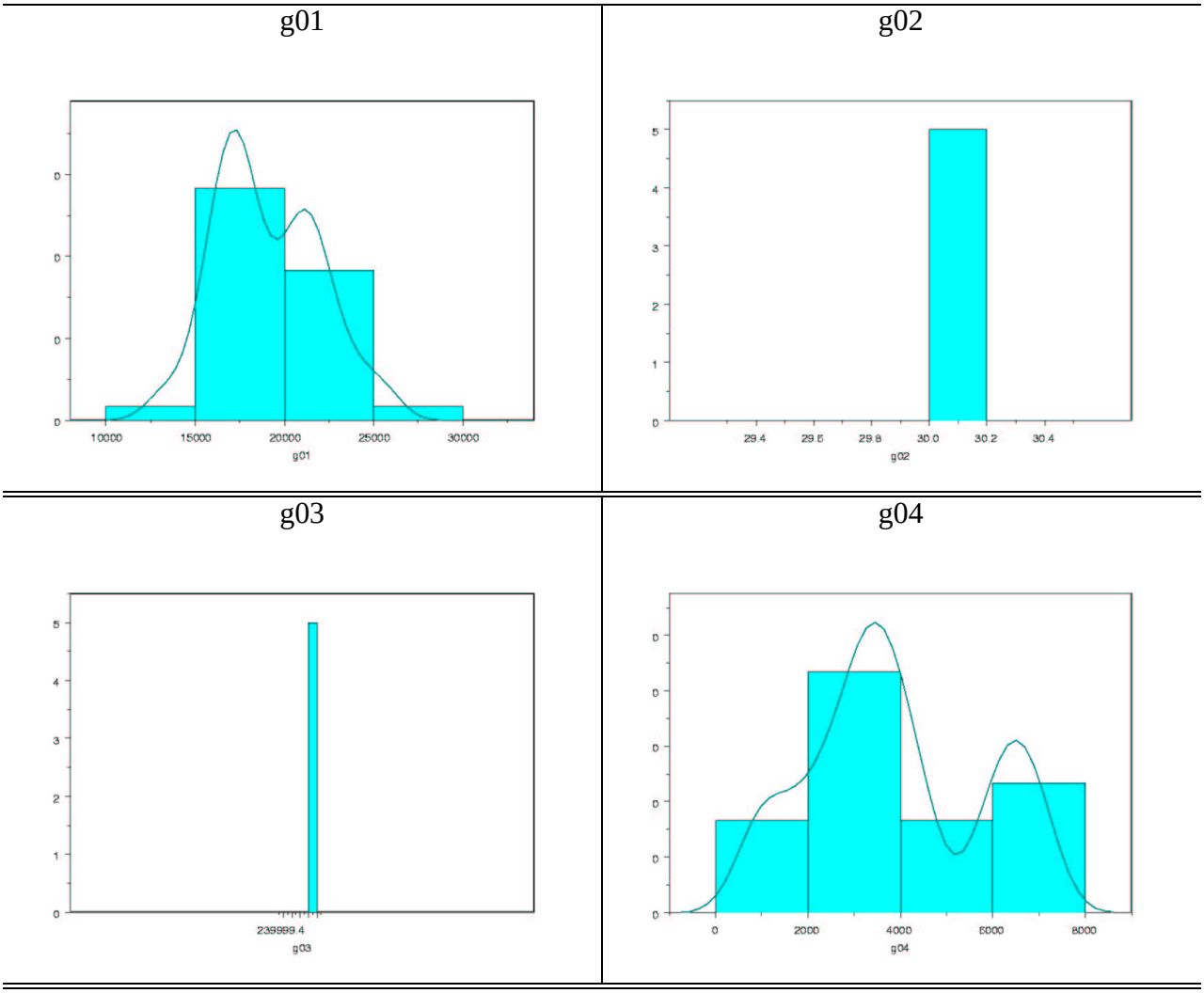


Figure C-19: Histogram and density line of the distribution of results for the ALL-FEASIBLE performance measure obtained by the Stochastic Ranking in 30 independent runs.

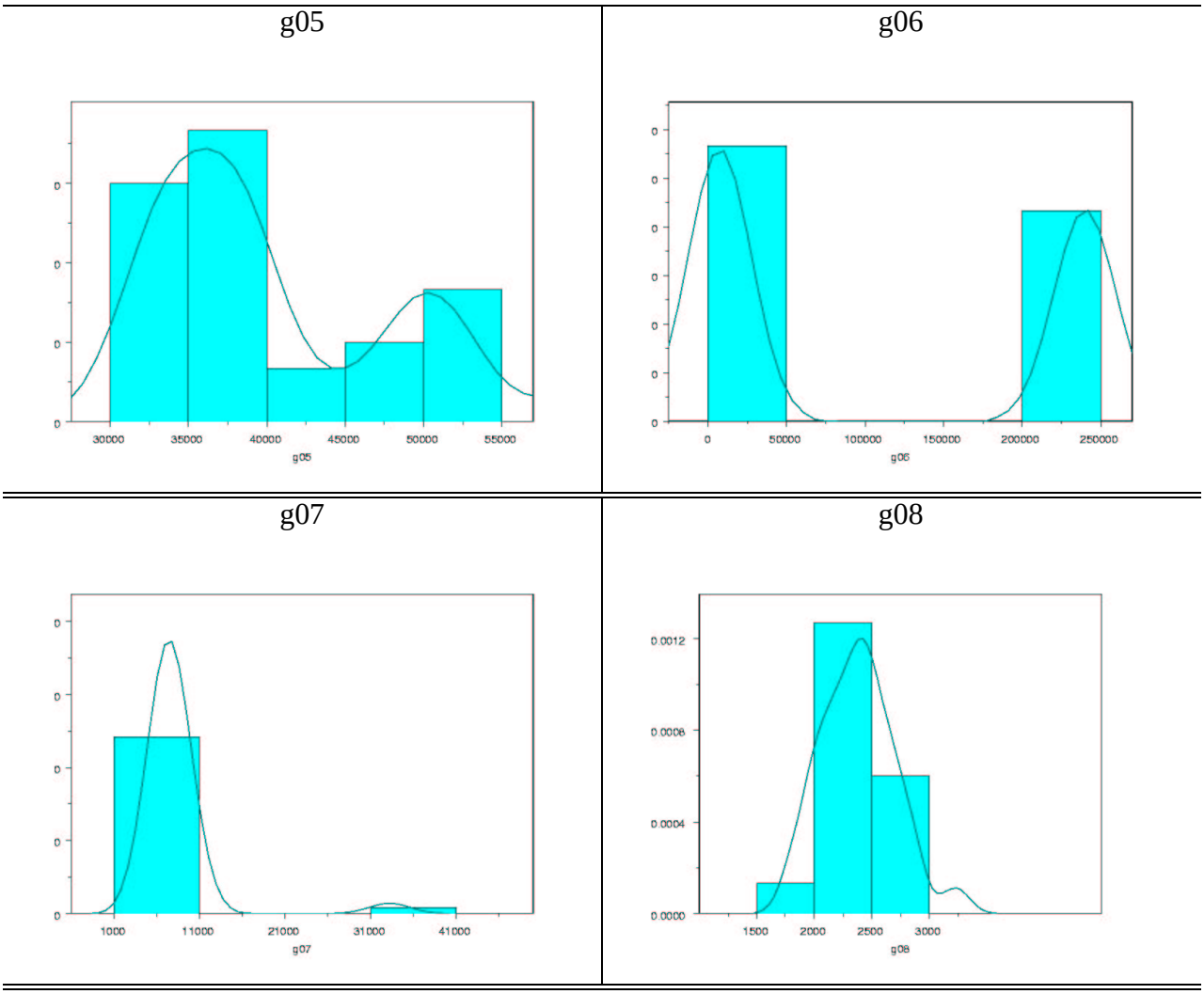


Figure C-20: Histogram and density line of the distribution of results for the ALL-FEASIBLE performance measure obtained by the Stochastic Ranking in 30 independent runs.

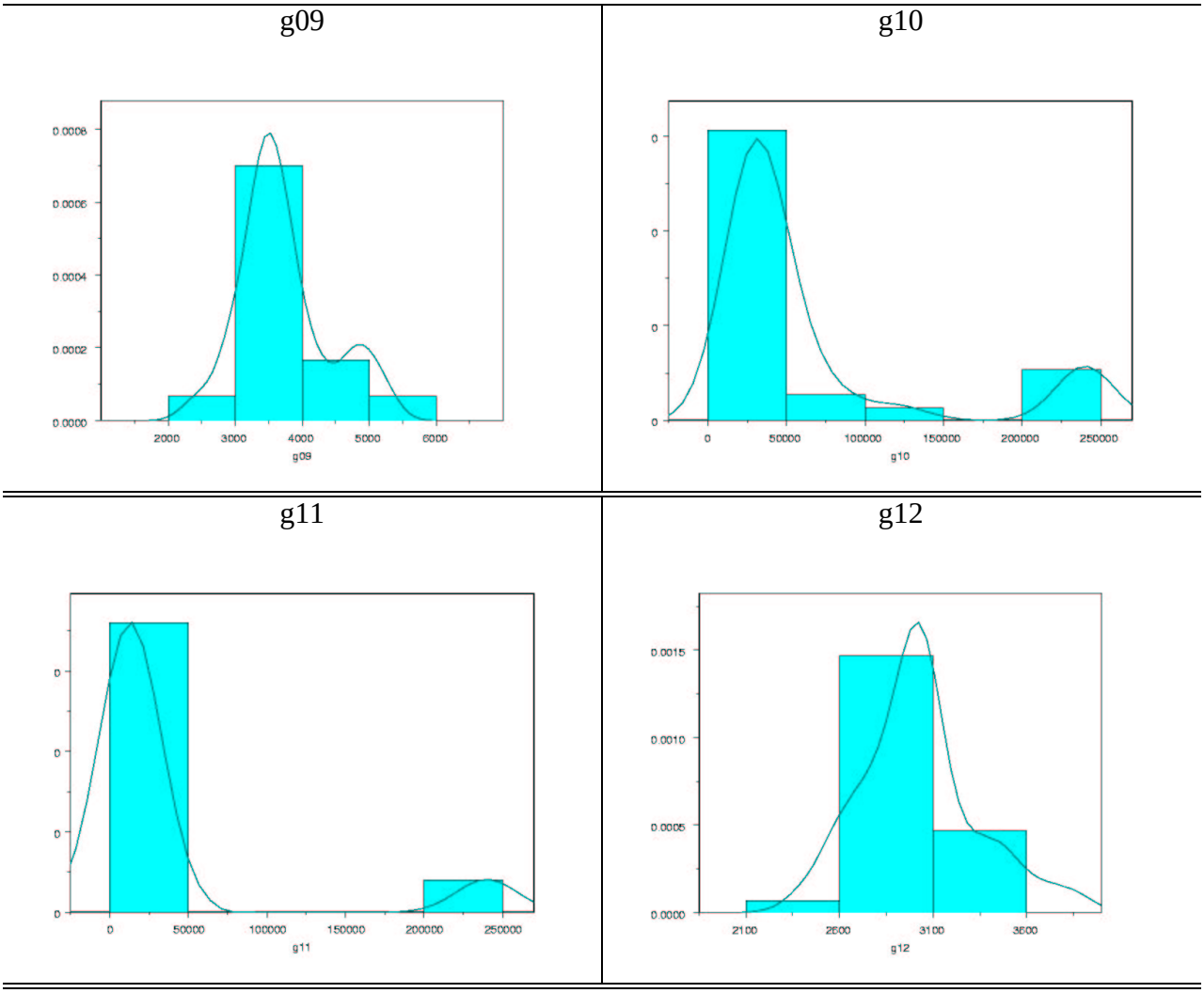


Figure C-21: Histogram and density line of the distribution of results for the ALL-FEASIBLE performance measure obtained by the Stochastic Ranking in 30 independent runs.

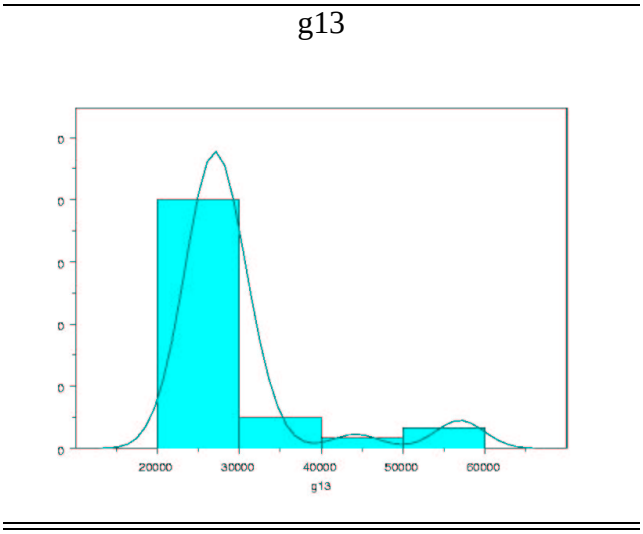


Figure C-22: Histogram and density line of the distribution of results for the ALL-FEASIBLE performance measure obtained by the Stochastic Ranking in 30 independent runs.

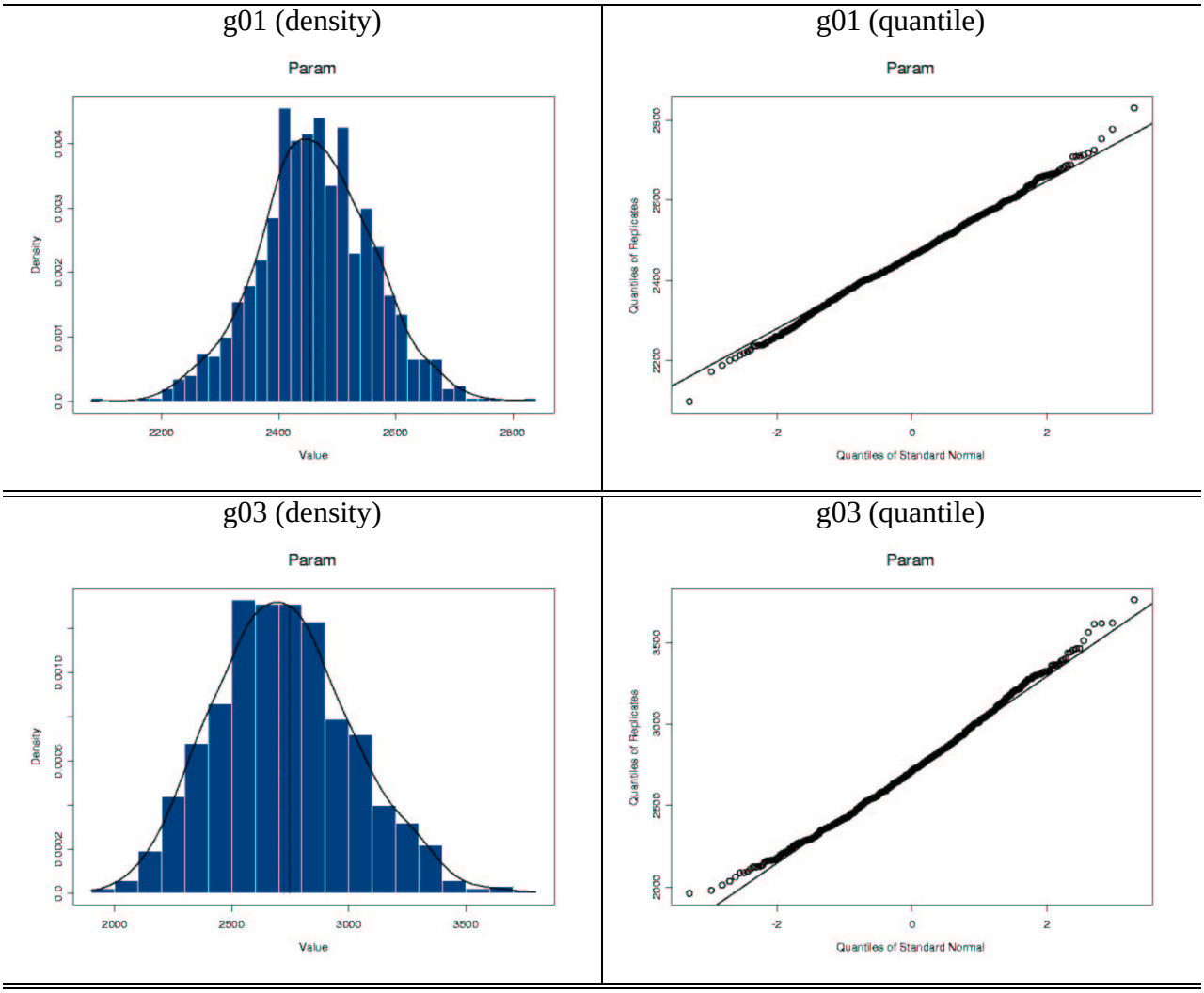
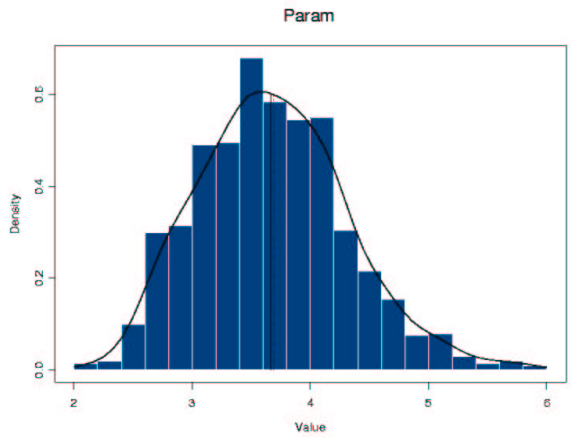
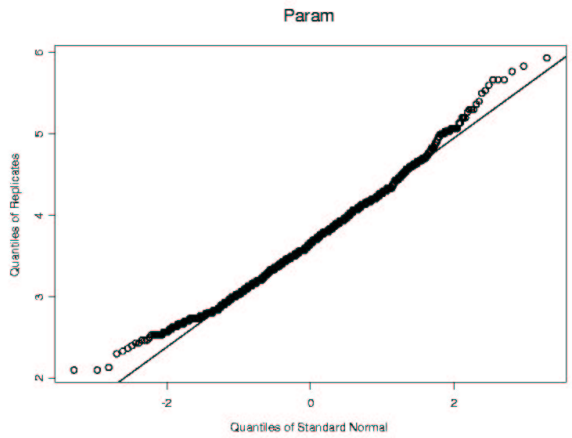


Figure C-23: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the EVALS performance measure. Also shown is the normal quantile graph.

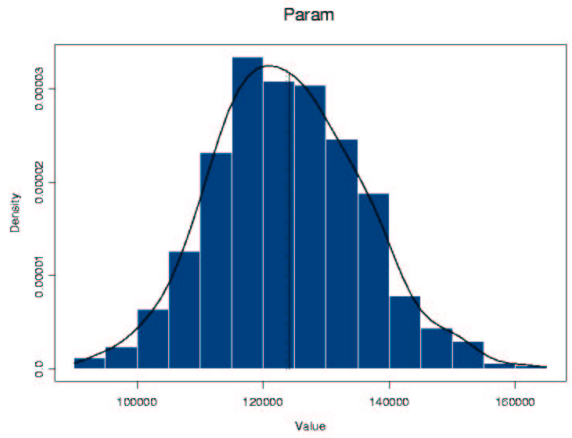
g04 (density)



g04 (quantile)



g05 (density)



g05 (quantile)

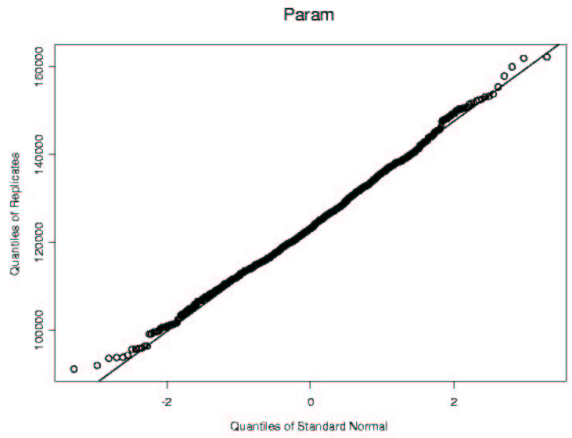
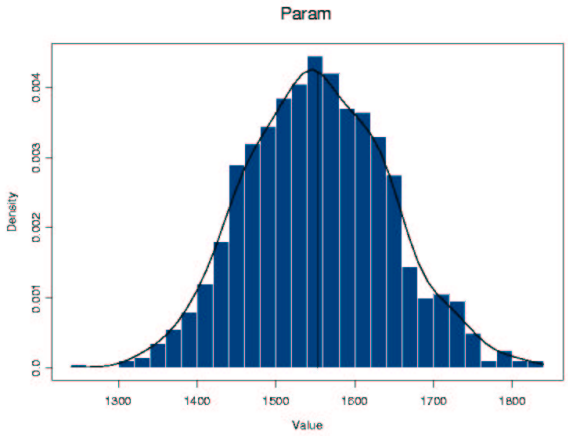
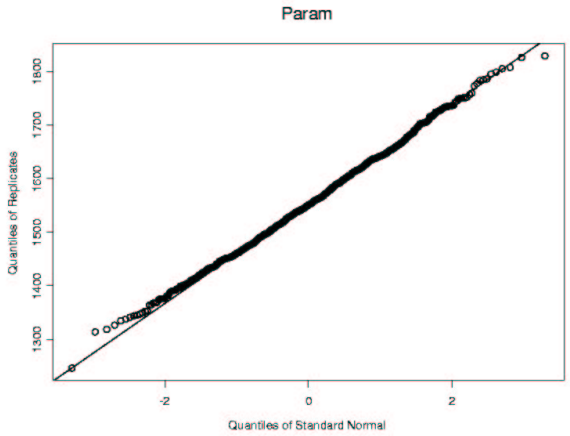


Figure C-24: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the EVALS performance measure. Also shown is the normal quantile graph.

