Differential Evolution Algorithms for Constrained Global Optimization

Zaakirah Kajee-Bagdadi

A thesis submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg in fulfillment of the requirements for the degree of Master of Science.

Johannesburg, 2007

Declaration

I declare that this thesis is my own, unaided work. It is being submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg. It has not been submitted before for any other degree or examination in any other University.

_____

SIGNATURE OF CANDIDATE

_____

DATE

Abstract

In this thesis we propose four new methods for solving constrained global optimization problems. The first proposed algorithm is a differential evolution (DE) algorithm using penalty functions for constraint handling. The second algorithm is based on the first DE algorithm but also incorporates a filter set as a diversification mechanism. The third algorithm is also based on DE but includes an additional local refinement process in the form of the pattern search (PS) technique. The last algorithm incorporates both the filter set and PS into the DE algorithm for constrained global optimization. The superiority of feasible points (SFP) and the parameter free penalty (PFP) schemes are used as constraint handling mechanisms.

The new algorithms were numerically tested using two sets of test problems and the results where compared with those of the genetic algorithm (GA). The comparison shows that the new algorithms outperformed GA. When the new methods are compared to each other, the last three methods performed better than the first method i.e. the DE algorithm. The new algorithms show promising results with potential for further research.

Keywords: constrained global optimization, differential evolution, pattern search, filter method, penalty function, superiority of feasible points, parameter free penalty.

Acknowledgments

I begin in the name of ALLAH (swt), the Creator and Sustainer of the Universe, for him is all praise.

I would like to express my heartfelt gratitude to my supervisor, Professor Montaz Ali. It was his encouragement, commitment and support that led me to do this thesis. I sincerely appreciate the many hours he has spent to make this thesis possible. His insight, critique and comments have been invaluable.

To my parents, I cannot fully express my gratitude. Thank you for all the support, encouragement and patience that you have shown over these years. Many thanks to the rest of my family, siblings, in-laws and friends for the encouragement and support. To my husband and soul mate, you have just been incredibly supportive and committed to my success. Thank you for everything.

I would also like to acknowledge the financial support granted by the NRF.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The field of optimization has been the focus of much attention in recent years. Optimization techniques and concepts are not limited to any particular discipline and are playing an increasingly important role in the solution and modeling of engineering, economic, design and scientific systems. Optimization is viewed as a decision problem that involves finding the best values of the decision variables over all possibilities. The best values would give the smallest objective function value for a minimization problem or the largest objective function value for a maximization problem. In terms of real world applications, the objective function is often a representation of some physically significant measure such as profit, loss, utility, risk or error. Hence optimizing the system or design to make it as effective or functional as possible is an important part of the overall application.

1.1 Problem description

The general optimization problem can be mathematically represented as:

$$\text{minimize} \quad f(x) \quad \text{subject to} \quad x \in \Omega. \tag{1.1}$$

The function $f : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$ is a real valued objective function. Without loss of generality, we can confine ourselves to minimization problems only since minimizing f is equivalent to maximizing $-f$. The vector x is composed of n independent variables such that $x = [x^1, x^2, \ldots, x^n] \in \mathbb{R}^n$. These variables are the decisions variables and the set $\Omega \in \mathbb{R}^n$ is the feasible set. The vector x that results in the smallest objective function value is referred to as the *minimizer*. Minimizers are further classified according to the following definitions.

Local minimizer: A point $x^* \in \Omega$ is a *local minimizer* of f if there exist some $\epsilon > 0$ such that,

$$f(x) \geq f(x^*),\ \forall x \in \Omega \backslash \{x^*\} \text{ and } \quad ||x - x^*|| < \epsilon,$$

where $f(x^*)$ is known as the local minimum.

Global minimizer: A point $x^* \in \Omega$ is a *global minimizer* of f if

$$f(x) \geq f(x^*),\ \forall x \in \Omega \backslash \{x^*\},$$

where $f(x^*)$ is known as the global minimum. The difference between a local and the global minimum is presented in Figure 1.1.

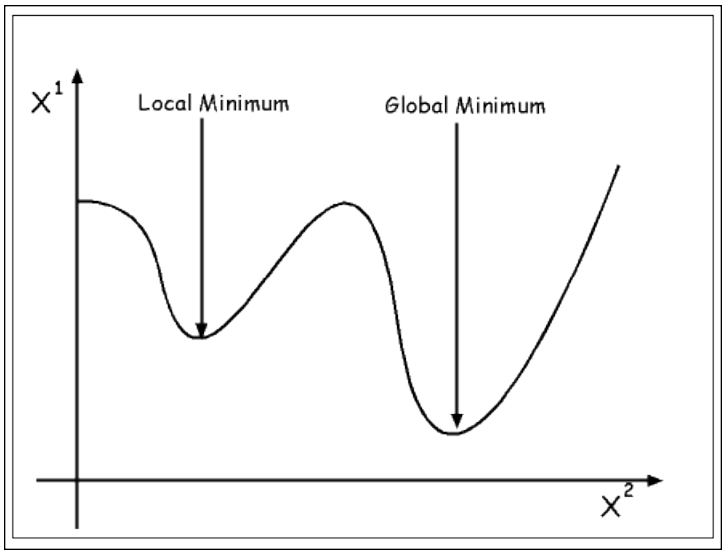

Figure 1.1: *Global vs local minima*

Given a function that contains multiple minima on its feasible set only the smallest minimum will be the global minimum and all others will be classified as local minima. In general, global minimizers are difficult to locate and verify. The task of locating the global optimum is referred to as global optimization.

Optimization problems can be categorically differentiated according to the various properties of the objective function, constraints and decision variables. The objective function can either be linear or nonlinear and the constraints, if any are present, are also classified as being either linear or nonlinear. The decision variables can be continuous or discrete or a combination of both. If a problem has a linear objective function and linear constraints it is considered to be a linear optimization problem, whereas if a problem has either a nonlinear objection function or constraints or both, it is classified as a nonlinear optimization problem. These definitions apply to problems with continuous decision variables. If however the decision variable is discrete the problem is classified as a discrete optimization problem and problems containing both discrete and continuous variables are called mixed-integer problems. For the purpose of this thesis we limit ourselves to continuous variables only.

Nonlinear optimization problems arise frequently in many applications. Hence finding methods to effectively solve these problems is important. Also, if the problems are characterized by multi modal objective functions then finding solutions for this type of problem requires global as opposed to local solution techniques. Hence we will focus ourselves on the solution of nonlinear global optimization problems.

The solution of non-linear optimization problems is highly dependent on the underlying mathematical structure of the problem. In addition attributes such as differentiability, Lipschitz continuity and continuity are extremely influential on the selection of a solution method. There are two possible approaches to solving optimization problems, namely: deterministic and stochastic. In the context of global optimization, some stochastic methods are often referred to as heuristics. We will briefly discuss some deterministic and stochastic methods for non-linear optimization problems.

1.2 Deterministic methods

Deterministic methods exploit the underlying mathematical structure of the problem for solving specific problem types. These methods are mathematically concrete and extremely effective within their scope.

The majority of deterministic methods are focused on local optimization. For unconstrained problems various methods such as Trust Region methods, Newton and Quasi-Newton methods and Conjugate Direction methods are used [41]. On the other hand, the Sequential Quadratic Programming methods, Projected Gradient methods and Interior Point methods are designed for constrained local optimization [41].

Deterministic methods that deal specifically with global optimization problems are fairly limited. The most successful are branch and bound methods [32]. These methods work by systematically dividing the feasible region into successively smaller subregions. Locally optimal solutions are found for each of these subregions and the best amongst the local solutions is assumed to be the global optimum. Another similar successful method is the Interval Arithmetic Method [24] that operates on intervals as opposed to points. The Interval Arithmetic Method uses interval arithmetic techniques to isolate stationary points. Other methods include multi-dimensional bi-section method [61] and Lipschtiz Global Optimization developed by Janos Pinter [43].

Deterministic methods, unfortunately, have certain shortcomings which are greatly emphasized in practical applications. Problems that are characterized by features such as:

- a non differentiable (non-smooth) objective function and/or constraints,
- computationally expensive objective function values or exact gradients,
- noisy objective functions or constraints, and
- discontinuous objective functions

are stumbling blocks for deterministic methods. If a problem is multi modal then using a local optimization method for finding the global minimum becomes extremely dependent on choosing a good starting point that is sufficiently close to the global optimum. In most instances the location of the minimizer and number of local minima or saddle points are not known *a priori*. This makes choosing a good starting point impossible. Also if the number of minima is fairly large, deterministic methods that require computing all these minima are very impractical.

Another important issue is that in order to calculate the derivatives and Lipschitz constant, an explicit mathematical expression of the problem is required. This is not always available, as is the case with 'black box' functions. These involve simulation models or the solution of a set of partial differential equations which are usually only represented implicitly in the optimization problem. Many deterministic methods such as the multi-dimensional bisection and Interval Arithmetic are mathematically involved and computationally inefficient. These features make them unpopular with many practitioners. Although some progress has been made on deterministic methods for global optimization, the problem remains intractable.

1.3 Heuristic methods

Heuristic methods represent a broad class of computational global optimization strategies that use novel approaches to intelligent search for optimal values. They are often inspired by physical processes, natural evolution and stochastic events. Some of the salient features of these methods as opposed to their deterministic counterparts include that they are completely problem independent, do not require adherence to any mathematical requirements and are fairly easy to implement. Differentiability, linearity, convexity or Lipschitz continuity are irrelevant as the search is guided by different mechanisms. Even though these methods are unorthodox and have a minimal mathematical basis or convergence guarantee, they have nonetheless proven themselves as effective and practical global optimization strategies hence

increasing the range of solvable optimization problems.

The lack of deterministic global optimization methods is what originally spurned the development of heuristic methods. Not surprisingly, most of these methods are aimed at solving global optimization problems. During the 1970's and 1980's much research was focused of 'two-phase' methods. These are population based iterative schemes that have two distinctive phases, a global phase that identifies a set of points and a local phase that searches the neighborhood of potential points for improving solutions. Examples of these methods include multi-start methods, clustering methods, multilevel single linkage [48, 49] and topographical multi-level single linkage [3]. Other population based methods that do not use 'two-phase' strategies include the genetic algorithm (GA) [22], differential evolution (DE) [55], particle swarm optimization [58], controlled random search [4] and ant colony [17] . These methods use sophisticated mechanisms to manipulate the population set at each step to create an improved population. The main difference between population based methods that do not include 'two-phase ' strategies is how new members are created for the population. Stochastic methods are not only limited to population based methods but also include single solution or point based methods such as simulated annealing [50], tabu search [21] and hit-and-run based methods [60]. Another example of a single solution method is the DIRECT method [27].

The nature of stochastic methods also lends them to be easily hybridized. Attractive features from different algorithms can easily be incorporated to produce new improved algorithms. A common hybridization technique is to combine local search techniques with global strategies [1]. The local search quickly locates local minima while the global strategy ensures that the search does not get trapped in local minima.

1.4 Thesis outline

The focus of this thesis is the solution of constrained global optimization problems using the differential evolution algorithm [55] as an underlying global solver. Although some research

has been done on the use of heuristic methods for constrained global problems, most has been focused around the use of the genetic algorithm [15, 38, 46, 53]. Our first goal is to provide a DE algorithm for constrained global optimization that uses penalty functions for constraint handling. We will then carry out numerical testing on this new algorithm and compare our results with those obtained by GA [38]. We also wish to explore the use of the filter [19] and pattern search [8, 34, 35, 56] methods to improve the differential evolution algorithm. The filter method will be used to provide a diversification mechanism while the pattern search method will be used for local exploration. Overall we will present the DE algorithm for constrain optimization along with 3 additional hybrid methods based on the DE algorithm.