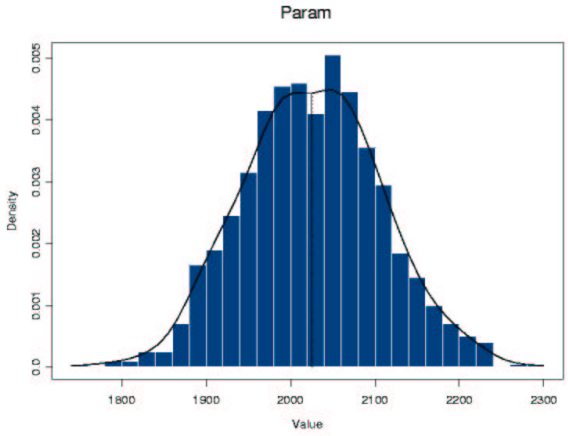
g06 (density)



g06 (quantile)



g07 (density)



g07 (quantile)

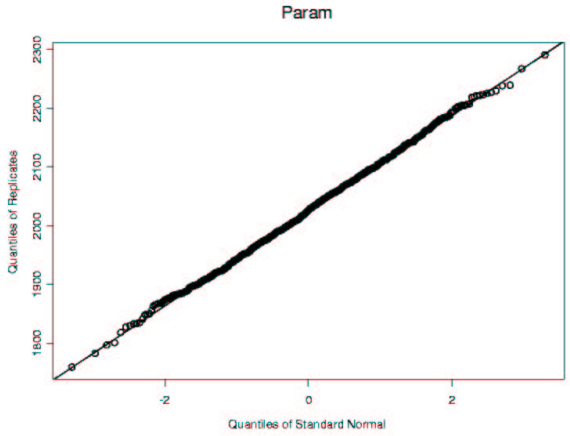


Figure C-25: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the EVALS performance measure. Also shown is the normal quantile graph.

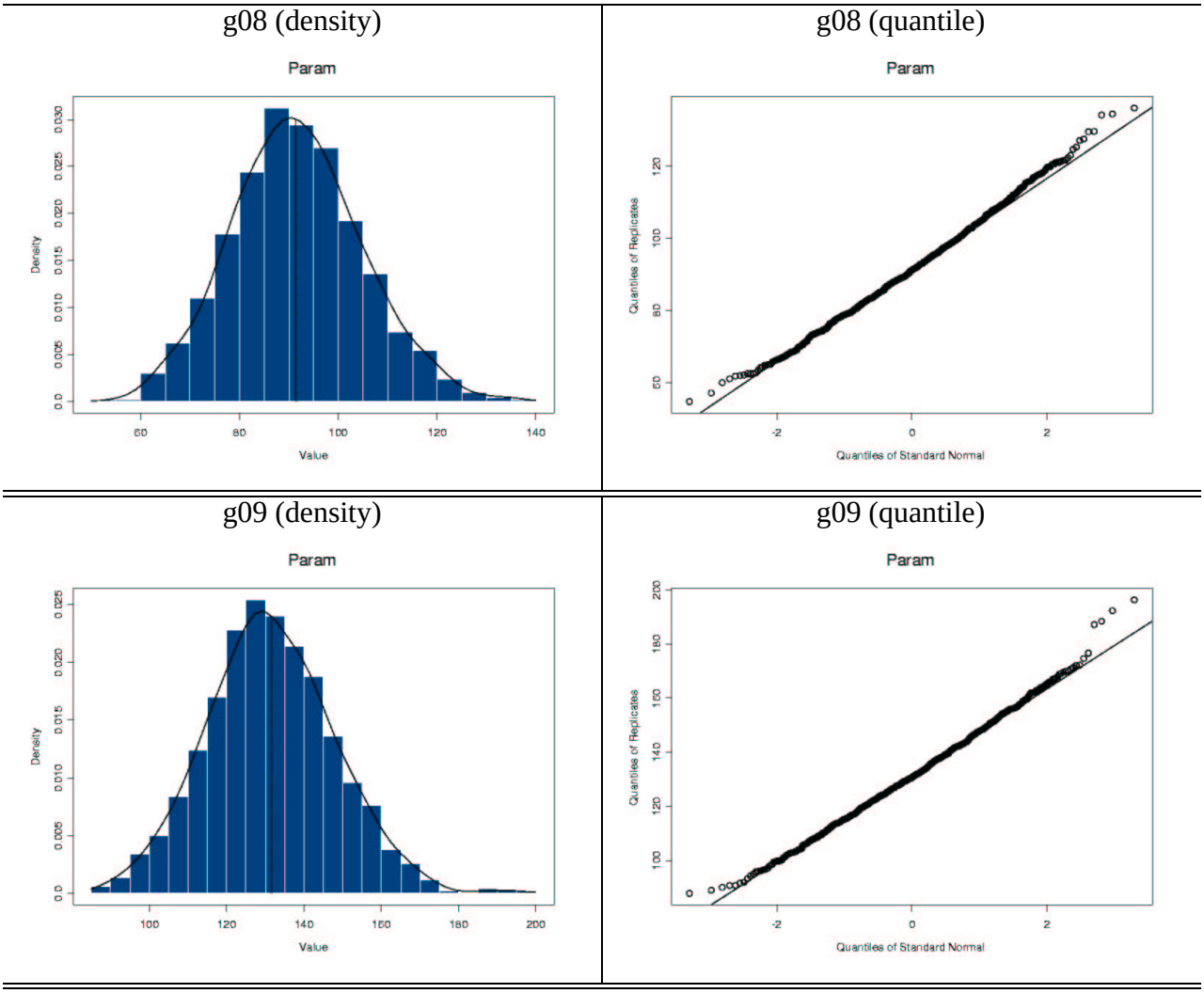


Figure C-26: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the EVALS performance measure. Also shown is the normal quantile graph.

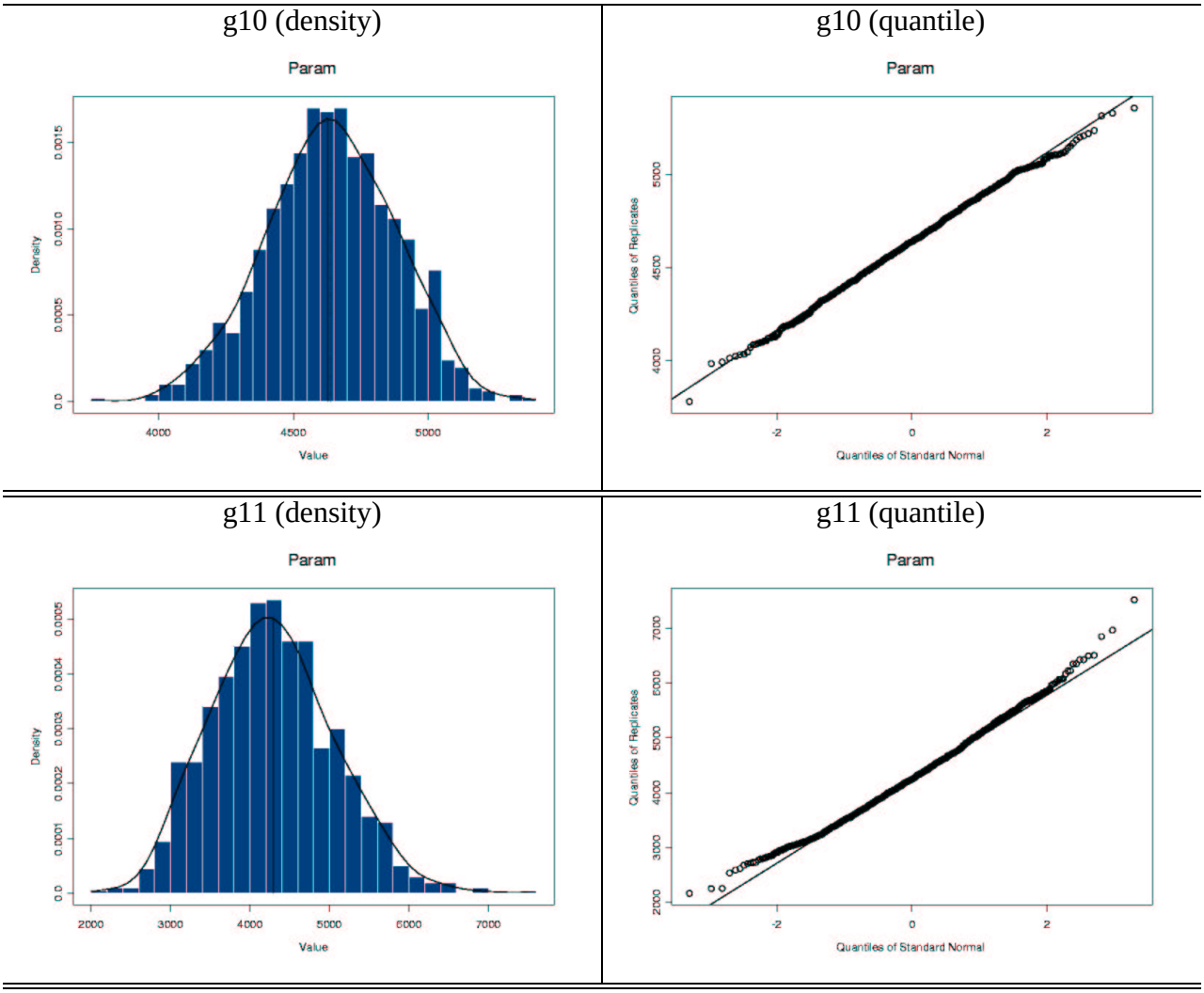
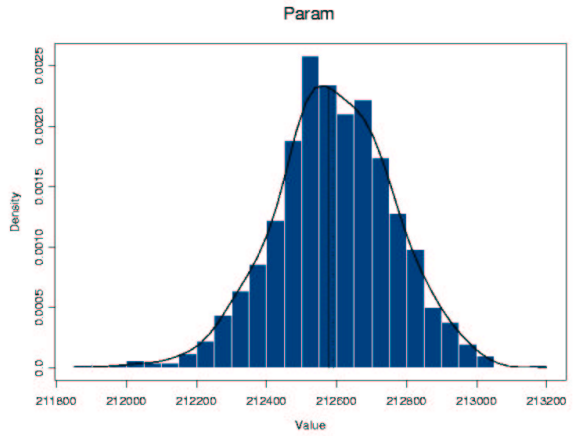
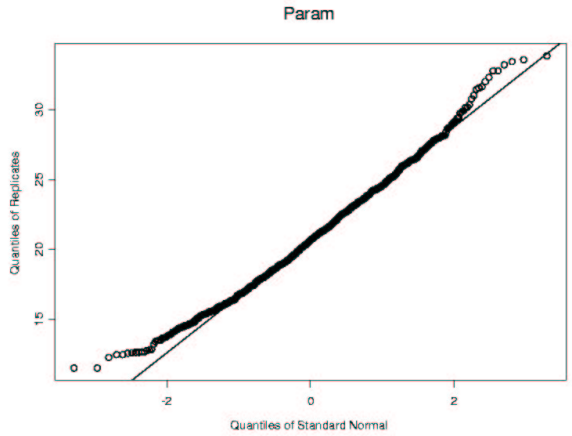


Figure C-27: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the EVALS performance measure. Also shown is the normal quantile graph.

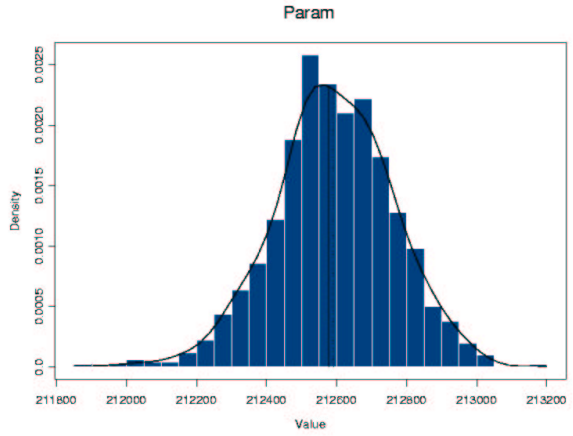
g12 (density)



g12 (quantile)



g13 (density)



g13 (quantile)

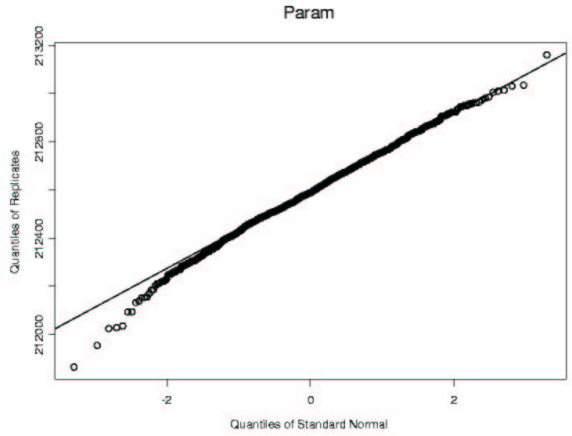


Figure C-28: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the EVALS performance measure. Also shown is the normal quantile graph.

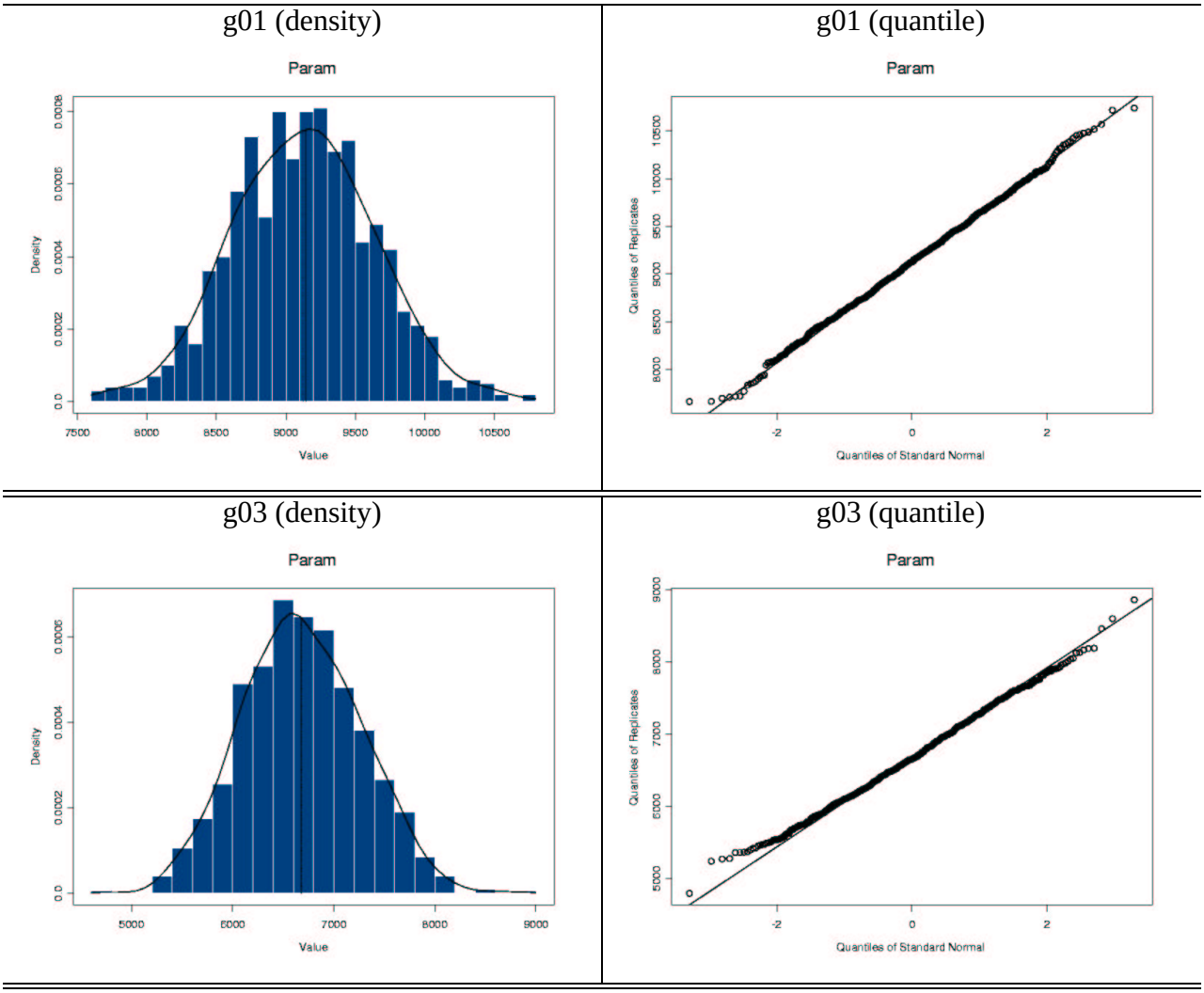
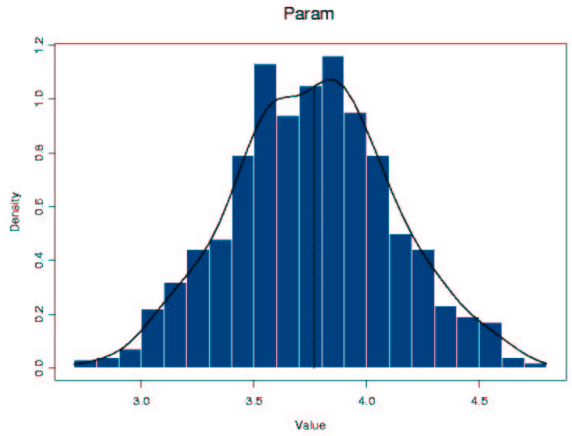
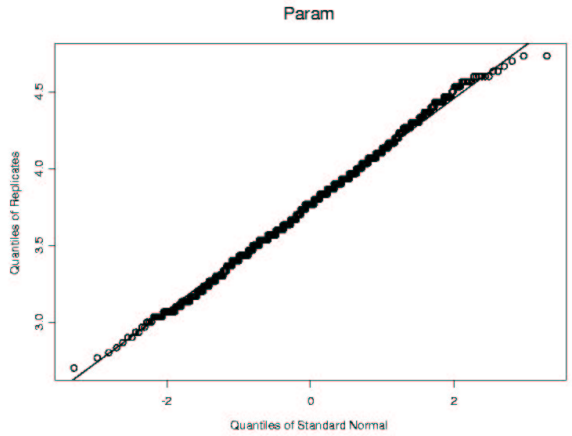


Figure C-29: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the EVALS performance measure. Also shown is the normal quantile graph.

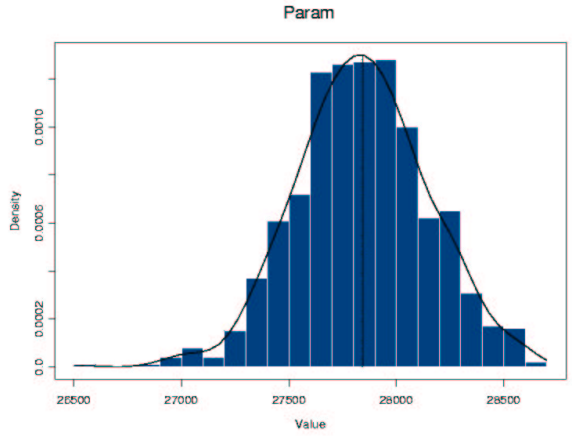
g04 (density)



g04 (quantile)



g05 (density)



g05 (quantile)

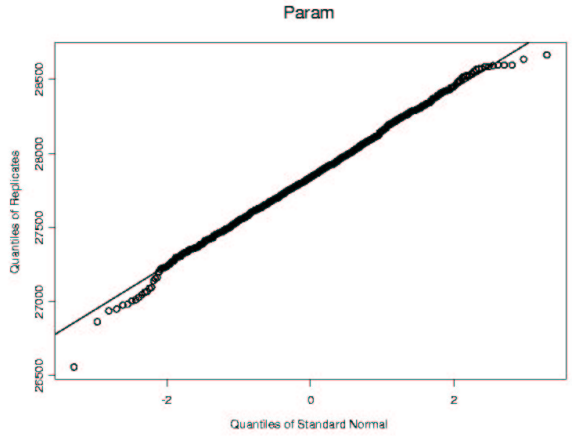


Figure C-30: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the EVALS performance measure. Also shown is the normal quantile graph.