The thesis is organized as follows: Chapter 2 outlines the differential evolution algorithm as it is implemented for unconstrained optimization. We also briefly present GA [38] in this section as this will facilitate a more complete understanding of the differences between DE and GA. In chapter 3 we formally introduce the constrained optimization problem and discuss penalty methods for constraint handling. The filter method is discussed in Chapter 4 and Chapter 5 centers around the pattern search method. In Chapter 6 we provide a detailed description of our proposed DE based new approaches. Chapter 7 contains the numerical results obtained together with an analysis and comparison of these results. In Chapter 8 we make some concluding remarks.

Chapter 2

Differential evolution

In this chapter we present the DE algorithm for unconstrained global optimization. Before we give a detailed description of the DE algorithm we look at a few general requirements of a global optimization solver and state how the DE algorithm can fulfill these requirements. Depending on the intended use of a global optimization solver, there are many aspects that need to be considered before an appropriate solver is selected. We have imposed the following requirements on the solver:

- Generality: The solver should be insensitive to the underlying problem structure. This will allow it to be applicable to a larger problem set.
- Reliability: The global optimum should be found with a reasonable degree of accuracy.
- Efficiency: The computational complexity of the algorithm should ensure that the algorithm is viable for small to moderate problems e.g. problems with dimensions of up to 100.
- Ease of use: The algorithm should be inherently simple to understand and implement. The number of parameters should also be limited so that too much fine tuning is not required for the algorithm to perform well.

By considering all the above requirements the DE algorithm appears to be one of the most appealing choices as an underlying global optimizer. Initially introduced by Storn and Price [55], the algorithm has undergone substantial testing and modification to improve its performance and applicability [31, 2]. The DE algorithm has proven itself to be an extremely robust and efficient evolutionary type algorithm. What follows is a description of the algorithm.

2.1 Description of the DE algorithm

The DE algorithm can be described as an evolutionary type, stochastic optimization algorithm. As with all evolutionary algorithms, it operates using a set or population $S = \{x_1, x_2, \ldots, x_N\}$ of potential solutions or points to explore the solution space. The size of the population, given by the value N, remains constant throughout. At each generation the algorithm aims to create a new population by replacing points in the current population S with better points. In essence the population is simply a set of points $x_{i,g}$, where i is the index of the member in the population and g indicates the generation or iteration to which the population belongs. Each $x_{i,g}$ consists of n components, where n is the dimension of the problem. Through a repeated process of reproduction (mutation and crossover) and selection, the population S is guided toward the global minimum. We will now take a detailed look at the different processes involved in the DE algorithm:

INITIALIZATION

The first step is to initialize the population. In general, every member of the population is seeded uniformly within a given hyber box. Most problems are considered to be box constrained since the variables are subject to boundary constraints. This leaves us with the following simple initialization formula for each component:

$$x_{i,0}^j = l^j + rand \times (u^j - l^j)\ ,\ j = 1, 2, \ldots, n,\ \ \forall\ i, \tag{2.1}$$

where $rand \in [0, 1]$ is a uniformly distributed random value generated for each j and u^j and l^j are the respective upper and lower limits for the j^{th} variable or component. For certain problems, information might be available that would favor exploration in certain areas. In this cases the population can be seeded around these areas of interest.

MUTATION

The defining characteristic of the DE algorithm is the method via which the new trial points are generated. At every generation g, each member of S is targeted to be replaced with a better trial point. Considering $x_{i,g}$ as the target point, the corresponding trial point $y_{i,g}$ is created using the target point and a mutated point $\hat{x}_{i,g}$. For the simplest case, a mutated point is created by adding the weighted difference of two population members to a third. However there are various other possible schemes for generating the mutated points. Some possible mutation schemes for the i^{th} target point are given below:

$$\hat{x}_{i,g} = x_{p(1)} + F \times (x_{p(2)} - x_{p(3)}), \tag{2.2}$$

$$\hat{x}_{i,g} = x_b + F \times (x_{p(2)} - x_{p(3)}), \tag{2.3}$$

$$\hat{x}_{i,g} = x_{p(1)} + \lambda \times (x_b - x_{p(1)}) + F \times (x_{p(2)} - x_{p(3)}), \tag{2.4}$$

where F and λ are scaling parameters and x_b is the best point in the current population. $x_{p(1)}$, $x_{p(2)}$ and $x_{p(3)}$ are randomly chosen points such that $p(1) \neq p(2) \neq p(3) \neq i$ i.e. all points are unique and none of these points corresponds to the target point $x_{i,g}$. Figure 2.1 illustrates the location of $\hat{x}_{i,g}$ as would be given by equation (2.2).

There are other variants to the schemes described by equations (2.2) to (2.4). In order to distinguish between different schemes a standard notation is used to indicate the scheme type: $DE/a/b/c$. The variable a specifies the base vector used that will be perturb is chosen. It can which can either be random e.g. $x_{p(1)}$, as is the case for equation (2.2) and (2.4) or the best vector is the population, x_b, as in equation (2.3). The second variable b indicates how many vector pairs form the difference vectors. For equations (2.2) and (2.3) the value for b

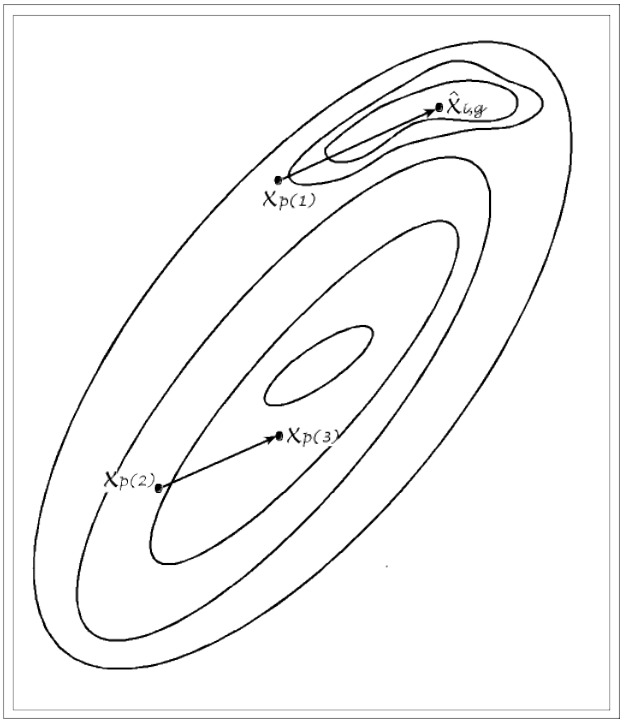

Figure 2.1: *Mutation using equation (2.2)*

is 1 while for equation (2.4) b is 2. The variable c indicates what type of crossover method is used. Binomial crossover is represented by the abbreviation bin and exponential crossover by exp. We will discuss the crossover process below.

CROSSOVER

The target or parent point $x_{i,g}$ together with the new mutated point $\hat{x}_{i,g}$ are recombined to create the trial point $y_{i,g}$. There are two popular types of crossover methods used with the DE algorithm, namely binomial and exponential. For the purpose of this thesis we only use the binomial method which will be discussed below.

Binomial recombination starts at the first component of the vector and generates a random number $r^j \in [0, 1]$ for each component. If $r^j \leq c_r$ then the j^{th} component of $y_{i,g}$ is taken from $x^j_{i,g}$, otherwise if $r^j > c_r$ then the component is taken from $\hat{x}_{i,g}$. This process continues until all components from $x_{i,g}$ have been considered. In order to ensure that at least one component in $y_{i,g}$ is from $x_{i,g}$, a random integer $I_i \in \{1, 2, \ldots, n\}$ is generated. The component in $y_{i,g}$ corresponding to I_i is taken from $\hat{x}_{i,g}$. The trial vector can contain components from $\hat{x}_{i,g}$ at multiple, separated points. We refer to Figure 2.2 for an illustration of this,

where $y_{i,g} = \left(x^1_{i,g}, \hat{x}^2_{i,g}, x^3_{i,g}, \hat{x}^4_{i,g}, \hat{x}^5_{i,g}, x^6_{i,g}\right)$. Binomial recombination can be mathematically formulated as:

$$y^j_{i,g} = \begin{cases} \hat{x}^j_{i,g} & if\ r^j \leq c_r \text{ or } j = I_i, \\ x^j_{i,g} & otherwise. \end{cases} \tag{2.5}$$

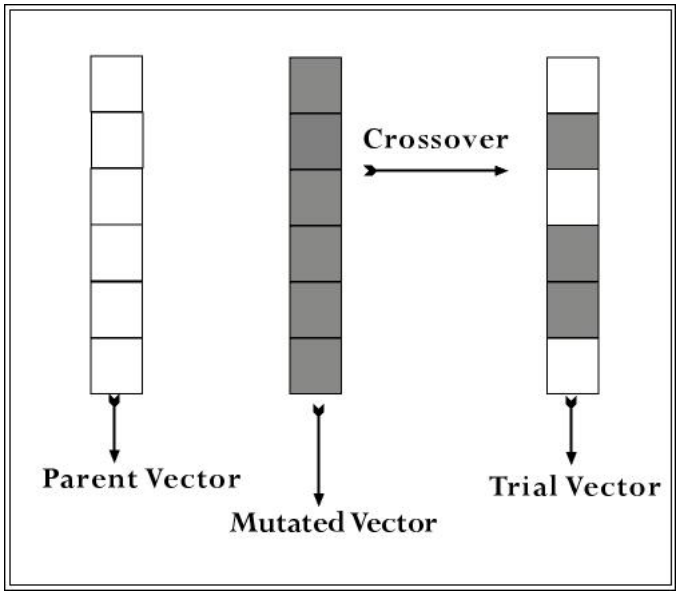

Figure 2.2: *Binomial Crossover*

ACCEPTANCE

At each iteration the DE algorithm attempts to replace each point in S with a better point. Therefore at each generation g, N competitions are held to determine the members of S for the next iteration. The i^{th} competition is held to replace $x_{i,g}$ in S. This is done by comparing the function values of the trial points $y_{i,g}$ to those of $x_{i,g}$, the target points. If $f(y_{i,g}) < f(x_{i,g})$ then $y_{i,g}$ replaces $x_{i,g}$ in S, otherwise S retains the original $x_{i,g}$. This can be written mathematically as:

$$x_{i,g+1} = \begin{cases} y_{i,g} & if\ f(y_{i,g}) < f(x_{i,g}), \\ x_{i,g} & otherwise. \end{cases} \tag{2.6}$$

The DE algorithm maintains a greedy selection scheme that ensures that the current generation is equal to or better than the previous generation.

STOPPING CRITERIA

An important aspect for a stochastic algorithm is deciding when to stop the algorithm. We know that stochastic methods converge with a probability of 1 to an optimal value as time goes to infinity [57]. However upholding such a convergence guarantee is impractical. Therefore the user will need to decide on some preset conditions that will terminate the algorithm. Deciding on what stopping criteria to use is dependent on many factors such as the application of the algorithm, accuracy required, cost and time constraints. Some of the most common stopping criteria used for the DE algorithm include:

- a preset number of maximum generations,
- the difference between the best and worst function values in the population is very small,
- the best function value has not improved beyond some tolerance value for a predefined number of generations,
- the distance between solution vectors in the population is small.

We have thus far described the basic steps involved. Algorithm 1 gives the elementary pseudo code for the DE algorithm for bound constrained optimization only.

2.2 Parameter selection

The parameters N, c_r and F are central to the overall performance of the DE algorithm. For different problems, simply varying the parameters can greatly improve or hinder the performance of DE. In the original paper that introduces the DE algorithm, the suggested value for the scaling parameter F was in the range $[0, 2]$ [55]. However empirical testing has shown that for most problems the optimal value for F is in the range $[0.4, 1]$ [5, 42]. Further suggestions regarding F have been to randomize F for each mutation point and increase exploration by having F in $[-1, -0.4] \cup [0.4, 1]$ [2].