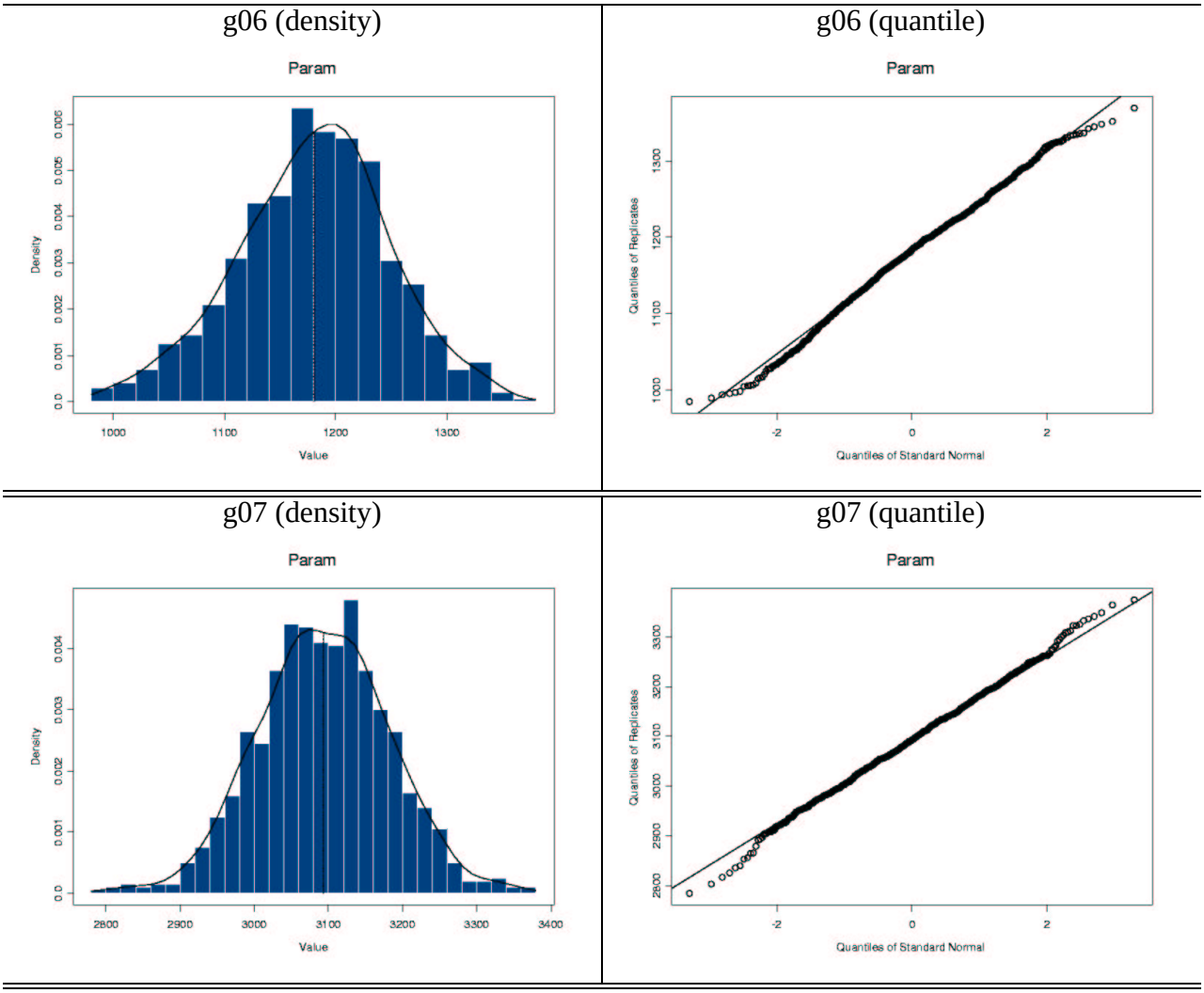
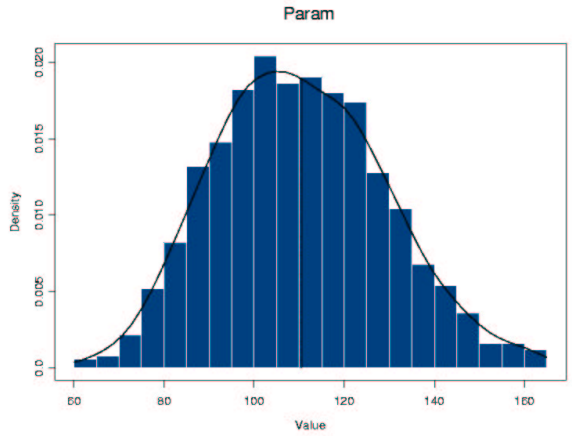
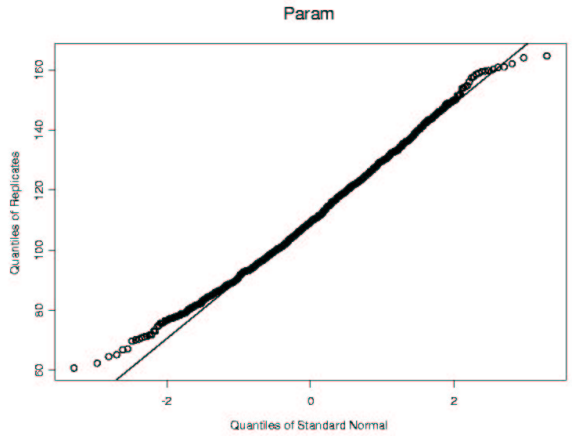


Figure C-31: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the EVALS performance measure. Also shown is the normal quantile graph.

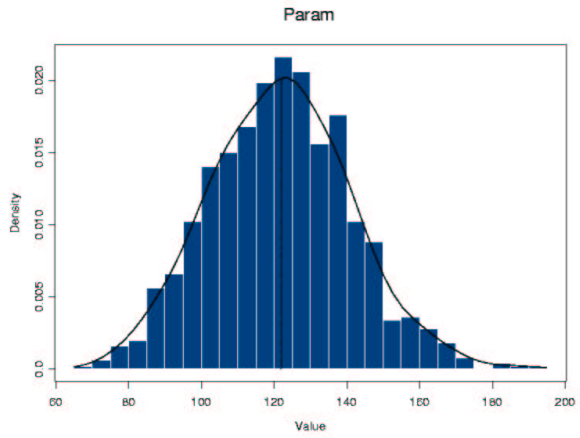
g08 (density)



g08 (quantile)



g09 (density)



g09 (quantile)

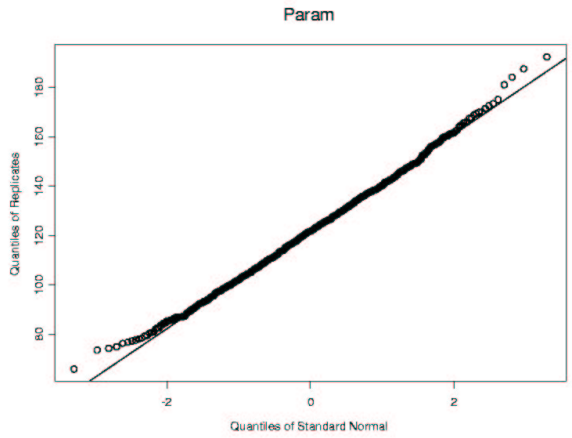


Figure C-32: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the EVALS performance measure. Also shown is the normal quantile graph.

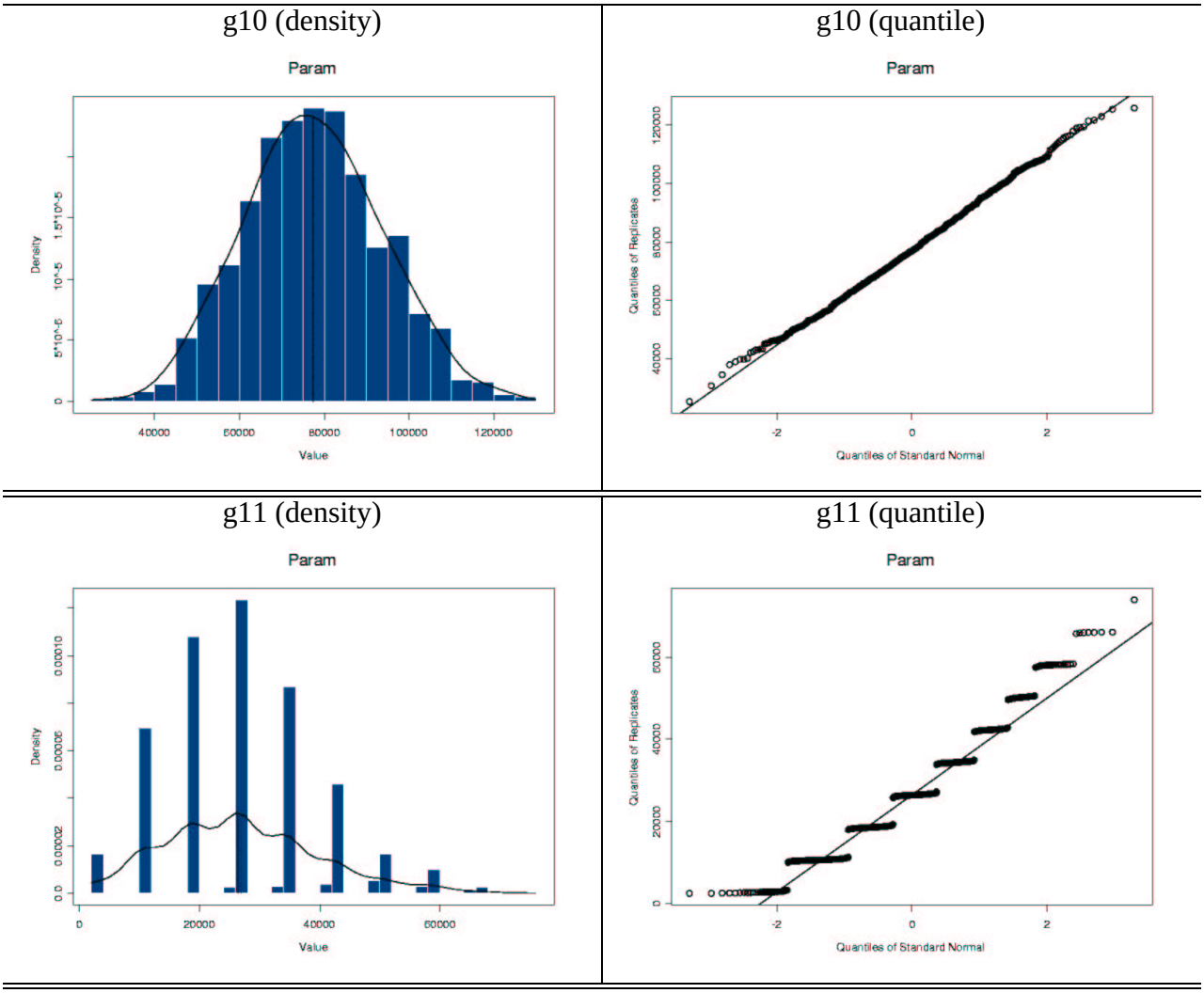
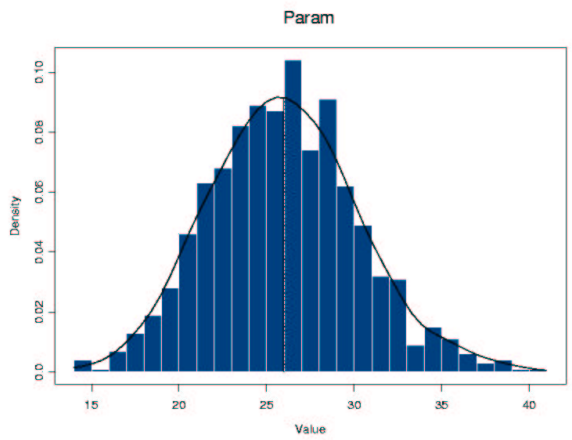
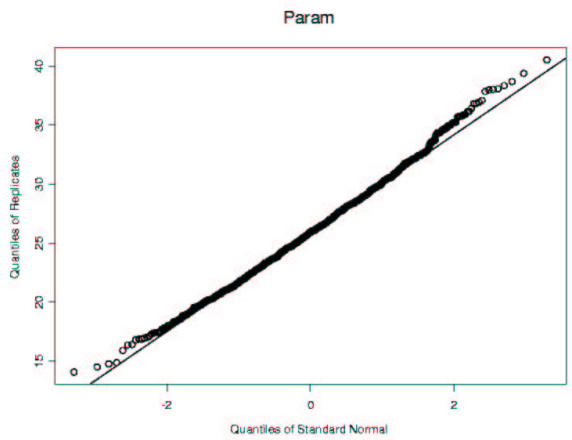


Figure C-33: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the EVALS performance measure. Also shown is the normal quantile graph.

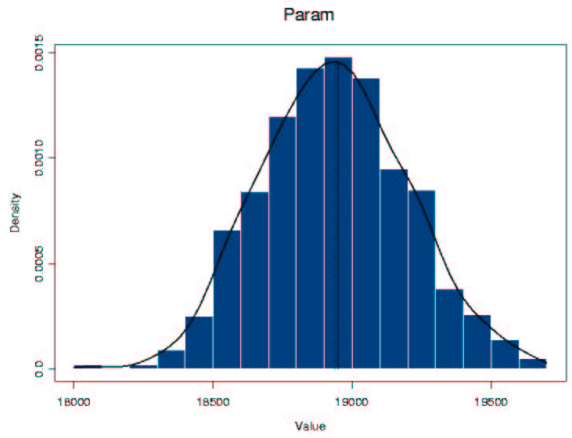
g12 (density)



g12 (quantile)



g13 (density)



g13 (quantile)

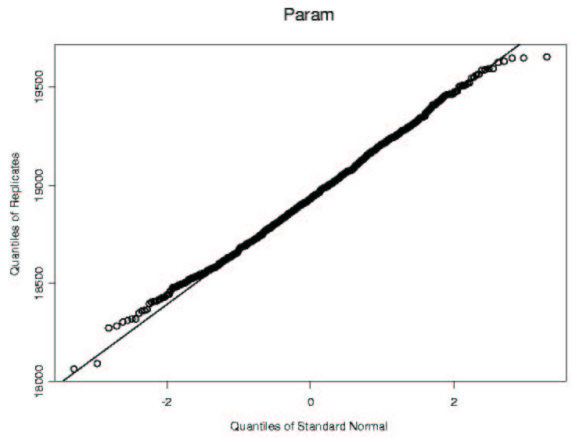
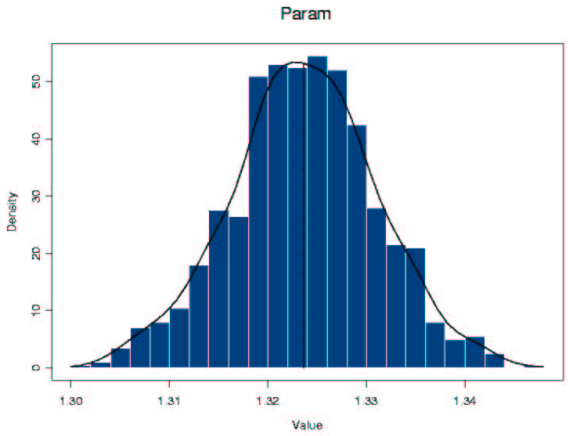
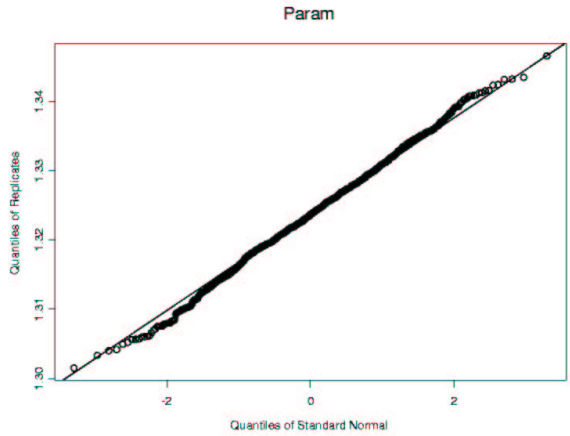


Figure C-34: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the EVALS performance measure. Also shown is the normal quantile graph.

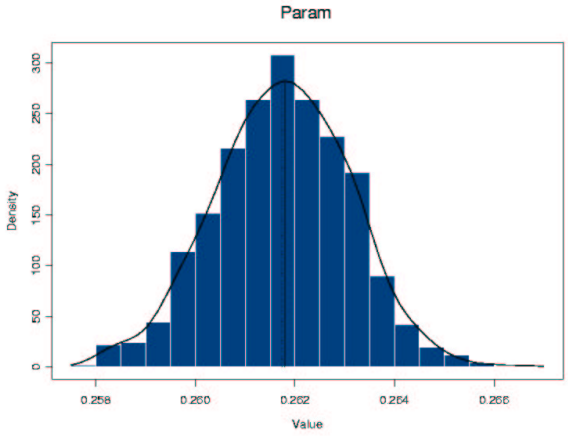
g01 (density)



g01 (quantile)



g02 (density)



g02 (quantile)

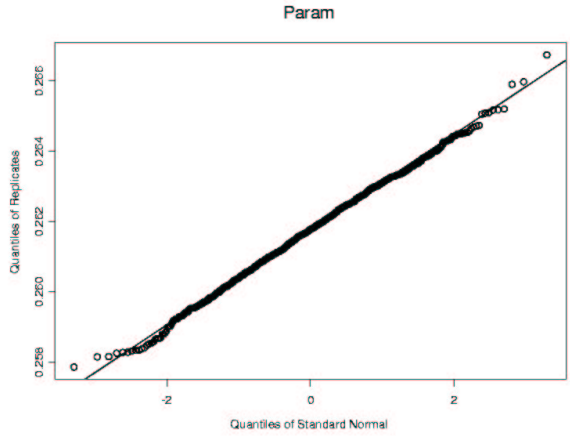
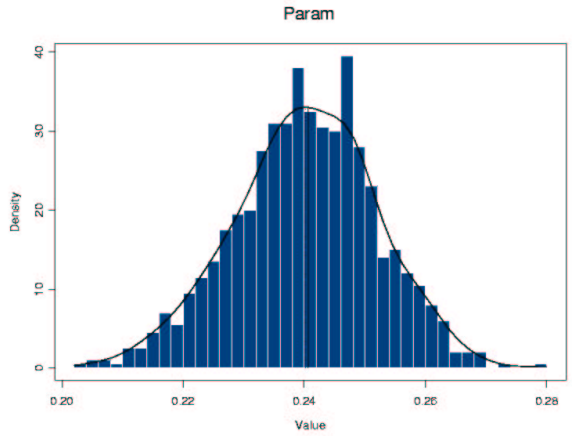
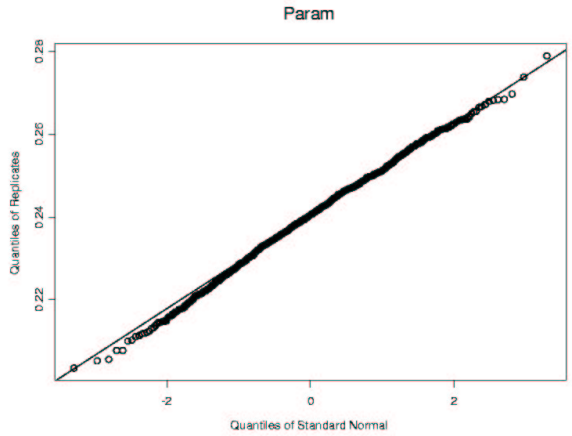


Figure C-35: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

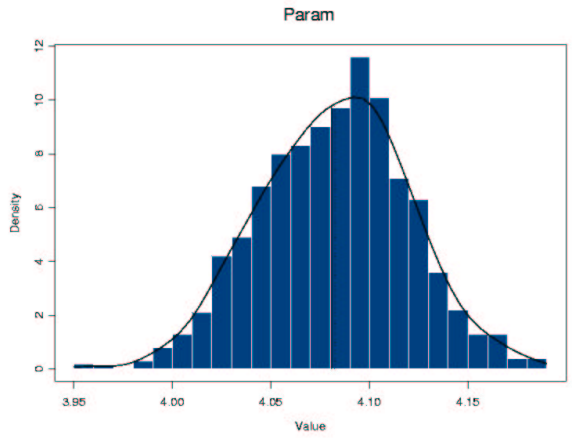
g03 (density)



g03 (quantile)



g04 (density)



g04 (quantile)