Algorithm 1 The DE algorithm for unconstrained optimization

1. Set control parameters N, c_r, F and $g = 0$
2. Initialize Population, $S = \{x_{1,0}, x_{2,0}, \ldots, x_{N,0}\}$ using (2.1)
3. Evaluate objective function f for each member in the population
4. **IF** Stopping Criteria not met
   (a) **FOR** $i = 1$ **TO** N,
      i. generate trial point $y_{i,g}$ via:
         - **Mutation** using (2.2), (2.3) or (2.4)
         - **Crossover** using (2.5)
      ii. Evaluate $f(y_{i,g})$

   END
5. Update population using (2.6)
6. Set $g = g + 1$, calculate stopping criteria and go to 4.

The crossover parameter $c_r \in [0, 1]$ is used to control the diversity of the trial vector. Higher values of c_r results in faster convergence. In general $c_r = 0.5$ is suggested as a good choice for most unconstrained problems [2, 6]. The population size, N, is often determined by the dimension of the problem. A popular setting for N is $N = 10 \times n$, where n is the dimension of the problem. However values smaller than $10 \times n$ may be used when the dimension of the problem is very high.

All the suggested parameter values have been found by empirically testing the DE algorithm on unconstrained problems only. However it is intuitive that the choice of parameter values will be affected by the presence of constraints. If the DE algorithm was to be applied to a single problem only, the obvious choice would be to empirically find the best combination of parameter values. However, our aim is to create a general purpose solver, hence we wish to obtain a set of parameter values that performs efficiently on most problems. Indeed it will be shown in Chapter 7 that good parameter values for the DE algorithm for constrained global optimization are different than for unconstrained global optimization. The pseudo code and implementation of the DE algorithm for constrained global optimization will be discussed in Chapter 6, with numerical results presented in Chapter 7.

2.3 Genetic algorithm (GA)

The GA method is one of the most popular and successful stochastic search methods. It falls into the broad class of evolutionary algorithms. Since we will be comparing the results for our proposed DE based algorithms to the results obtained by GA [38], we will give a very brief description of the GA algorithm and highlight some difference between both these evolutionary algorithms. GA algorithm presented here is in the form implemented in [38].

Just as with DE, the real coded GA involves maintaining a population S of N points. At each generation GA replaces a portion of points in S with better points that have been obtained via the process of selection, crossover and mutation. The basic steps involved are outlined below.

Initialization An initial population is created by generating N random points from the search space. This initialization process is similar to the one described for the DE algorithm by equation (2.1).

Selection This process involves choosing the best individuals as parents for the crossover operator. Different techniques such as tournament selection or roulette-wheel selection can be used to achieve this. The tournament selection method was used in [38]. In this process a preset number of individuals, determine by the *tournament size* parameter, are compared. The best individual amongst them is selected as the parent.

Crossover This involves the recombination or cross breeding of information between 2 parents to create offspring(s) for the next generation. The *crossover rate* is the parameter that controls this process. The heuristic crossover method was used in [38].

Mutation Mutation means that the new offspring are modified with some probability determined by a parameter, called a *mutation rate*. The reader is referred to [38] for full details of the mutation process as well as the *mutation exponent* parameter p.

Elitism To ensure that the population contains the best solution produced so far the best individual(s) is (are) copied to the next generation. This is referred to as elitism and

the the number of best individuals copied to the next generation is determined by the *elitism size* parameter.

The basic real coded GA algorithm is presented below:

Algorithm 2 The real coded GA

---

1. Set control parameters N, tournament size, crossover rate, mutation rate, mutation exponent, elitism size and $g = 0$
2. Initialize Population, $S = \{x_{1,0}, x_{2,0}, \ldots, x_{N,0}\}$
3. Evaluate objective function f for each member in the population
4. **IF** Stopping Criteria not met
   (a) Generate Offspring via:
   - Selection
   - Crossover
   - Mutation

   (b) Carry out elitism and update the population by replacing the parents by offspring.

   END
5. Set $g = g + 1$, calculate stopping criteria and go to 4.

The real coded GA [38] incorporates constraint handling techniques, such as penalty functions when applied to constrained global optimization problems. Despite both GA and DE belonging to the class of evolutionary stochastic algorithms there are many differences between them. Next we outline some of these differences.

- GA selects two parents for crossover and the child is a recombination of the parents. The child is then mutated with some probability. In the DE algorithm, at least three parents are selected and a mutated point is created that is a perturbation of one of them e.g. as in (2.2) to (2.4). The child is a recombination of the parent and the mutated point.
- The DE algorithm aims to replace all points in the current population at each generation, where as GA only replaces a subset of the population at each generation.

- The DE algorithm relies on a point to point comparison. If a trial point is worst than the target but is better than the rest of the current population, it is still rejected regardless of its comparative superiority. GA on the other hand targets the worst points in the population and replaces them with children.

- The DE algorithms has 3 parameters: the population size, crossover rate and a scaling parameter. GA has 6 parameters that include the population size, tournament size, crossover rate, mutation rate, mutation exponent and elitism size.

Chapter 3

Constrained global optimization

Most real systems are often subject to constraints. These are manifestations of physical, mathematical or design restrictions placed on the system e.g. gravity, stress, cost etc. These are interpreted as constraints on the resulting mathematical model. The imposition of constraints often causes the location of the minimum of a problem to change as has been illustrated in Figure 3.1. We extend on the general optimization problem given by equation (1.1) to facilitate the imposition of constraints:

$$\text{minimize} \quad f(x)$$

subject to

$$g_k(x) \leq 0, \quad k = 1, \ldots, K,$$

$$h_l(x) = 0, \quad l = 1, \ldots, L, \quad L < n,$$

$$l^j \leq x^j \leq u^j, \quad j = 1, \ldots, n. \tag{3.1}$$

In the above equation $g_k(x)$ is the k^{th} inequality constraint and $h_l(x)$ is the l^{th} equality constraints. Each variable lies within its respective range $[l^j, u^j]$. The feasible region is

therefore given by:

$$\Omega = \left\{ x = [x^1, x^2, \ldots, x^n] \in \mathbb{R}^n \mid g_k(x) \leq 0 \ , \ h_l(x) = 0 \ , \ l^j \leq x^j \leq u^j \ , \ \forall j \right\}$$

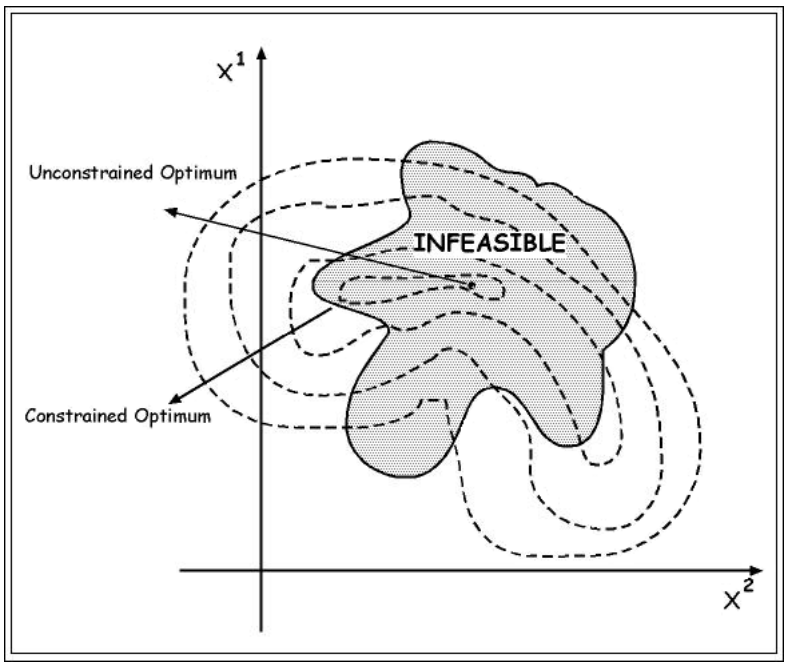

Figure 3.1: *Location of optima in constrained vs unconstrained optimization*

Since this thesis focuses on the DE algorithm, which is an evolutionary algorithm we will only look at constraint handling techniques that are compatible with evolutionary algorithms. Michalewicz and Schoenauer [37] have classified constraint handing techniques for evolutionary algorithms into four categories:

- techniques based on preserving the feasibility of solutions,
- techniques based on penalty functions,
- techniques that distinguish between feasible and infeasible solutions, and
- other hybrid techniques [47, 59] .

The first two approaches are undoubtedly more popular than the last two and we will only discuss these. Here we briefly present the first approach. Techniques based on preserving the

feasibility of solutions fall into two groups. The first group involves the use of specialized operators to transform infeasible individuals to feasible individuals. The Genocop system [37] is an example of a method from this group. It is however restricted to problems with linear constraints. The second group operates by restricting the search process to the boundary of the feasible space. This is because often the solutions to constrained problems lie on the boundary of the feasible space. A major drawback of both these groups is that they require a feasible starting point or population to begin with. In constrained optimization the feasible region for many problems is extremely small, often comprising a very small portion of the entire search space. Hence finding a feasible starting point is a difficult problem in itself [37].

The second popular approach to constraint handling is the use of penalty functions. Penalty functions avoid the pitfalls of feasibility preserving methods but have their own setbacks. The introduction of a penalty results in new parameters that need to be determine for each problem. This is in itself a difficult problem since users rarely have the required problem specific information available *a priori* to help select the best or even acceptable penalty parameters. These then have to be determined empirically. Penalty parameters are also known to directly influence the convergence properties of the underlying algorithm [37]. In evolutionary algorithms 'over penalization' of constraints results in poor exploration of the search region and premature convergence to solutions that are usually suboptimal. On the other hand, 'under penalization' results in very slow convergence toward feasible solutions. There is also no guarantee that a feasible solution will be found [12].

Despite the above mentioned shortcomings of penalty functions, they are still the most popular methods for solving constrained optimization problems. This in part can be attributed to their simplicity and the ease with which they can be incorporated into an existing algorithm. For the purpose of this thesis we will be using the penalty function approach for dealing with constraints. Next, we will look at a general problem reformulation to include penalty functions.

3.1 Penalty functions

In order to accommodate the penalty function, the objective function that is to be minimized is reformulated as follows:

$$\hat{f}(x) = f(x) + R\left(\sum_{k=1}^{K}\langle g_k(x)\rangle^q + \sum_{l=1}^{L}\langle h_l(x)\rangle^q\right), \tag{3.2}$$

where

$$\langle g_k(x)\rangle = max\left\{0, g_k(x)\right\}.$$

$$\langle h_l(x)\rangle = \left|h_l(x)\right|.$$

In this formulation, the fitness function $\hat{f}(x)$ combines the objective function value with a term that penalizes any constraint violation. The parameter R is the penalty parameter. For some implementations R can be iteration dependent and will take on different values at each iteration. The parameter q takes on the value of 1 for an exact penalty or 2 for a quadratic penalty. The function given in equation (3.2) can also be re-written by converting each equality constraint into two inequality constraints. This is done by including a small tolerance value δ to the inequality. The l^{th} equality constraint can be converted into the following two inequalities:

$$g_l(x) = h_l(x) - \delta \leq 0 \quad and \tag{3.3}$$

$$g_{l+1}(x) = -h_l(x) - \delta \leq 0 \tag{3.4}$$

where δ is a small positive value. This transformation will increase the total number of inequality constraints to $d = K + 2L$ and simplify the fitness function (3.2) to:

$$\hat{f}(x) = f(x) + R \times G(x), \tag{3.5}$$

where

$$G(x) = \sum_{k=1}^{d} \langle g_k(x) \rangle^q . \tag{3.6}$$

For the purpose of this thesis we limit ourselves to an exact penalty given by $q = 1$ for all our implementations and we will therefore be using (3.5) and (3.6) as our fitness function and constraint violation function respectively.