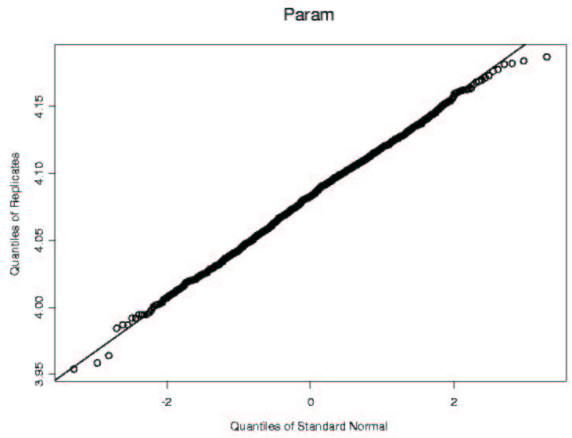


Figure C-36: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

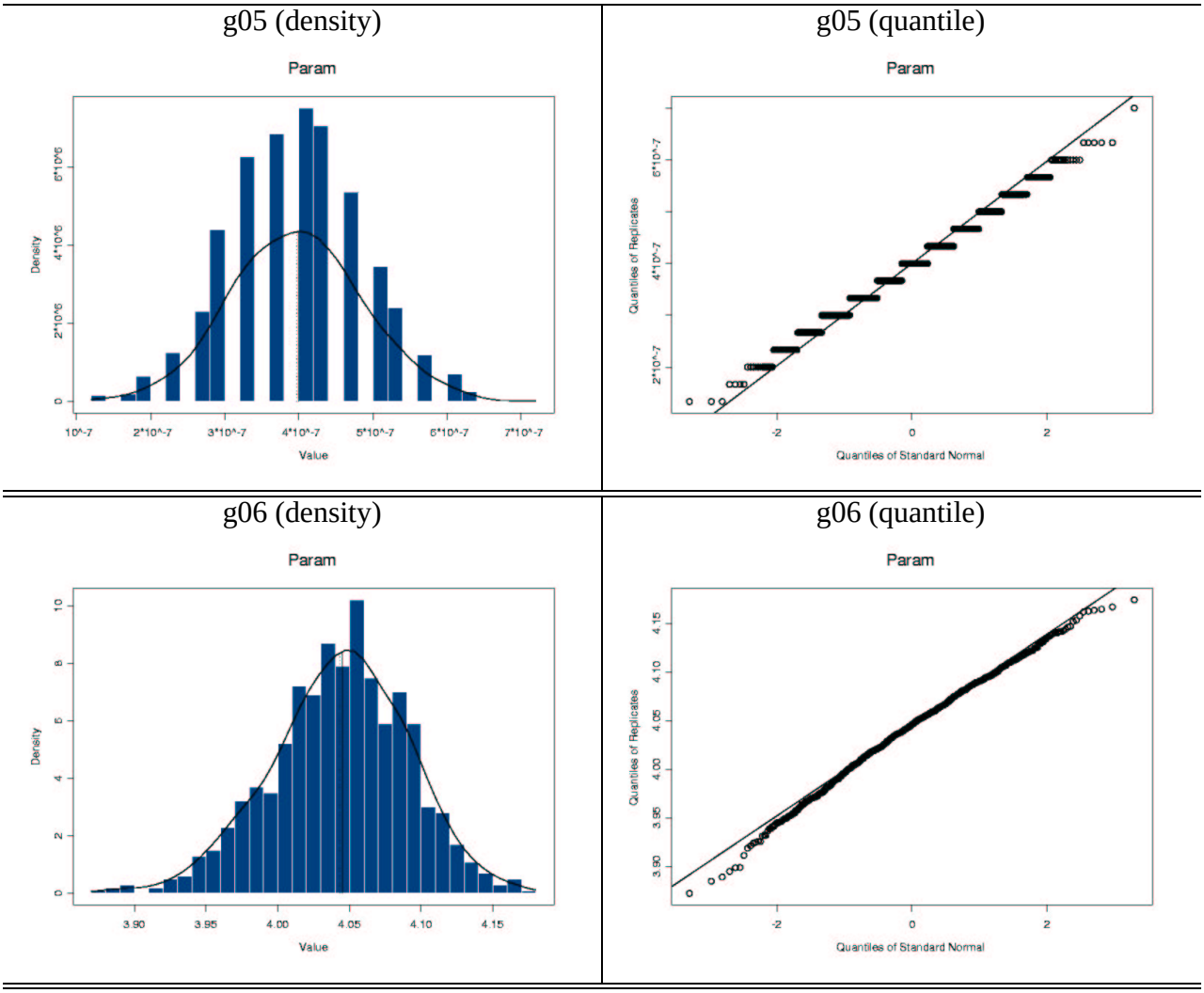
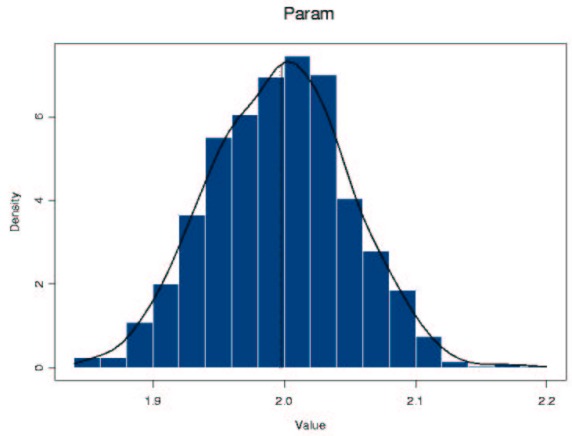
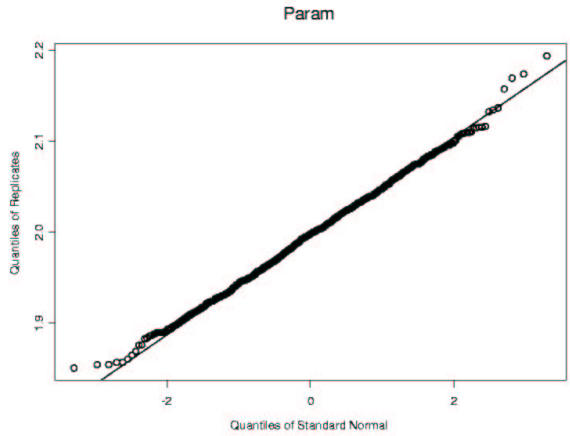


Figure C-37: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

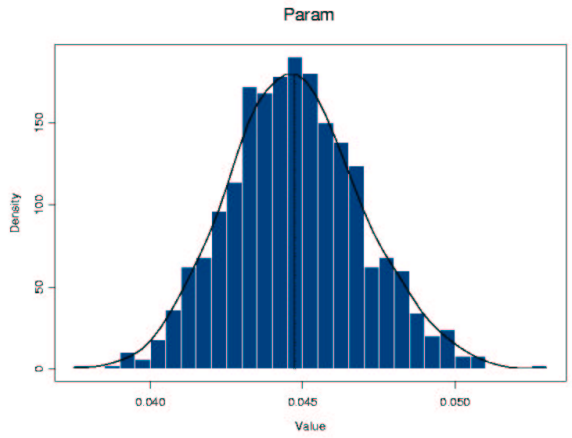
g07 (density)



g07 (quantile)



g08 (density)



g08 (quantile)

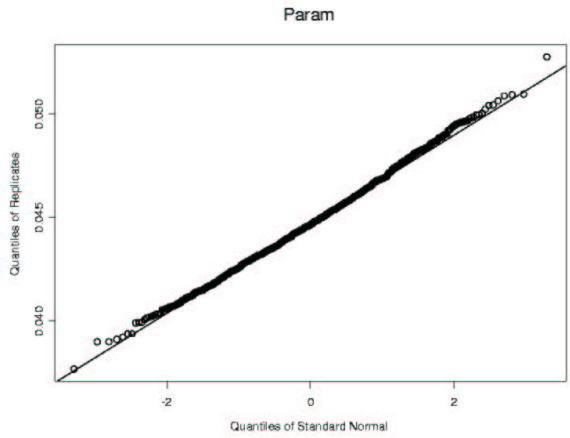


Figure C-38: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

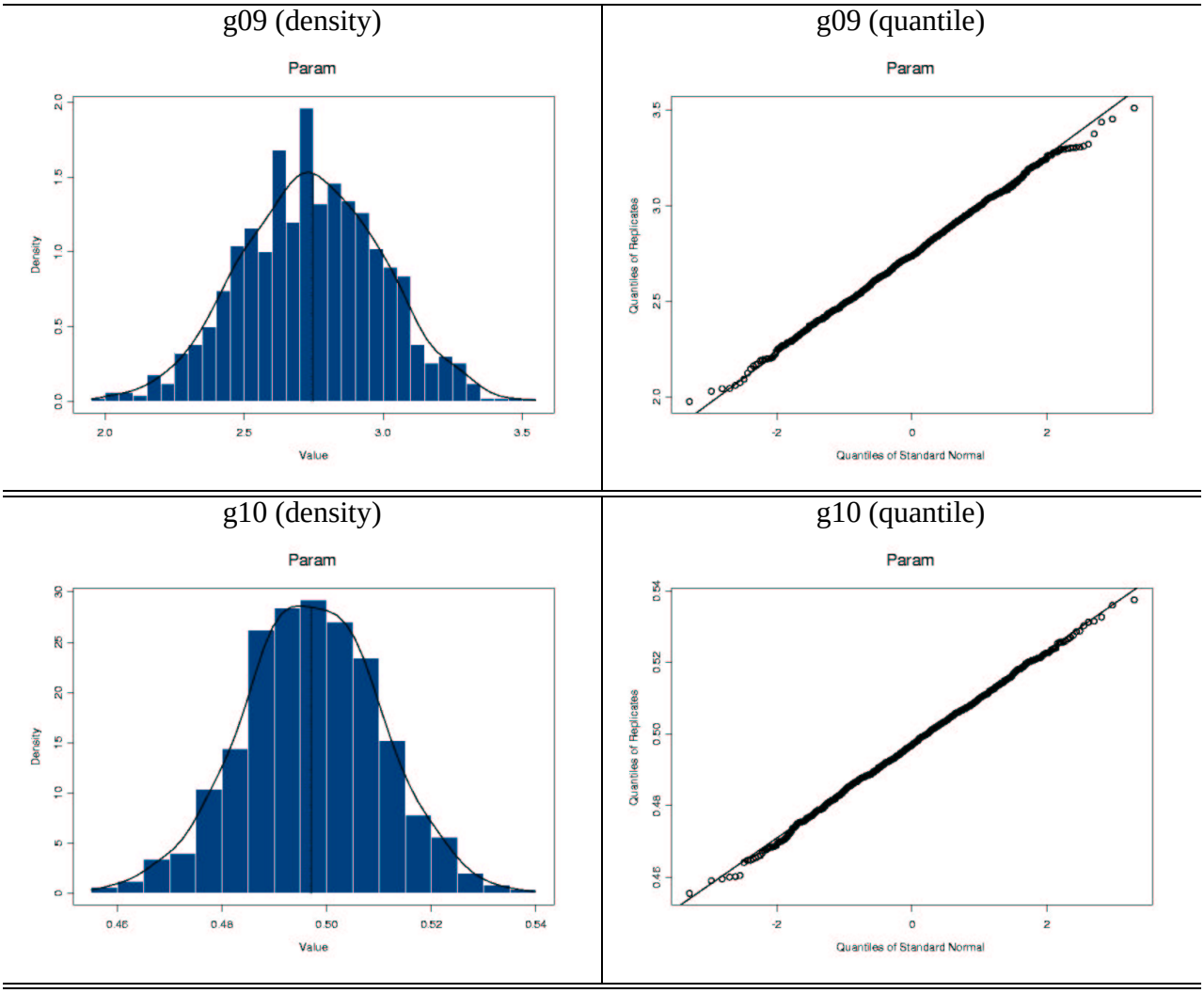
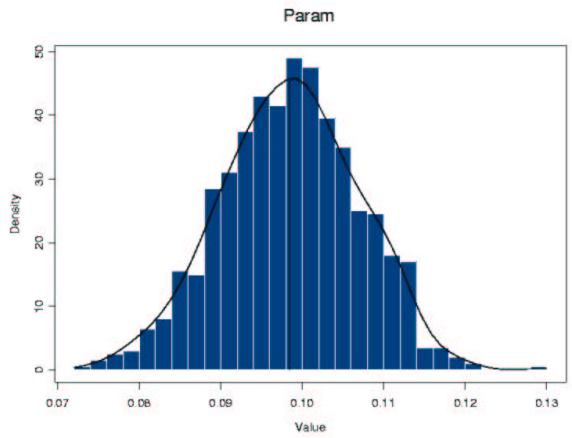
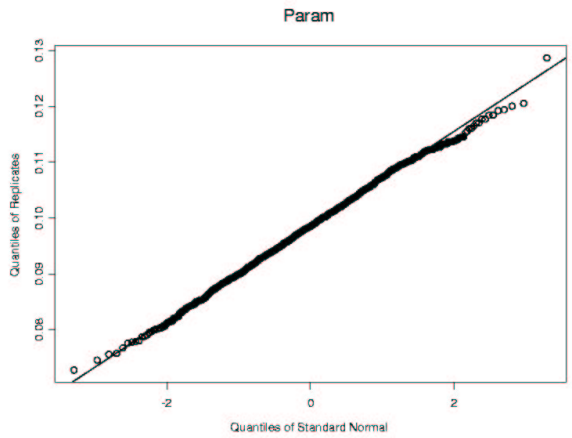


Figure C-39: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

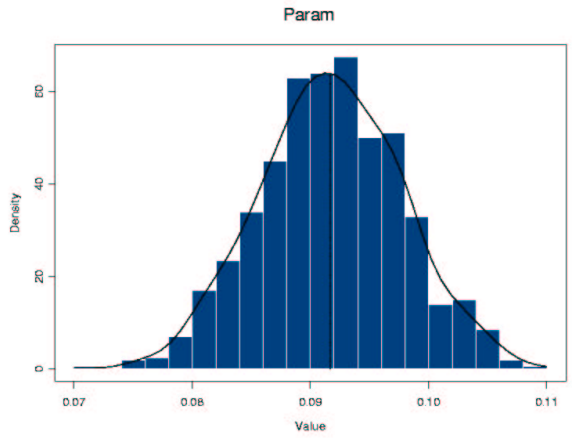
g11 (density)



g11 (quantile)



g12 (density)



g12 (quantile)

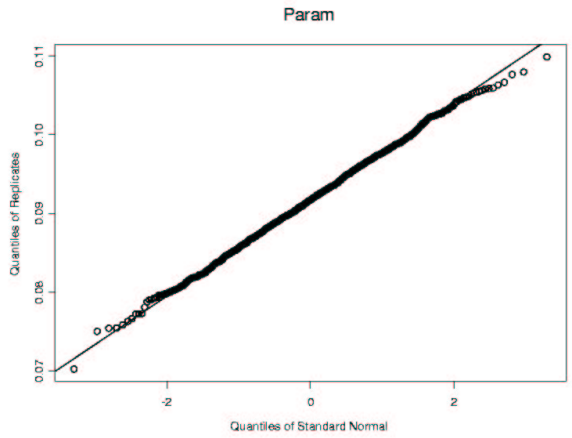
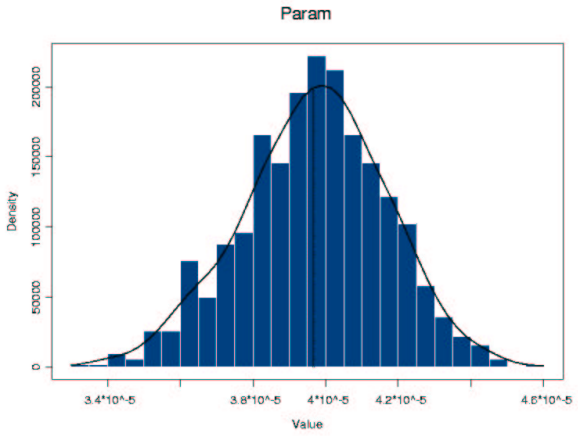


Figure C-40: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

g13 (density)



g13 (quantile)

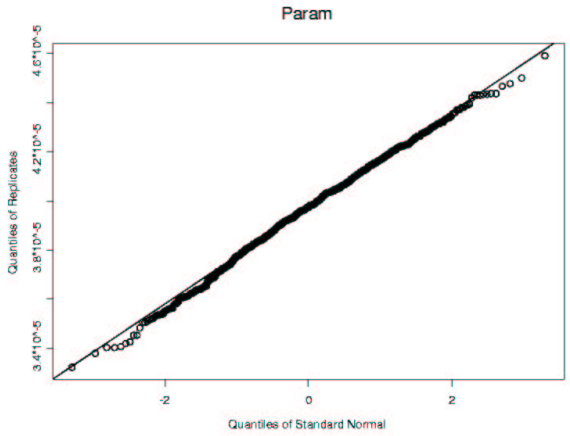
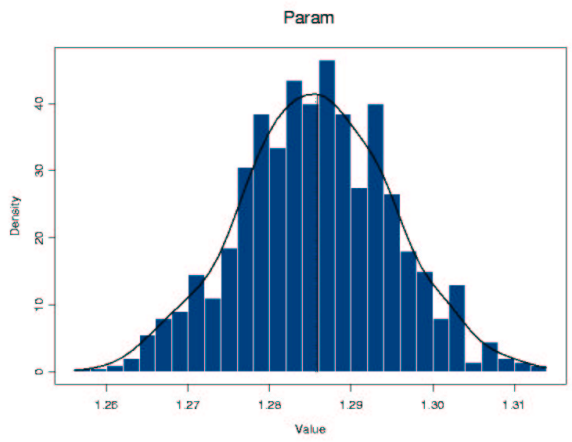
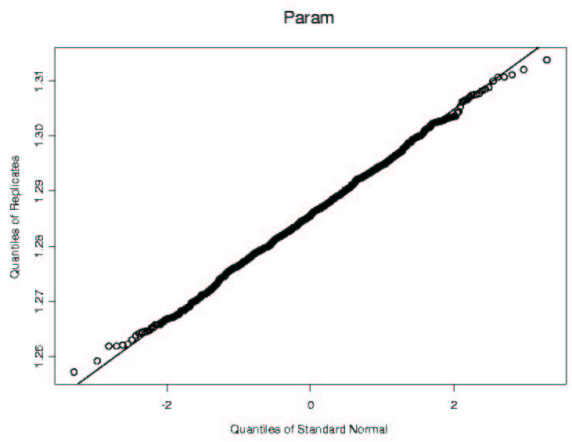


Figure C-41: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

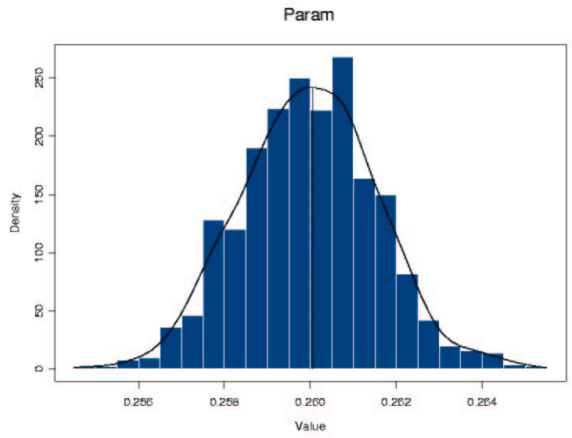
g01 (density)



g01 (quantile)



g02 (density)



g02 (quantile)

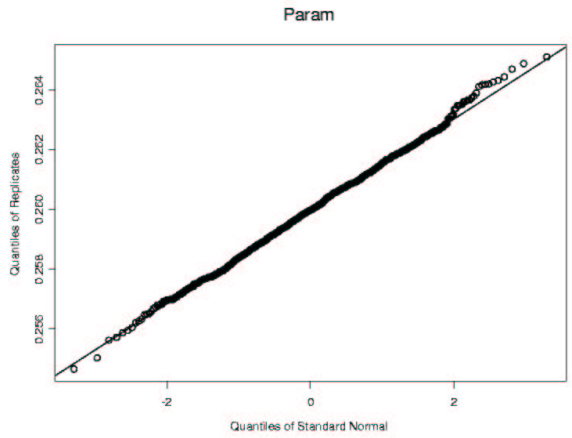


Figure C-42: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

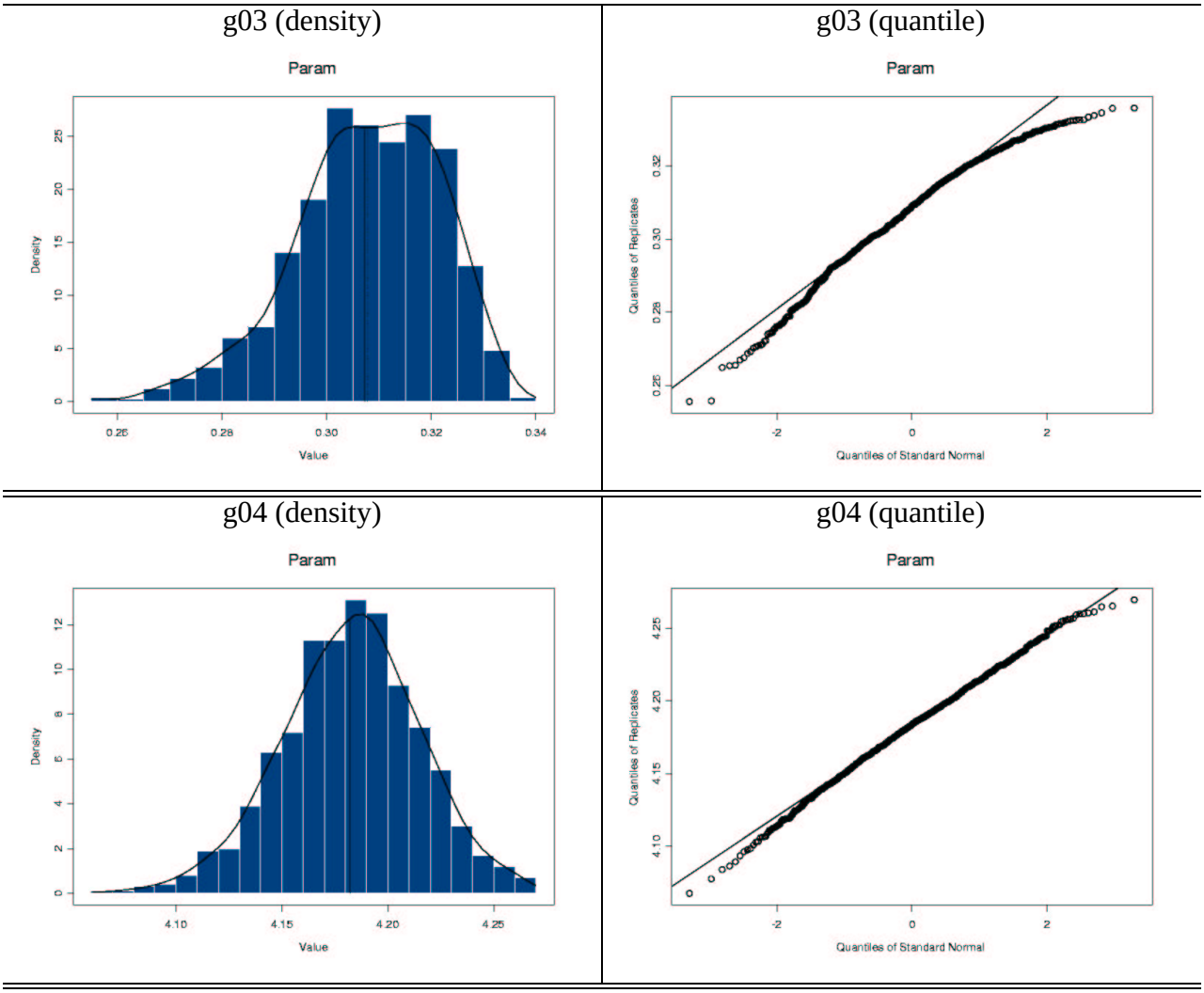


Figure C-43: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

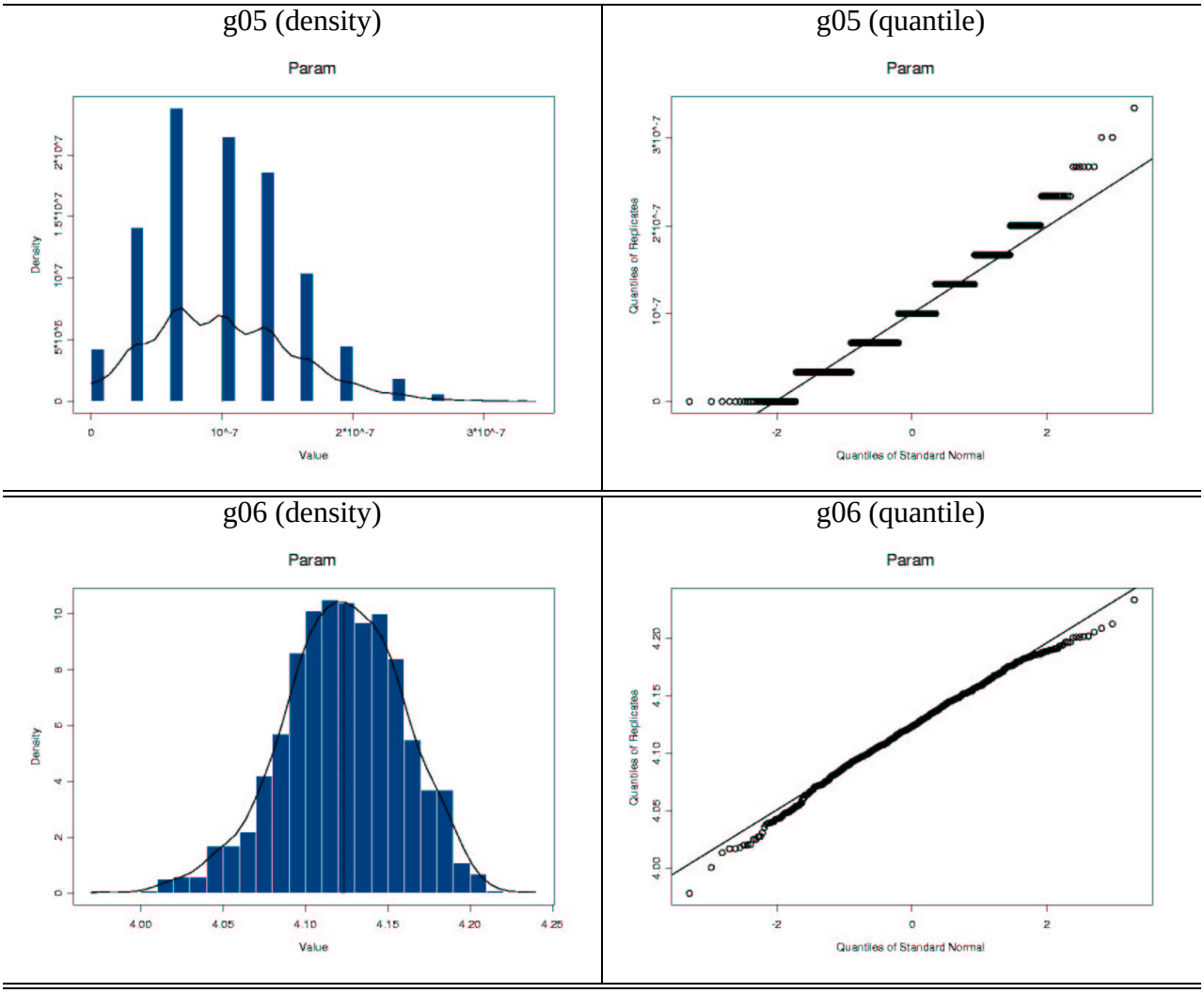


Figure C-44: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

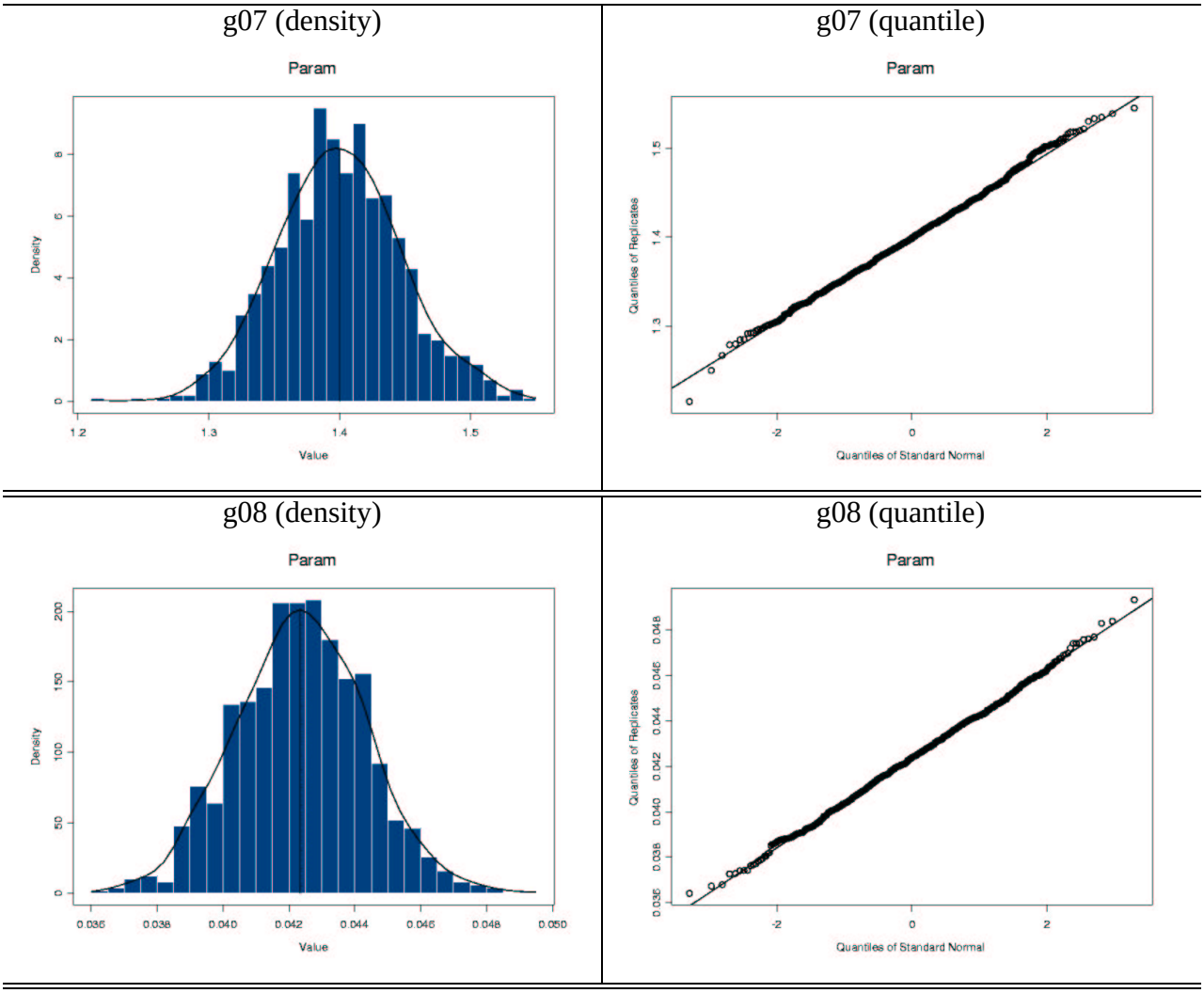
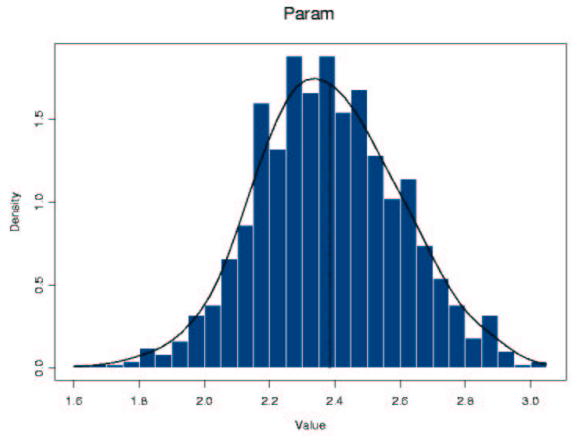
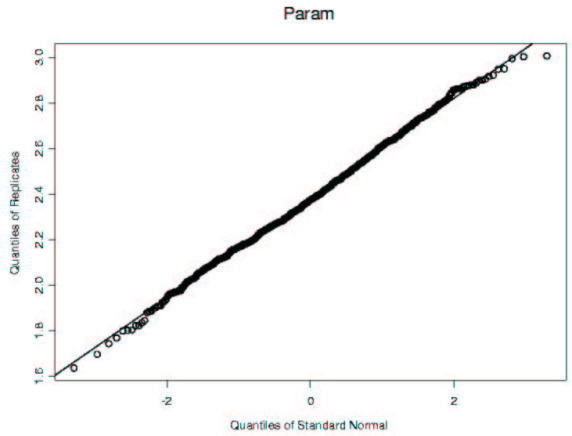


Figure C-45: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

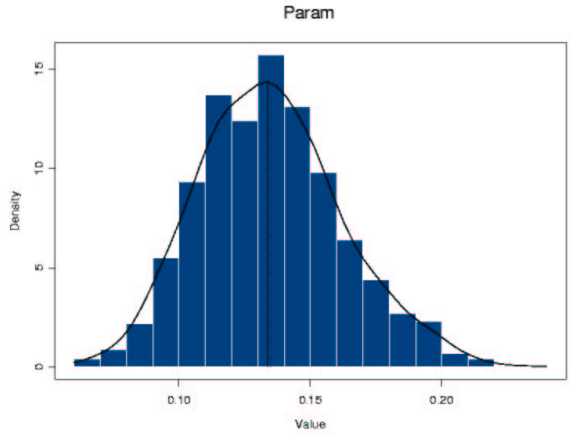
g09 (density)



g09 (quantile)



g10 (density)



g10 (quantile)

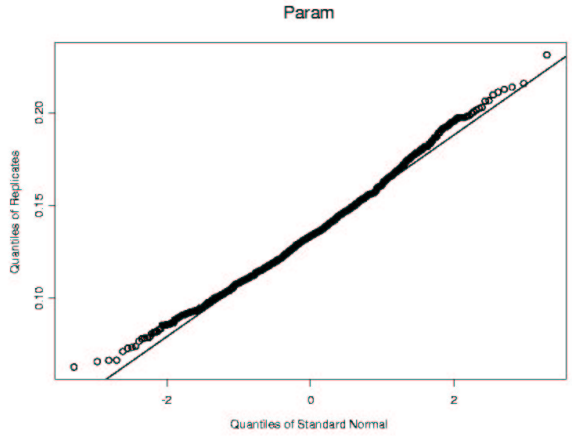


Figure C-46: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

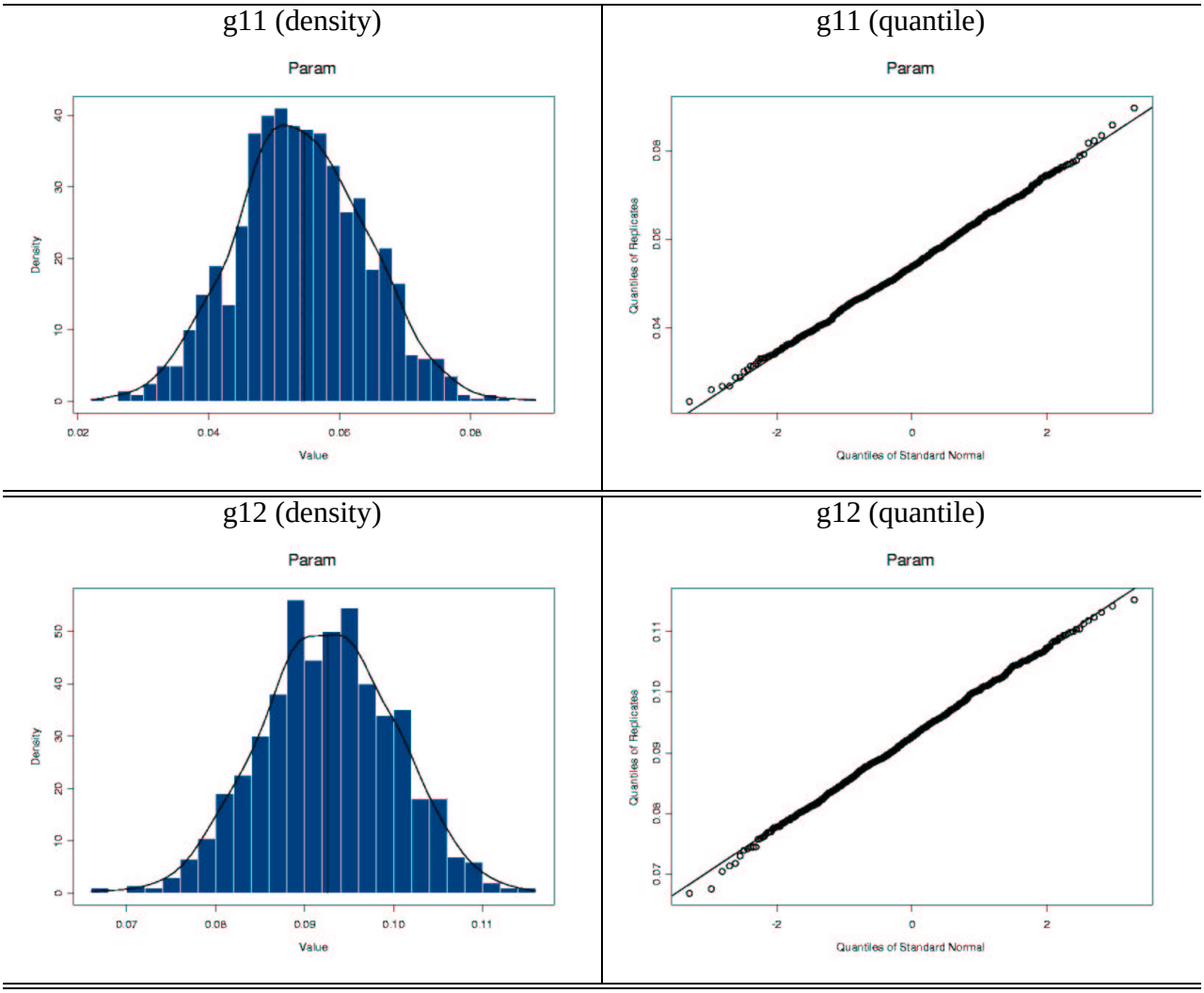


Figure C-47: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

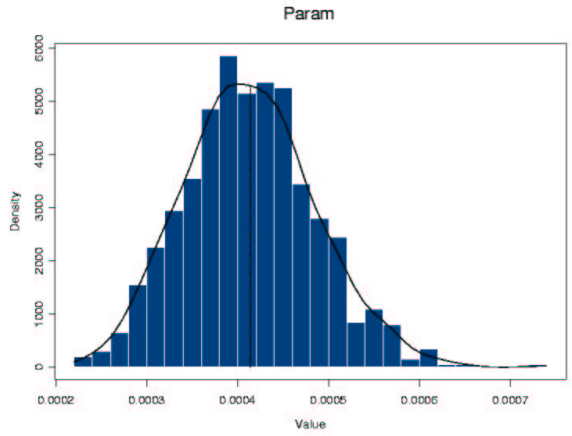
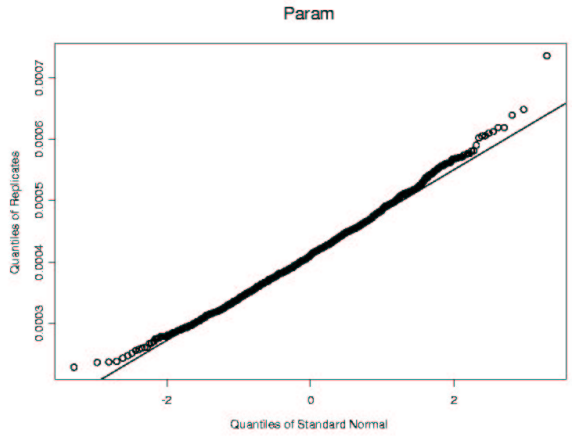
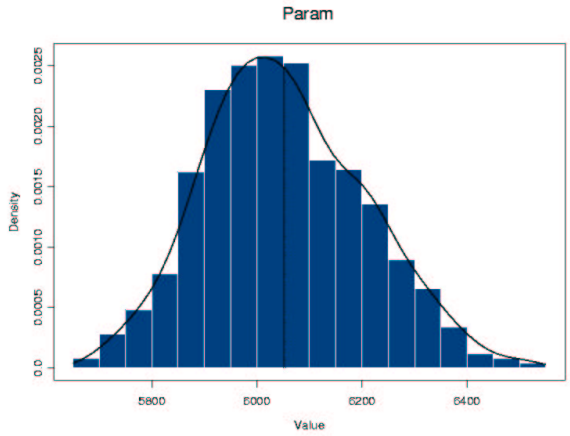
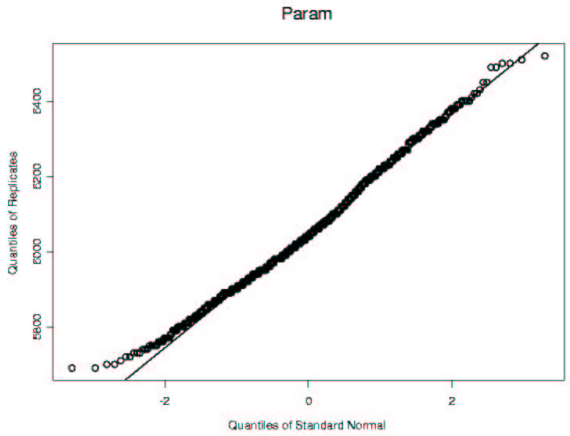
g13 (density)	g13 (quantile)
	

Figure C-48: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the PROGRESS-RATIO performance measure. Also shown is the normal quantile graph.