Often penalty functions vary on how the penalty coefficient, R, is calculated. Properties such as the number of violated constraints, level of violation, distance from feasible region etc. are used to determine penalties. Some of the most well known approaches to penalty functions include:

Static penalties: For each constraint several levels of violation are defined. Each level of violation has a value of R associated with it.

Dynamic penalties: These are time dependent penalties. As the number of iteration increases so does the penalty value.

Annealing penalties: A cooling scheme is used to determine the penalty value at each iteration.

Adaptive penalties: These have penalty function components that adjust depending on the search process.

Death penalties: This is a barrier method that simply rejects all infeasible individuals.

In this research we will focus on two specific adaptive penalty schemes that are explored in Miettinen et al [38]. They are the superiority of feasible points (SFP) scheme and the parameter free penalty (PFP) scheme.

3.1.1 The superiority of feasible points scheme

The superiority of feasible points (SFP) scheme was developed by Powell and Skolnik [46] and is based on the static penalty method but includes an additional term in the fitness function. The purpose of this additional function is to ensure that infeasible points always have worst fitness values than feasible points. Hence each infeasible individual is evaluated not only by its objective function, f, and penalty, G, but also by an additional iteration dependent function. At generation g, for population $S_g = \{x_{1,g}, x_{2,g}, \ldots, x_{N,g}\}$ the new fitness function is given by:

$$\hat{f}(x_{i,g}) = f(x_{i,g}) + R \times G(x_{i,g}) + \Theta_g(x_{i,g}), \;\; x_{i,g} \in \;\; S_g \tag{3.7}$$

where

$$\Theta_g(x_{i,g}) = \begin{cases} 0 & if \; S_g \cap \Omega = \emptyset \;\; or \;\; x_{i,g} \in \Omega, \\ \alpha & if \; S_g \cap \Omega \neq \emptyset \;\; and \;\; x_{i,g} \notin \Omega. \end{cases} \tag{3.8}$$

The value α is calculated by:

$$\alpha = max \left[0, \max_{y \in S_g \cap \Omega} f(y) - \min_{z \in S_g \setminus \Omega} [f(z) + R \times (G(z))] \right]. \tag{3.9}$$

The function Θ_g is used to penalize infeasible points only when the population already contains feasible points. This is given by the second term in (3.8). Hence when the population contains feasible members, the infeasible members will always be worst than the worst feasible member. This will ensure that infeasible points are never 'under penalized'. Clearly $f(x_{i,g})$ and $G(x_{i,g}) \; \forall \;\; x_{i,g} \in S_g$ must be known before α can be calculated.

To illustrate the SFP scheme we will look at an example using Problem 7 of Appendix A. Table 3.1 is a sample population of 5 points where all points are infeasible. Using a penalty coefficient of $R = 100$, the fitness value $\hat{f}(x)$ calculated by SFP for each point is simply:

$$\hat{f}(x) = f(x) + 100 \times G(x). \tag{3.10}$$

This is because when S_g does not contain any feasible points i.e. $S_g \cap \Omega = \emptyset$, then $\Theta_g = 0, \ \forall x \in S_g$. Using (3.10) we give the data for the infeasible population in Table 3.1.

i	x_i	$f(x_i)$	$G(x_i)$	$\hat{f}(x_i)$
1	(1.316 , 1.9932)	-3.3092	0.8605	82.740
2	(0.9601 , 3.8404)	-4.8005	3.8139	376.591
3	(1.8216 , 2.5196)	-4.3412	0.3084	26.497
4	(0.7853 , 2.3894)	-3.1747	1.4851	145.336
5	(1.0809 , 2.1941)	-3.2750	2.0977	206.493

Table 3.1: Example of SFP scheme: population with no feasible points

The objective of the SFP scheme is to ensures that points with the higher constraint violations have the worst fitness values irrespective of the objective function value. This can be seen in Table 3.1 for x_2 which has the worst constraint violation in the population. The third point, x_3, in Table 3.1 becomes the best point since it has the lowest constraint violation. This shows that the scheme gives preference to obtaining feasibility over good objective function values.

Next we illustrate the SFP scheme where the population is composed of both feasible and infeasible points. Using the same problem as above we present an example in Table 3.2. In this case, calculating the fitness function becomes slightly more complex. For each infeasible point $x_{i,g} \in S_g$, $\Theta_g(x_{i,g}) = \alpha$ and for each feasible point $x_{i,g} \in S_g$, $\Theta_g(x_{i,g}) = 0$. Once the values for the objective function and the constraint violation are calculated, the value for the additional penalty term, Θ_g, can be calculated.

From equation (3.9) we see that the value for α is dependent on the largest feasible objective function value and the smallest $f(z) + R \times G(z)$ for an infeasible point z. In this example the maximum objective function value obtained by a feasible point is $f(x_3) = -0.566$. The minimum value of $f(x) + R \times G(x)$ for the two infeasible points x_2 and x_5 is for the point x_5 as has been calculated in Table 3.1 as $\hat{f}(x_5) = 206.493$. Thus $\alpha = \max[0, -0.566 - 206.493] = 0$, results in the following population:

Table 3.2 shows that all the feasible points have better fitness values than the infeasible points. Also the infeasible point with the largest constraint violation has the worst fitness

i	x_i	$f(x_i)$	$G(x_i)$	$\hat{f}(x_i)$
1	(1.5663 , 0.4682)	-2.0345	0	-2.0345
2	(0.9601 , 3.8404)	-4.8005	3.8139	376.591
3	(0.3048 , 0.2613)	-0.5660	0	-0.5660
4	(2.3295 , 3.1785)	-5.5079	0	-5.5079
5	(1.0809 , 2.1941)	-3.2750	2.0977	206.493

Table 3.2: Example of SFP scheme: population with feasible and infeasible points

value in the population i.e. the point x_2. This again shows that for infeasible points the SFP scheme shows a preference toward obtaining feasibility. Next we look at the parameter free penalty scheme.