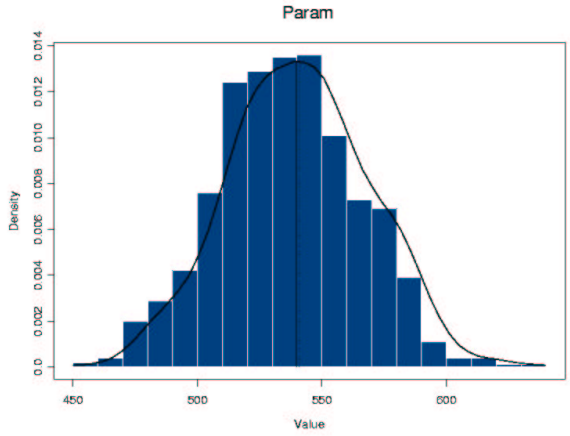
g01 (density)



g01 (quantile)



g04 (density)



g04 (quantile)

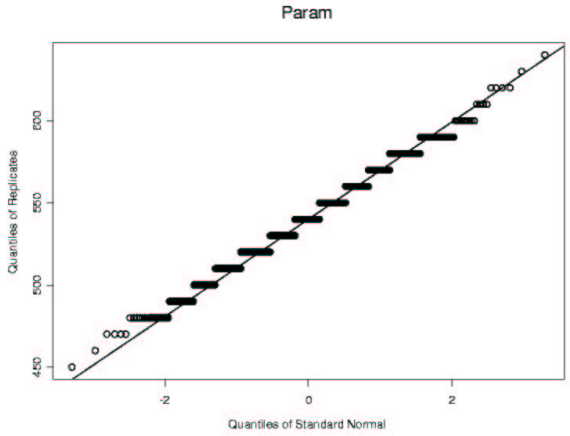


Figure C-49: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

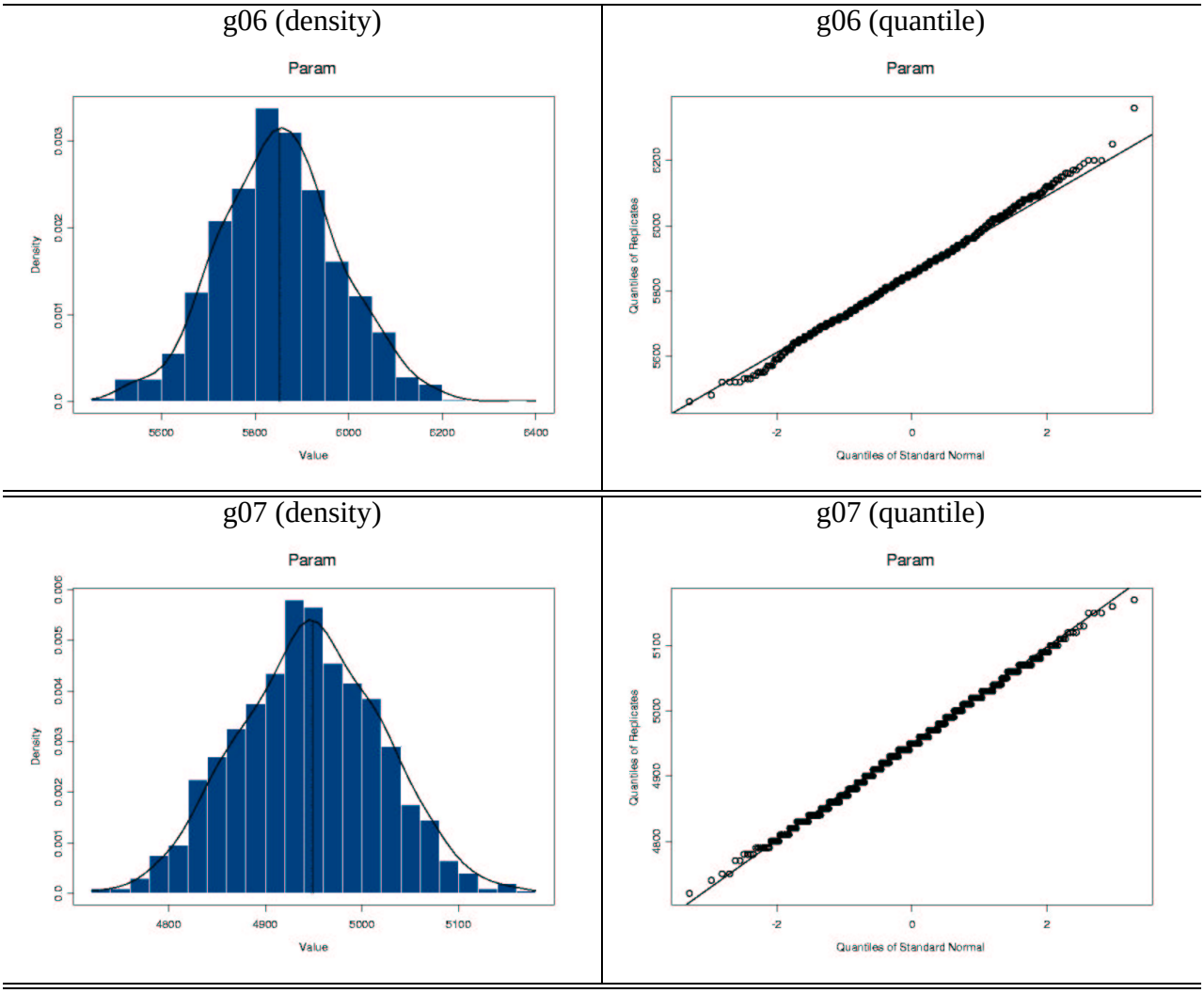


Figure C-50: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

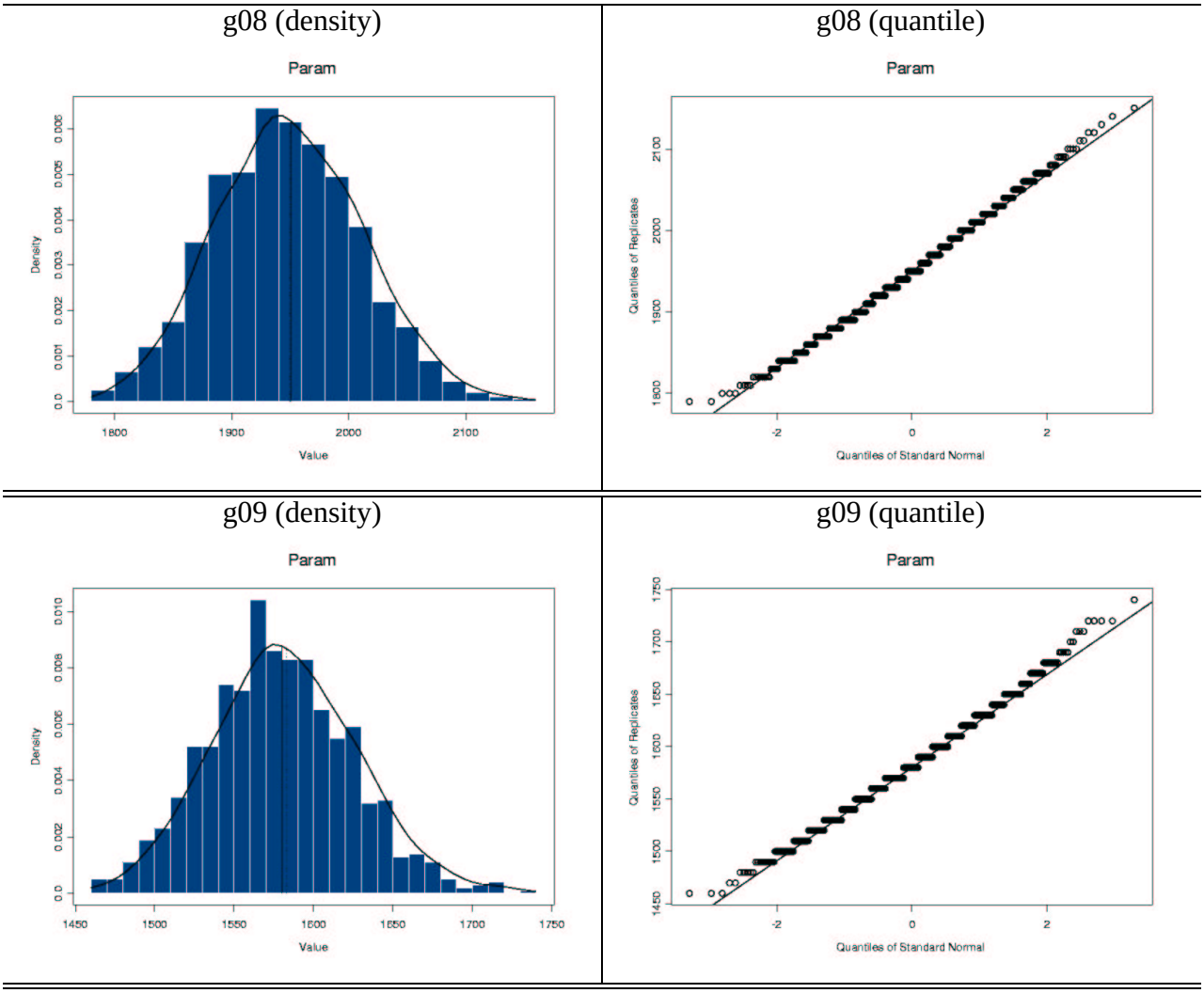
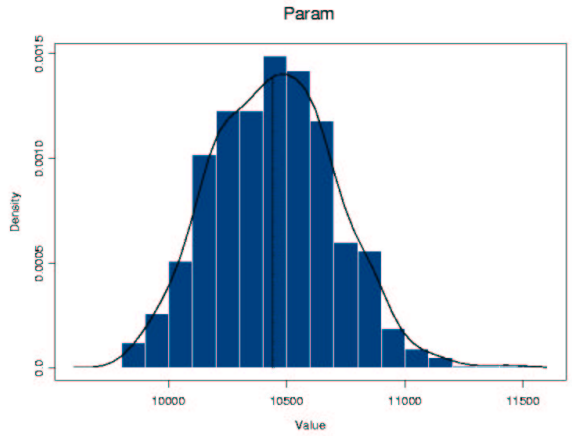
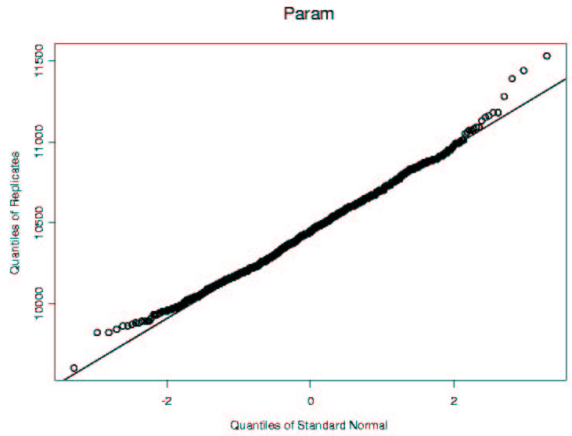


Figure C-51: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

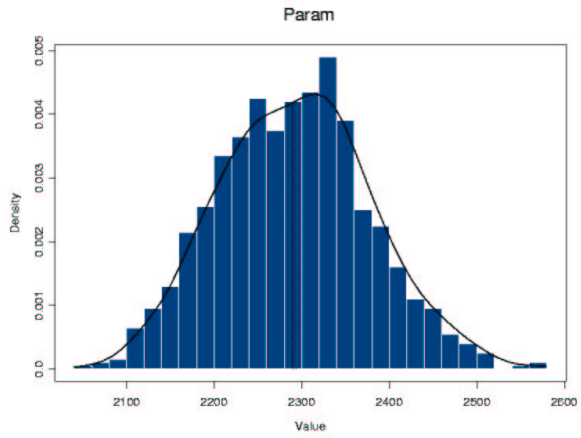
g10 (density)



g10 (quantile)



g12 (density)



g12 (quantile)

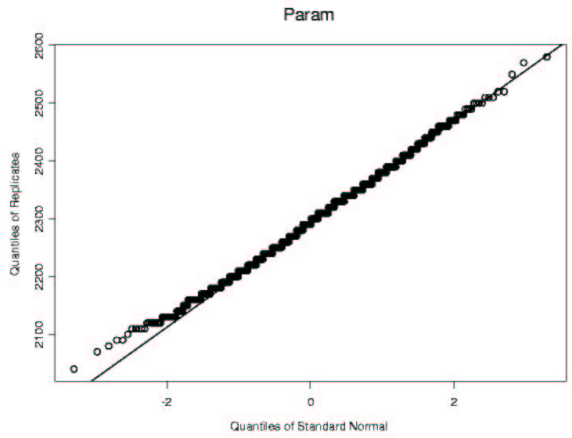
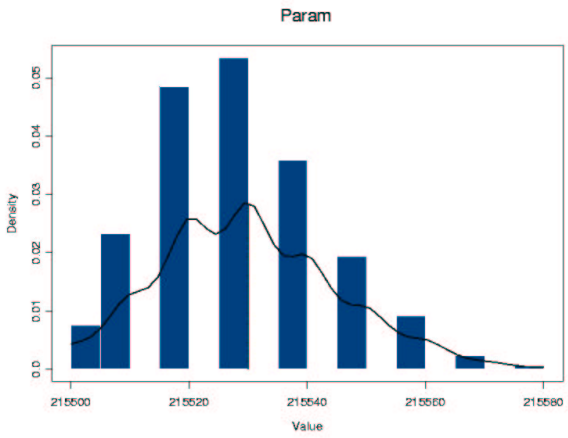


Figure C-52: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

g13 (density)



g13 (quantile)

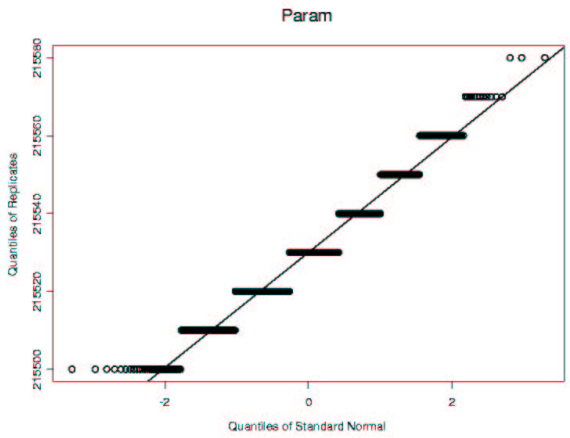
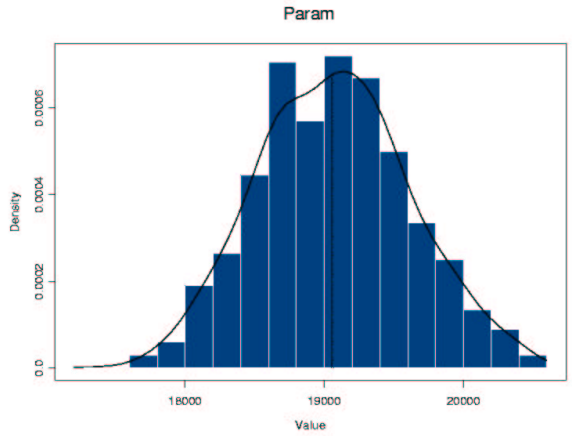
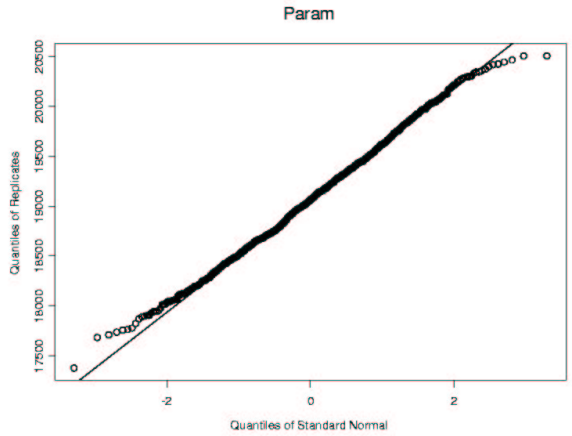


Figure C-53: Histogram and density line of the bootstrapping distribution of results obtained by the SMES for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

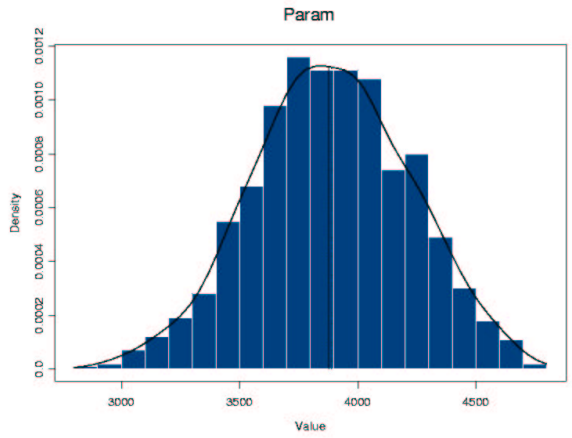
g01 (density)



g01 (quantile)



g04 (density)



g04 (quantile)

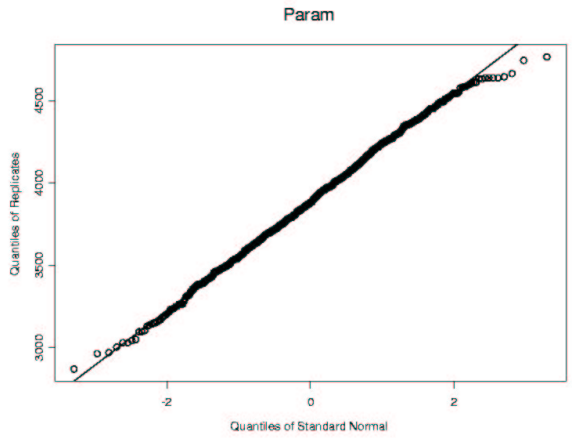
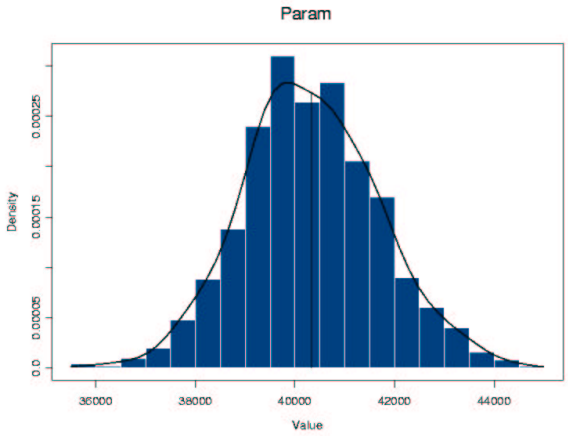
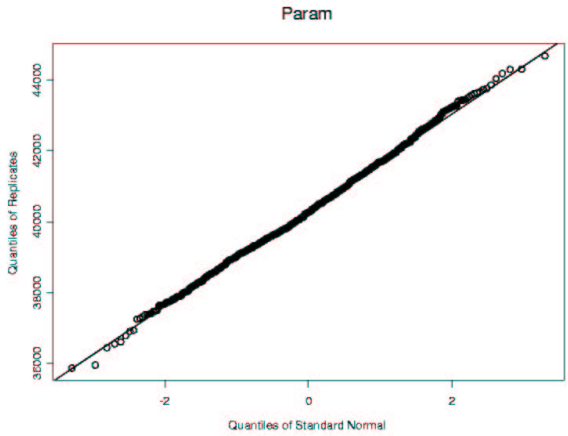


Figure C-54: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

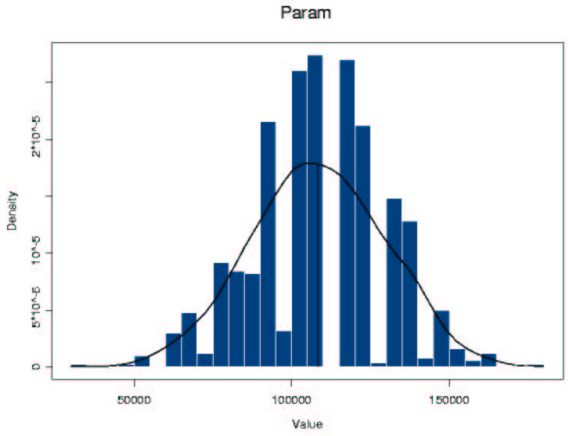
g05 (density)



g05 (quantile)



g06 (density)



g06 (quantile)

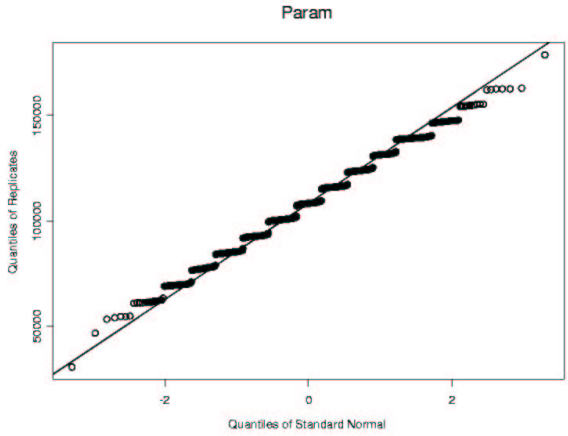


Figure C-55: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

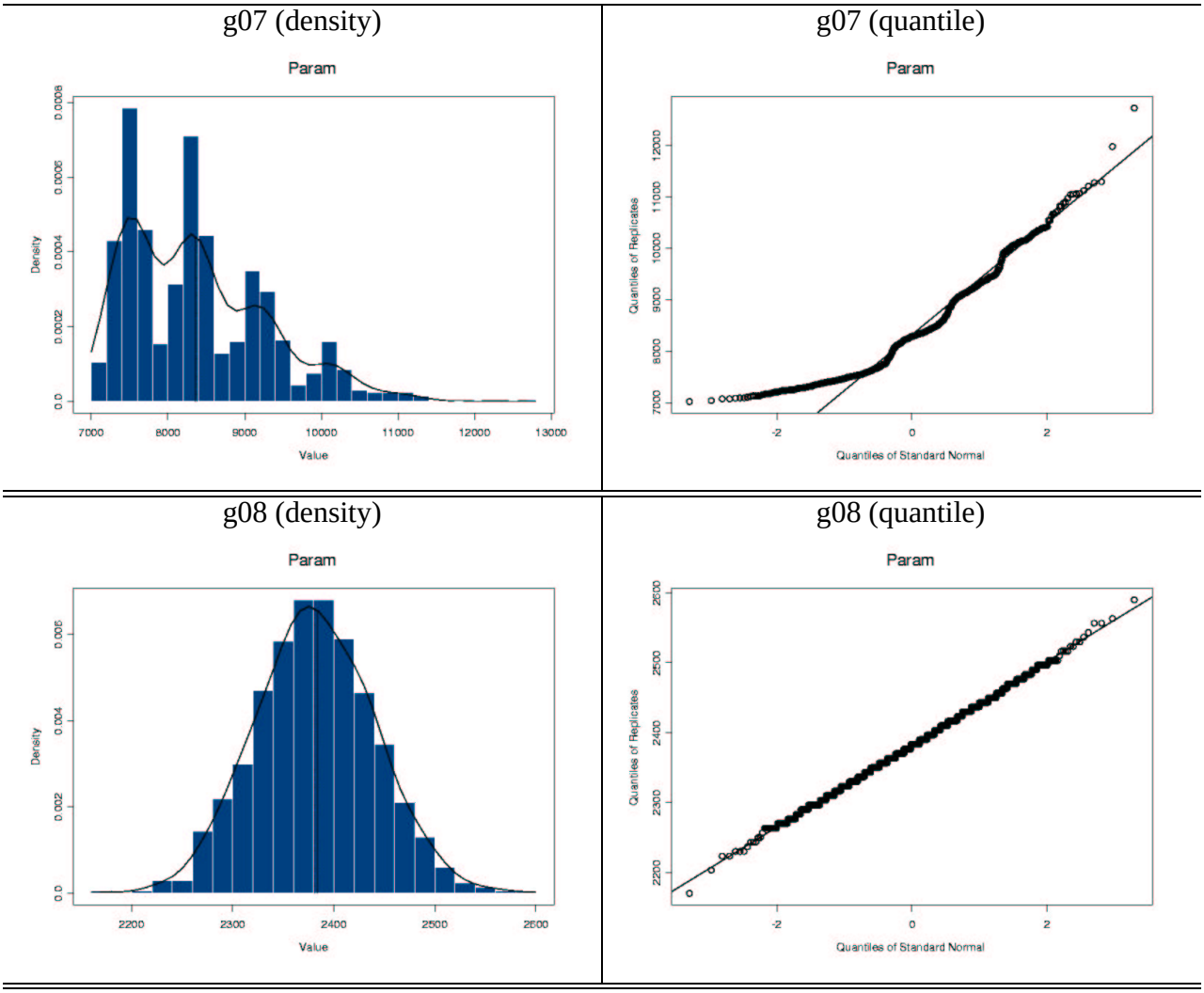


Figure C-56: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

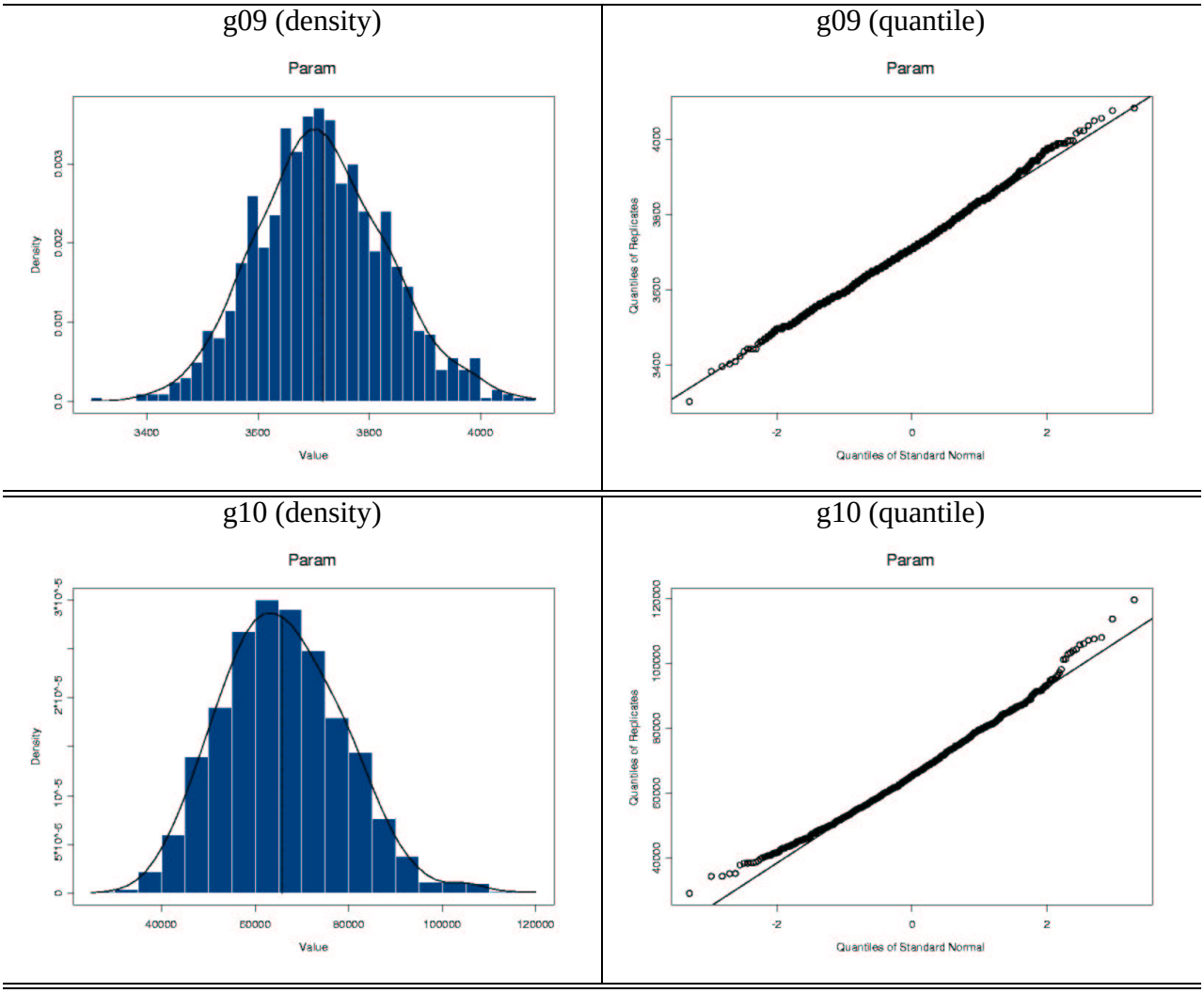
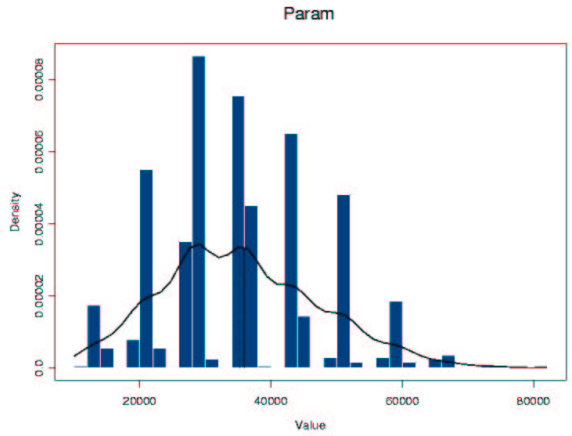
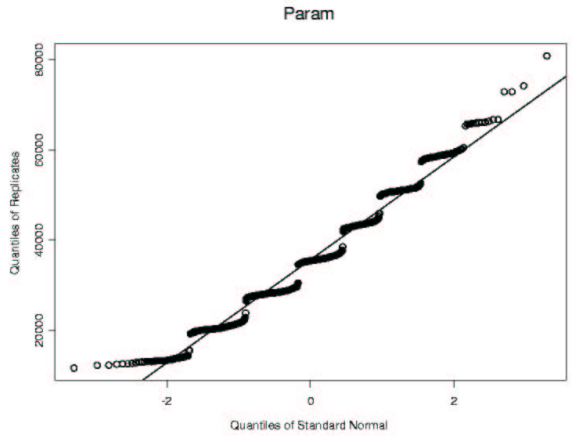


Figure C-57: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

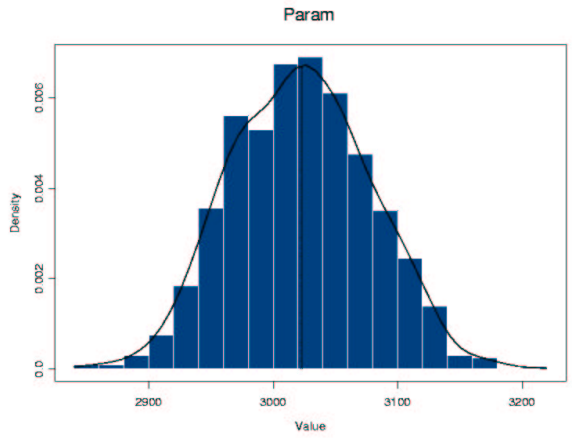
g11 (density)



g11 (quantile)



g12 (density)



g12 (quantile)

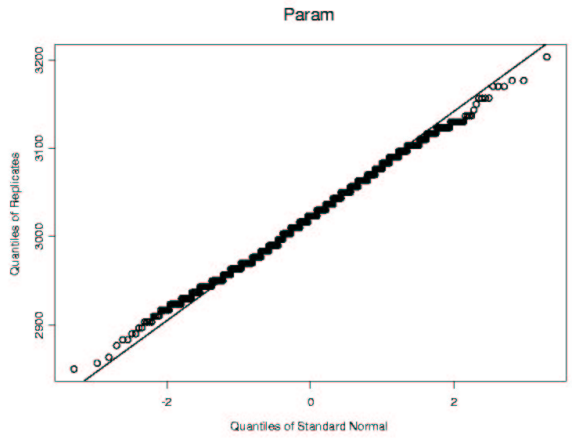


Figure C-58: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

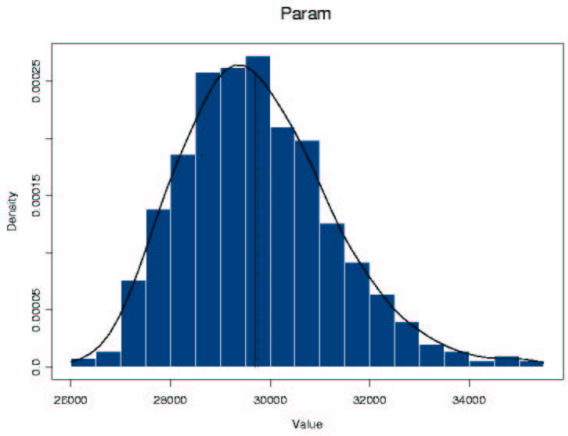
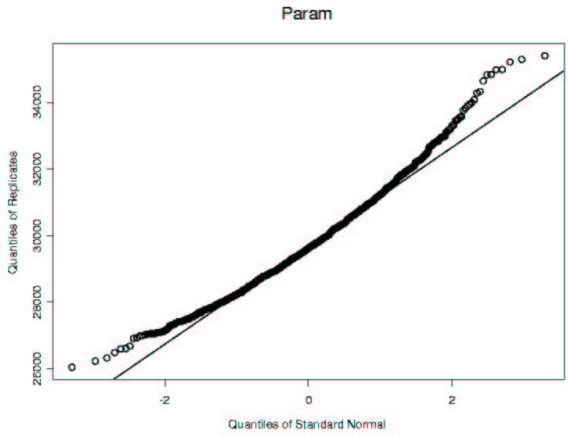
g13 (density)	g13 (quantile)
	

Figure C-59: Histogram and density line of the bootstrapping distribution of results obtained by the Stochastic Ranking for the ALL-FEASIBLE performance measure. Also shown is the normal quantile graph.

Bibliography

[1] David H. Ackley, editor. *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, Boston, Massachusetts, 1995.

[2] Hojjat Adeli and Nai-Tsang Cheng. Augmented lagrangian genetic algorithm for structural optimization. *Journal of Aerospace Engineering*, 7(1):104–118, January 1994.

[3] Shamim Akhtar, Kang Tai, and Tapabrata Ray. A socio-behavioural simulation model for engineering design optimization. *Engineering Optimization*, 34(4):341–354, 2002.

[4] Dirk V. Arnold. *Noisy Optimization with Evolution Strategies*. Kluwer Academic Publishers, New York, June 2002. ISBN 1-4020-7105-1.

[5] Jasbir S. Arora. *Introduction to Optimum Design*. McGraw-Hill, New York, 1989.

[6] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[7] Thomas Bäck. Introduction to the special issue: Self-adaptation. *Evolutionary Computation*, 9(2):iii–iv, 2001.

[8] Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel. A survey of evolution strategies. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algori thms*, pages 2–9, San Mateo, California, 1991. Morgan Kaufmann Publishers.

[9] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In John Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

[10] E. Baldwin. *Genética elemental*. Limusa, 1983.

[11] M.S. Bazaraa and C.M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley, New York, 1979.

[12] James C. Bean. Genetics and random keys for sequencing and optimization. Technical Report TR 92-43, Department of Industrial and Operations Engineering, The University of Michigan, 1992.

[13] James C. Bean. Genetics and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160, 1994.

[14] James C. Bean and Atidel Ben Hadj-Alouane. A dual genetic algorithm for bounded integer programs. Technical Report TR 92-53, Department of Industrial and Operations Engineering, The University of Michigan, 1992. To appear in R.A.I.R.O.-R.O. (invited submission to special issue on GAs and OR).

[15] Ashok Dhondu Belegundu. *A Study of Mathematical Programming Methods for Structural Optimization*. Department of civil and environmental engineering, University of Iowa, Iowa, 1982.

[16] Sheela V. Belur. CORE: Constrained optimization by random evolution. In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1997 Conference*, pages 280–286, Stanford University, California, July 1997. Stanford Bookstore.

[17] Hans-Georg Beyer. *The theory of Evolution Strategies*. Springer, Berlin, 2001.

[18] George Bilchev and Ian C. Parmee. The ant colony metaphor for searching continuous design spaces. In Terence C. Fogarty, editor, *Evolutionary Computing. AISB Workshop. Selected Papers*, pages 25–39, Sheffield, U.K., April 1995. Springer-Verlag. Lecture Notes in Computer Science No. 993.

[19] George Bilchev and Ian C. Parmee. Constrained and multi-modal optimisation with an ant colony search model. In Ian C. Parmee and M. J. Denham, editors, *Proceedings of 2nd International Conference on Adaptive Computing in Engineering Design and Control*. University of Plymouth, Plymouth, UK, March 1996.

[20] L. B. Booker. Intelligent behavior as an adaptation to the task environment. Technical Report 243, University of Michigan at Ann Arbor, Ann Arbor, Michigan, 1982.

[21] M.J. Box. A new method of constrained optimization and a comparison with other methods. *Computer Journal*, 8:42–52, 1965.

[22] A. Brindle. *Genetic Algorithms for Function Optimization*. PhD thesis, Department of Computer Science of the University of Alberta, Alberta, Canada, 1981.

[23] Bill P. Buckles and Fred E. Petry, editors. *Genetic Algorithms*. IEEE Computer Society Press, 1992.

[24] Eduardo Camponogara and Sarosh N. Talukdar. A genetic algorithm for constrained and multiobjective optimization. In Jarmo T. Alander, editor, *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, pages 49–62, Vaasa, Finland, August 1997. University of Vaasa.

[25] Susan Carlson-Skalak, Ron Shonkwiler, Sani Babar, and M. Aral. Annealing a genetic algorithm over constraints. Available at http://vlead.mech.virginia.edu/publications/shenkpaper/shenkpaper.html.

[26] V. Chankong and Y.Y Haimes. Multiobjective decision making: Theory and methodology. In Andrew P. Sage, editor, *Systems Science and Engineering*, volume 8, pages 62–109. North-Holland, 1983.

[27] Chan-Jin Chung and Robert G. Reynolds. A testbed for solving optimization problems using cultural algorithms. In Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck, editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 225–236, Cambridge, Massachusetts, March 1996. MIT Press.

[28] Carlos A. Coello Coello. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.

[29] Carlos A. Coello Coello, Alan D. Christiansen, and Arturo Hernández Aguirre. Using a new GA-based multiobjetive optimization for robot arms. *Robotica*, 16(4):401–414, 1998.

[30] Carlos A. Coello Coello and Nareli Cruz Cortés. A parallel implementation of an artificial immune system to handle constraints in genetic algorithms: Preliminary results. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 819–824, Piscataway, New Jersey, May 2002. IEEE Service Center.

[31] Carlos A. Coello Coello, Filiberto Santos Hernández, and Francisco Alonso Farrera. Optimal design of reinforced concrete beams using genetic algorithms. *Expert Systems with Applications : An International Journal*, 12(1):101–108, January 1997.

[32] Carlos A. Coello Coello and Efrén Mezura-Montes. Handling constraints in genetic algorithms using dominance-based tournaments. In I.C. Parmee, editor, *Proceedings of the Fifth International Conference on Adaptive Computing in Design and Manufacture (ACDM'2002)*, volume 5, pages 273–284, University of Exeter, Devon, UK, April 2002. Springer-Verlag.

[33] Carlos A. Coello Coello. Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems*, 17:319–346, 2000.

[34] Carlos A. Coello Coello. Treating constraints as objectives for single-objective evolutionary optimization. *Engineering Optimization*, 32(3):275–308, 2000.

[35] Carlos A. Coello Coello. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127, January 2000.

[36] Carlos A. Coello Coello and Nareli Cruz Cortés. Use of emulations of the immune system to handle constraints in evolutionary algorithms. In Cihan H. Dagli, Anna L. Buczak, Joydeep Ghosh, Mark J. Embrechts, Okan Erson, and Stephen Kercel, editors, *Intelligent Engineering Systems through Artificial Neural Networks (ANNIE'2001)*, volume 11, pages 141–146, St. Louis Missouri, USA, November 2001. ASME Press.

[37] Carlos A. Coello Coello and Arturo Hernández Aguirre. Design of combinational logic circuits through an evolutionary multiobjective optimization approach. *Artificial Intelligence for Engineering, Design, Analysis and Manufacture*, 16(1):39–53, 2002.

[38] Carlos A. Coello Coello and Ricardo Landa Becerra. Adding knowledge and efficient data structures to evolutionary programming: A cultural algorithm for constrained optimization. In W.B. Langdon, E.Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke, and N.Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 201–209, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[39] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, June 2002. ISBN 0-3064-6762-3.

[40] William A. Crossley and Edwin A. Williams. A study of adaptive penalty functions for constrained genetic algorithm based optimization. In *AIAA 35th Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, January 1997. AIAA Paper 97-0083.

[41] Charles Darwin. *The Origin of Species by Means of Natural Selection or the preservation of Favored Races in the Struggle for life*. Random House, New York, 1993. (Publicado originalmente en 1929).

[42] Dipankar Dasgupta and Douglas R. McGregor. A more biologically motivated genetic algorithm: The model and some results. *Cybernetics and Systems: An International Journal*, 25(3):447–469, May-June 1994.

[43] Dipankar Dasgupta and Zbigniew Michalewicz, editors. *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, Berlin, 1997.

[44] Yuval Davidor. A genetic algorithm applied to robot trajectory generation. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, chapter 12, pages 144–165. Van Nostrand Reinhold, New York, New York, 1991.

[45] Ernest Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32:281–331, 1987.

[46] Lawrence Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, New York, 1991.

[47] Leandro Nunes de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer Verlag, 2002.

[48] Kalyanmoy Deb. Genetic algorithms in multimodal function optimization. Master's thesis, University of Alabama, Alabama, USA, 1989.

[49] Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.

[50] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions of Systems, Man and Cybernetics-Part B*, 26(1):29–41, 1996.

[51] A. E. Eiben and J. K. van der Hauw. Adaptive penalties for evolutionary graph coloring. In J.-K Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution'97*, pages 95–106, Berlin, 1998. Springer-Verlag. Lecture Notes in Computer Science Vol. 1363.

[52] A.E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Verlag, 2003.

[53] T.G.W. Epperly. Global optimization test problems with solutions. Available at http://citeseer.nj.nec.com/147308.html.

[54] T.G.W. Epperly and R.E. Swaney. Branch and bound for global NLP: Iterative LP algorithm & results. In Ignacio E. Grossman, editor, *Global Optimization in Engineering Design*, Nonconvex Optimization and its Applications, chapter 2, pages 37–73. Kluwer Academic Publishers, Dordrecht, the Netherlands, 1996.

[55] H. Everett. Generalized lagrange mutiplier method for solving problems of optimum allocation of resources. *Oper. Res.*, 11:399–471, 1963.

[56] Raziyeh Farmani and Jonathan A. Wright. Self-adaptive fitness formulation for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 7(5):445–455, October 2003.

[57] Christodoulos A. Floudas, Panos M. Pardalos, Claire S. Adjiman, William R. Esposito, Zeynep H. Gümüs, Stephen T. Harding, John L. Klepeis, Clifford A. Meyer, and Carl A.

Schweiger. *Handbook of Test Problems in Local and Global Optimization*. Nonconvex Optimization and its Applications. Kluwer Academic Publishers, Dordrecht, the Netherlands, 1999.

[58] David B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, January 1994.

[59] David B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. The Institute of Electrical and Electronic Engineers, New York, 1995.

[60] D.B. Fogel. *Evolving Artificial Intelligence*. PhD thesis, University of California in San Diego, San Diego, CA., 1992.

[61] Lawrence J. Fogel. *Intelligence Through Simulated Evolution. Forty years of Evolutionary Programming*. John Wiley & Sons, New York, 1999.

[62] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[63] Mitsuo Gen and Runwei Cheng. Interval programming using genetic algorithms. In *Proceedings of the Sixth International Symposium on Robotics and Manufacturing*, Montpellier, France, 1996.

[64] Fred Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977.

[65] David Goldberg. *The Design of Innovation*. Kluwer Academic Publishers, New York, June 2002. ISBN 1-4020-7098-5.

[66] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., Reading, Massachusetts, 1989.

[67] David E. Goldberg and Kalyanmoy Deb. A comparison of selection schemes used in genetic algorithms. In G.J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, San Mateo, California, 1991.

[68] David E. Goldberg, Kalyanmoy Deb, and Bradley Korb. Don't worry, be messy. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 24–30, San Mateo, California, July 1991. University of California, San Diego, Morgan Kaufmann Publishers.

[69] Martina Gorges-Schleuter, Ingo Sieber, and Wilfried Jakob. Local interaction evolution strategies for design optimization. In *1999 Congress on Evolutionary Computation*, pages 2167–2174, Piscataway, NJ, 1999. IEEE Service Center.

[70] Garrison W. Greenwood and Yi-Ping Liu. Finding low energy conformations of atomic clusters using evolution strategies. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII*, pages 493–502, Berlin, 1998. Springer. Lecture Notes in Computer Science 1447.

[71] Atidel Ben Hadj-Alouane and James C. Bean. A genetic algorithm for the multiple-choice integer program. *Operations Research*, 45:92–101, 1997.

[72] P. Hajela and J. Lee. Constrained genetic search via schema adaptation. an immune network solution. In Niels Olhoff and George I. N. Rozvany, editors, *Proceedings of the First World Congress of Stuctural and Multidisciplinary Optimization*, pages 915–920, Goslar, Germany, 1995. Pergamon.

[73] P. Hajela and J. Lee. Constrained genetic search via schema adaptation. an immune network solution. *Structural Optimization*, 12:11–15, 1996.

[74] P. Hajela and J. Yoo. Constraint handling in genetic search using expression strategies. *AIAA Journal*, 34(12):2414–2420, 1996.

[75] Hatem Hamda and Marc Schoenauer. Adaptive techniques for evolutionary topological optimum design. In I.C. Parmee, editor, *Proceedings of the Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM'2000)*, pages 123–136, University of Plymouth, Devon, UK, April 2000. Springer-Verlag.

[76] S. Ben Hamida and Marc Schoenauer. An adaptive algorithm for constrained optimization problems. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Proceedings of 6th Parallel Problem Solving From Nature (PPSN VI)*, pages 529–538, Heidelberg, Germany, September 2000. Paris, France, Springer-Verlag. Lecture Notes in Computer Science Vol. 1917.

[77] Sana Ben Hamida and Marc Schoenauer. ASCHEA: New results using adaptive segregational constraint handling. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 884–889, Piscataway, New Jersey, May 2002. IEEE Service Center.

[78] Steven A. Harp and Tariq Samad. Genetic synthesis of neural network architecture. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, chapter 15, pages 202–221. Van Nostrand Reinhold, New York, New York, 1991.

[79] Arturo Hernández-Aguirre, Salvador Botello-Rionda, Carlos A. Coello Coello, Giovanni Lizárraga-Lizárraga, and Efrén Mezura-Montes. Handling constraints using multiobjective optimization concepts. *International Journal for Numerical Methods in Engineering*, 59(15):1989–2017, April 2004.

[80] Arturo Hernández Aguirre, Salvador Botello Rionda, Giovanni Lizárraga Lizárraga, and Carlos A. Coello Coello. IS-PAES: A constraint-handling technique based on multiobjective optimization concepts. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 73–87, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[81] David M. Himmelblau. *Applied Nonlinear Programming*. Mc-Graw-Hill, USA, 1972.

[82] R. Hinterding, Z. Michalewicz, and A.E. Eiben. Adaptation in evolutionary computation: A survey. In *Proceedings of the Fourth IEEE Conference on Evolutionary Computation*, pages 65–69, Piscataway, New Jersey, 1997. IEEE Press.

[83] Robert Hinterding and Zbigniew Michalewicz. Your brains and my beauty: Parent matching for constrained optimisation. In *Proceedings of the 5th International Conference on Evolutionary Computation*, pages 810–815, Anchorage, Alaska, May 1998.

[84] Frank Hoffmeister and Joachim Sprave. Problem-independent handling of constraints by use of metric penalty functions. In Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck, editors, *Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP'96)*, pages 289–294, San Diego, California, February 1996. The MIT Press.

[85] John H. Holland. *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, Ann Arbor, Michigan, 1975.

[86] John H. Holland. *Adaptation in Natural an Artificial Systems*. MIT Press, Cambridge, Massachusetts, second edition, 1992.

[87] A. Homaifar, S. H. Y. Lai, and X. Qi. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–254, 1994.

[88] J. Horn and N. Nafpliotis. Multiobjective optimization using the niched pareto genetic algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[89] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[90] E. Hyvoenen. Constraint reasoning based on interval arithmetic—The tolerance propagation approach. *Artificial Intelligence*, 58:71–112, 1992.

[91] John Jenkins, editor. *Genetics*. Houghton Mifflin Company, Boston, Massachusetts, 1984.

[92] F. Jiménez, A.F. Gómez-Skarmeta, and G. Sánchez. How evolutionary multi-objective optimization can be used for goals and priorities based optimization. In *Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados (AEB'02)*, pages 460–465. Mérida España, 2002.

[93] Fernando Jiménez and José L. Verdegay. Evolutionary techniques for constrained optimization problems. In Hans-Jürgen Zimmermann, editor, *7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)*, Aachen, Germany, 1999. Verlag Mainz. ISBN 3-89653-808-X.

[94] Xidong Jin and Robert G. Reynolds. Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: A cultural algorithm approach. In *1999 Congress on Evolutionary Computation*, pages 1672–1678, Washington, D.C., July 1999. IEEE Service Center.

[95] Siddall J.N. *Analytical Decision-Making in Engineering Design*. Prentice-Hall, Englewood, Cliffs, NJ, 1972.

[96] J. Joines and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In David Fogel, editor, *Proceedings of the first IEEE Conference on Evolutionary Computation*, pages 579–584, Orlando, Florida, 1994. IEEE Press.

[97] A. K. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptative Systems*. PhD thesis, University of Michigan, 1975.

[98] B. K. Kannan and S. N. Kramer. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design. Transactions of the ASME*, 116:318–320, 1994.

[99] S. Kazarlis and V. Petridis. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th Parallel Problem Solving from Nature (PPSN V)*, pages 211–220, Heidelberg, Germany, September 1998. Amsterdan, The Netherlands, Springer-Verlag. Lecture Notes in Computer Science Vol. 1498.

[100] S. A. Kazarlis, S. E. Papadakis, J. B. Theocharis, and V. Petridis. Microgenetic algorithms as generalized hill climbing operators for GA optimization. *IEEE Transactions on Evolutionary Computation*, 5(3):204–217, June 2001.

[101] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, UK, 2001.

[102] Dae Gyu Kim and Phil Husbands. Riemann mapping constraint handling method for genetic algorithms. Technical Report CSRP 469, COGS, University of Sussex, UK, 1997.

[103] Dae Gyu Kim and Phil Husbands. Mapping based constraint handling for evolutionary search; thurston's circle packing and grid generation. In Ian Parmee, editor, *The Integration of Evolutionary and Adaptive Computing Technologies with Product/System Design and Realisation*, pages 161–173. Springer-Verlag, Plymouth, United Kingdom, April 1998.

[104] J.-H. Kim and H. Myung. Evolutionary programming techniques for constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 1:129–140, July 1997.

[105] S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[106] Joshua D. Knowles and David W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[107] Ryszard Kowalczyk. Constraint consistent genetic algorithms. In *Proceedings of the 1997 IEEE Conference on Evolutionary Computation*, pages 343–348, Indianapolis, USA, April 1997. IEEE.

[108] John R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, Massachusetts, 1992.

[109] Slawomir Koziel and Zbigniew Michalewicz. A decoder-based evolutionary algorithm for constrained parameter optimization problems. In T. Bäck, A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th Parallel Problem Solving from Nature (PPSN V)*, pages 231–240, Heidelberg, Germany, September 1998. Amsterdan, The Netherlands, Springer-Verlag. Lecture Notes in Computer Science Vol. 1498.

[110] Slawomir Koziel and Zbigniew Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[111] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, pages 32–44, Spring 1992.

[112] Angel Kuri-Morales and Carlos Villegas Quezada. A universal eclectic genetic algorithm for constrained optimization. In *Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98*, pages 518–522, Aachen, Germany, September 1998. Verlag Mainz.

[113] Jouni Lampinen. A constraint handling approach for the differential evolution algorithm. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 2, pages 1468–1473, Piscataway, New Jersey, May 2002. IEEE Service Center.

[114] T. Van Le. A fuzzy evolutionary approach to constrained optimization problems. In *Proceedings of the Second IEEE Conference on Evolutionary Computation*, pages 274–278, Perth, November 1995. IEEE.

[115] G. E. Liepins and Michael D. Vose. Representational issues in genetic optimization. *Journal of Experimental and Theoretical Computer Science*, 2(2):4–30, 1990.

[116] Gunar E. Liepins and W. D. Potter. A genetic algorithm approach to multiple-fault diagnosis. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, chapter 17, pages 237–250. Van Nostrand Reinhold, New York, New York, 1991.

[117] C. B. Lucasius, M. J. J. Blommers, L. M. C. Buydens, and G. Kateman. A genetic algorithm for conformational analysis of DNA. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, chapter 18, pages 251–281. Van Nostrand Reinhold, New York, New York, 1991.

[118] O.L. Mangasarian. *Nonlinear Programming*. McGraw-Hill, New York, 1969.

[119] G.P. McCormick. Second order conditions for constrained optima. *SIAM J. Appl. Math*, 15:641–652, 1967.

[120] Efrén Mezura-Montes. Uso de la Técnica Multiobjetivo NPGA para el Manejo de Restricciones en Algoritmos Genéticos. Master's thesis, Universidad Veracruzana, Xalapa, México, 2001. (In Spanish).

[121] Efrén Mezura-Montes and Carlos A. Coello Coello. On the usefulness of the evolution strategies self-adaptation mechanism to handle constraints in global optimization. Technical Report EVOCINV-01-2003, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México D.F., México, 2003. Available in the Constraint Handling Techniques in Evolutionary Algorithms Repository at http://www.cs.cinvestav.mx/˜constraint/.

[122] Efrén Mezura-Montes and Carlos A. Coello Coello. A simple evolution strategy to solve constrained optimization problems. In Erick Cantú-Paz, James A. Foster, Kalyanmoy Deb, Lawrence David Davis, Rajkumar Roy, Una-May O' Reilly, Hans-Georg Beyer, Russell Standish, Graham Kendall, Stewart Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, Mitch A. Potter, Alan C. Schultz, Kathryn A. Dowsland, Natasha Jonoska, and Julian Miller, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003)*, pages 640–641, Heidelberg, Germany, July 2003. Chicago, Illinois, Springer Verlag. Lecture Notes in Computer Science Vol. 2723.

[123] Efrén Mezura-Montes and Carlos A. Coello Coello. A simple multimembered evolution strategy to solve constrained optimization problems. Technical Report EVOCINV-04-2003, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México D.F., México, 2003. Available in the Constraint Handling Techniques in Evolutionary Algorithms Repository at http://www.cs.cinvestav.mx/˜constraint/.

[124] Efrén Mezura-Montes and Carloa A. Coello Coello. Adding a diversity mechanism to a simple evolution strategy to solve constrained optimization problems. In *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, volume 1, pages 6–13, Piscataway, New Jersey, December 2003. Canberra, Australia, IEEE Service Center.

[125] Efrén Mezura-Montes, Carlos A. Coello Coello, and Ricardo Landa-Becerra. Engineering optimization using a simple evolutionary algorithm. In *Proceedings of the Fiftheenth International Conference on Tools with Artificial Intelligence (ICTAI'2003)*, pages 149–156, Los Alamitos, CA, November 2003. Sacramento, California, IEEE Computer Society.

[126] Zbigniew Michalewicz. Genetic algorithms, numerical optimization, and constraints. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, pages 151–158, San Mateo, California, July 1995. University of Pittsburgh, Morgan Kaufmann Publishers.

[127] Zbigniew Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 135–155. The MIT Press, Cambridge, Massachusetts, 1995.

[128] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third edition, 1996.

[129] Zbigniew Michalewicz and Naguib F. Attia. Evolutionary optimization of constrained problems. In *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 98–108. World Scientific, 1994.

[130] Zbigniew Michalewicz, Kalyanmoy Deb, Martin Schmidt, and Thomas Stidsen. Test-case generator for nonlinear continuous parameter optimization techniques. *IEEE Transactions on Evolutionary Computation*, 4(3):197–215, September 2000.

[131] Zbigniew Michalewicz and Cezary Z. Janikow. Handling constraints in genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, pages 151–157, San Mateo, California, 1991. University of California, San Diego, Morgan Kaufmann Publishers.

[132] Zbigniew Michalewicz and G. Nazhiyath. Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints. In David B. Fogel, editor, *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, pages 647–651, Piscataway, New Jersey, 1995. IEEE Press.

[133] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[134] Hyun Myung and Jong-Hwan Kim. Hybrid interior-lagrangian penalty based evolutionary optimization. In V.W. Porto, N. Saravanan, D. Waagen, and A.E. Eiben, editors, *Proceedings of the 7th International Conference on Evolutionary Programming (EP98)*, pages 85–94, Heidelberg, Germany, March 1998. San Diego, California, USA, Springer-Verlag. Lecture Notes in Computer Science Vol. 1447.

[135] Ryohei Nakano and Takeshi Yamada. Conventional genetic algorithm for job shop problems. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, pages 474–479, San Mateo, California, 1991. University of California, San Diego, Morgan Kaufmann Publishers.

[136] A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[137] NikolausHansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[138] Bryan A. Norman and Alice E. Smith. Random keys genetic algorithm with adaptive penalty function for optimization of constrained facility layout problems. In Thomas Bäck, Zbigniew Michalewicz, and Xin Yao, editors, *Proceedings of the 1997 International Conference on Evolutionary Computation*, pages 407–411, Indianapolis, Indiana, 1997. IEEE.

[139] Andreas Ostermeier, Andreas Gawelczyk, and Nikolaus Hansen. A derandomized approach to self-adaptation of evolution strategies. *Evolutionary Computation*, 2(4):369–380, 1995.

[140] Ahmet Irfan Oyman, Kalyanmoy Deb, and Hans-Georg Beyer. An alternative constraint handling method for evolution strategies. In *Proceedings of the Congress on Evolutionary Computation 1999 (CEC'99)*, volume 1, pages 612–619, Piscataway, New Jersey, July 1999. IEEE Service Center.

[141] J. Paredis. Co-evolutionary constraint satisfaction. In *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, pages 46–55, New York, 1994. Springer Verlag.

[142] Vilfredo Pareto. *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.

[143] I. C. Parmee and G. Purchase. The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Adaptive Computing in Engineering Design and Control-'94*, pages 97–102, Plymouth, UK, 1994. University of Plymouth.

[144] Rebecca Parsons, Stephanie Forrest, and Christian Burks. Genetic algorithms for DNA sequence assembly. In *Proceedings of the 1st International Conference on Intelligent Systems in Molecular Biology*. AAAI Press, July 1993.

[145] Rebecca J. Parsons, Stephanie Forrest, and Christian Burks. Genetic algorithms, operators and DNA fragment assembly. *Machine Learning*, 21(1–2):11–33, October/November 1995.

[146] David Powell and Michael M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pages 424–431, San Mateo, California, July 1993. University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers.

[147] M.J.D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*. Academic Press, London, England, 1969.

[148] Kenneth V. Price. An introduction to differential evolution. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 79–108. Mc Graw-Hill, UK, 1999.

[149] S. Rao. Game theory approach for multiobjective structural optimization. *Computers and Structures*, 25(1):119–127, 1986.

[150] Singiresu S. Rao. *Engineering Optimization*. John Wiley and Sons, third edition, 1996.

[151] Khaled Rasheed. An adaptive penalty approach for constrained genetic-algorithm optimization. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick L. Riolo, editors, *Proceedings of the Third Annual Genetic Programming Conference*, pages 584–590, San Francisco, California, 1998. Morgan Kaufmann Publishers.

[152] Tapabrata Ray. Constraint robust optimal design using a multiobjective evolutionary algorithm. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 419–424, Piscataway, New Jersey, May 2002. IEEE Service Center.

[153] Tapabrata Ray, Tai Kang, and Seow Kian Chye. An evolutionary algorithm for constrained optimization. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, July 2000. Morgan Kaufmann Publishers.

[154] Ingo Rechenberg. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment*, August 1965. Library Translation No. 1122, Farnborough, Hants, UK.

[155] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.

[156] G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell. *Engineering Optimization. Methods and Applications*. John Wiley and Sons, 1983.

[157] Robert G. Reynolds. An introduction to cultural algorithms. In *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131–139, River Edge, 1994. New Jersey, World Scientific.

[158] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some guidelines for genetic algorithms with penalty functions. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)*, pages 191–197, San Mateo, California, June 1989. George Mason University, Morgan Kaufmann Publishers.

[159] Rodolphe G. Le Riche, Catherine Knopf-Lenoir, and Raphael T. Haftka. A segregated genetic algorithm for constrained structural optimization. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, pages 558–565, San Mateo, California, July 1995. University of Pittsburgh, Morgan Kaufmann Publishers.

[160] Günter Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5:96–101, January 1994.

[161] Günter Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovac, Hamburg, Germany, 1997. ISBN 3-86064-554-4.

[162] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.

[163] Thomas Philip Runarsson and Xin Yao. Evolutionary search and constraint violations. In *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, volume 2, pages 1414–1419, Piscataway, New Jersey, December 2003. Canberra, Australia, IEEE Service Center.

[164] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[165] Martin Schmidt and Zbigniew Michalewicz. Test-case generator TCG-2 for nonlinear parameter optimisation. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Proceedings of 6th Parallel Problem Solving From Nature (PPSN VI*

), pages 539–548, Heidelberg, Germany, September 2000. Paris, France, Springer-Verlag. Lecture Notes in Computer Science Vol. 1917.

[166] Marc Schoenauer and Zbigniew Michalewicz. Evolutionary computation at the edge of feasibility. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature (PPSN IV)*, pages 245–254, Heidelberg, Germany, September 1996. Berlin, Germany, Springer-Verlag.

[167] Marc Schoenauer and Spyros Xanthakis. Constrained GA optimization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pages 573–580, San Mateo, California, July 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[168] Martin Schütz and Joachim Sprave. Application of partially mixed-integer evolution strategies with mutation rate pooling. In Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck, editors, *Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP'96)*, pages 345–354, San Diego, California, February 1996. The MIT Press.

[169] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Great Britain, 1981.

[170] Hans-Paul Schwefel, editor. *Evolution and Optimization Seeking*. John Wiley & Sons, New York, 1995.

[171] H.P. Schwefel. Projekt MHD-staustrahlrohr: Experimentelle optimierung einer zweiphasendüse, teil I. Technical Report Technischer Bericht 11.034/68, 35, AEG Forschungsinstitut, Berlin, 1968.

[172] Frank Schweitzer, Werner Ebeling, Helge Rosé, and Olaf Weiss. Optimization of road networks using evolutionary strategies. *Evolutionary Computation*, 5(4):419–438, 1998.

[173] Alice E. Smith and David W. Coit. Constraint handling techniques—Penalty functions. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.

[174] Alice E. Smith and David M. Tate. Genetic optimization using a penalty function. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pages 499–503, San Mateo, California, July 1993. University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers.

[175] Stephen Smith. Using evolutionary algorithms incorporating the augmented lagrangian penalty function to solve discrete and continuous constrained non-linear optimal control

problems. In Pierre Collet, Cyril Fonlupt, Jin-Kao Hao, Evelyne Lutton, and Marc Schoenauer, editors, *Proceedings of the 5th International Conference on Artificial Evolution (AE 2001)*, pages 295–308, Heidelberg, Germany, October 2001. Le Creusot, France, Springer-Verlag. Lecture Notes in Computer Science Vol. 2310.

[176] Patrick D. Surry and Nicholas J. Radcliffe. The COMOGA method: Constrained optimisation by multiobjective genetic algorithms. *Control and Cybernetics*, 26(3):391–412, 1997.

[177] Gilbert Syswerda. Uniform crossover in genetic algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Alg orithms*, pages 2–9, San Mateo, California, jun 1989. George Mason University, Morgan Kaufmann Publishers.

[178] Gilbert Syswerda. Schedule optimization using genetic algorithms. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, chapter 21, pages 332–349. Van Nostrand Reinhold, New York, New York, 1991.

[179] Benjamin W. Wah and Yixin Chen. Hybrid constrained simulared annealing and genetic algorithms for nonlinear constrained optimization. In *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, volume 2, pages 925–932, Piscataway, New Jersey, May 2001. IEEE Service Center.

[180] A. Wetzel. *Evaluation of Effectiveness of Genetic Algorithms in Combinatorial Optimization*. PhD thesis, University of Pittsburgh, 1983.

[181] D. Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. David Schaffer, editor, *Proceedings of the Third Conference on Genetic Algorithms*, pages 116–121, San Mateo, California, jun 1989. George Mason University, Morgan Kaufmann Publishers.

[182] D. Whitley, S. Rana, and R. Heckendorn. Representation issues in neighborhood search and evolutionary algorithms. In D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications*, chapter 3, pages 39–57. John Wiley and Sons, West Sussex, England, 1998.

[183] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[184] Baolin Wu and Xinghuo Yu. Fuzzy penalty function approach for constrained function optimization with evolutionary algorithms. In *Proceedings of the 8th International Conference on Neural Information Processing*, pages 299–304, Shanghai, China, November 2001. Fudan University Press.

[185] Quanshi Xia. Global optimization test problems. Available at http://www.mat.univie.ac.at/~neum/glopt/xia.txt.

[186] Jing Xiao, Zbigniew Michalewicz, and Krzysztof Trojanowski. Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation*, 1(1):18–28, 1997.

[187] T. Yokota, M. Gen, K. Ida, and T. Taguchi. Optimal design of system reliability by an improved genetic algorithm. *Transactions of Institute of Electronics, Information and Computer Engineering*, J78-A(6):702–709, 1995. (In Japanese).