3.1.2 The parameter free penalty scheme

The method of parameter free penalty (PFP) scheme was introduced by Deb [15]. The scheme presented here is a modification of the SFP scheme as suggest by Miettinen et al [38]. The most significant feature is the lack of a penalty coefficient R. Therefore the user does not need to supply any parameter values for the scheme. Here, just as with the SFP scheme an additional penalty term, Θ_g, is added to the fitness function. This term ensures that infeasible points are always worst than feasible ones but without the need of a penalty parameter. The fitness function in the PFP scheme is as follows:

$$\hat{f}(x_{i,g}) = f(x_{i,g}) + G(x_{i,g}) + \Theta_g(x_{i,g}), \quad x_{i,g} \in S_g \tag{3.11}$$

where,

$$\Theta_g(x_{i,g}) = \begin{cases} 0 & if \;\; x_{i,g} \in \Omega, \\ -f(x_{i,g}) & if \;\; S_g \cap \Omega = \emptyset, \\ -f(x_{i,g}) + \max_{y \in S_g \cap \Omega} f(y) & if \; S_g \cap \Omega \neq \emptyset \;\; and \;\; x_{i,g} \notin \Omega. \end{cases} \tag{3.12}$$

We note that just as with the SFP scheme, $f(x_{i,g})$ and $G(x_{i,g}) \; \forall \;\; x_{i,g} \in S_g$ must be known before Θ can be calculated. Looking at the first term in (3.12) we can see that if a point is

feasible i.e. $G(x) = 0$, then the fitness value is equal to the objective function value. From the second term, when all members of the population are infeasible i.e. $S_g \cap \Omega = \emptyset$, the fitness value to be minimized consists of only the constraint violation. This directs the search toward a feasible region. However this might adversely affect the convergence because the objective function value is completely neglected in this case. Finally, when the population contains both infeasible and feasible points the PFP scheme ensures that infeasible points are always worst than the worst feasible point. This is done by adding the object function value of the worst feasible point to the constraint violation of each infeasible point. This value is the resulting fitness value. Computationally this method also provides an advantage in that if a point is infeasible only the constraint violation has to be calculated and not the objective function value.

Using the same example as with the SFP scheme we will illustrate the PFP scheme. For this scheme, if a point is infeasible the objective function value does not need to be calculated, however we have included them in the table below. Table 3.3 gives a population of infeasible points for Problem 7. The fitness function value is simply equal to the value of the constraint violation. This will ensure that when all points are still infeasible in the population the main objective will be to find feasible points.

i	x_i	$f(x_i)$	$G(x_i)$	$\hat{f}(x_i)$
1	(1.316 , 1.9932)	-3.3092	0.8605	0.8605
2	(0.9601 , 3.8404)	-4.8005	3.8139	3.8139
3	(1.8216 , 2.5196)	-4.3412	0.3084	0.3084
4	(0.7853 , 2.3894)	-3.1747	1.4851	1.4851
5	(1.0809 , 2.1941)	-3.2750	2.0977	2.0977

Table 3.3: Example of PFP scheme: population with no feasible points

When the population contains both feasible and infeasible points, the fitness function calculations for any infeasible points can be simplified as:

$$\hat{f}(x) = G(x) + \max_{y \in S_g \cap \Omega} f(y) \tag{3.13}$$

Hence the fitness value for all infeasible points will be worst than the worst feasible point. In Table 3.4 the largest $f(x)$ value for any feasible point in this population is given by $f(x_3) = -0.566$. Using (3.13) the fitness value for x_2 will be calculated as $\hat{f}(x_2) = G(x_2)+f(x_3) = 3.8139 - 0.566 = 3.2479$. The resulting fitness values are given in Table 3.4.

i	x_i	$f(x_i)$	$G(x_i)$	$\hat{f}(x_i)$
1	(1.5663 , 0.4682)	-2.0345	0	-2.0345
2	(0.9601 , 3.8404)	-4.8005	3.8139	3.2479
3	(0.3048 , 0.2613)	-0.5660	0	-0.5660
4	(2.3295, 3.1785)	-5.5079	0	-5.5079
5	(1.0809 , 2.1941)	-3.2750	2.0977	1.5317

Table 3.4: Example of PFP scheme: population with feasible and infeasible points

The implementation of these penalty methods in conjunction with the differential evolution algorithm will be fully discussed in Chapter 6.

Chapter 4

The filter algorithm for constrained optimization

The concept of filters was first introduced as a globalization strategy for sequential linear programming (SLP) and sequential quadratic programming techniques (SQP) [19]. The filter algorithm attempts to avoid the shortcomings of penalty functions by decomposing the constrained optimization problem into a bi-objective problem. In essence, instead of combining the objective function and constraint function, the filter method tries to simultaneously minimizes both the functions separately. This can be formalized as:

$$\text{minimize [f(x),G(x)] ,} \tag{4.1}$$

$$\text{where} \quad G(x) = \sum_{k=1}^{d} \langle g_k(x) \rangle \, .$$

The filter algorithm is based on a concept used in multi objective optimization known as dominance. A filter set consisting of a list of pairs $[f(x), G(x)]$ is created such that no pair dominates another. We can define dominance as follows:

Dominance: For a pair of vectors $x = [x^1, x^2, \ldots, x^n]$ and $y = [y^1, y^2, \ldots, y^n]$, with finite components, $x \ \mathit{dominates} \ y$, written as $x \prec y$, if and only if $\forall \ i = 1, \ldots, n, \ x^i \leq y^i$,

and $x \neq y$. The notation $x \preceq y$ is used to indicate that either $x \prec y$ or that $x = y$.

Using the above definition and (4.1), a point $x \in \mathbb{R}^n$ is said to *dominate* $y \in \mathbb{R}^n$ i.e. $x \prec y$, if and only if $[f(x), G(x)] \prec [f(y), G(y)]$. Hence $f(x) \leq f(y)$ and $G(x) \leq G(y)$. If two pairs have the same f and G values the points are considered equivalent. A filter $\mathcal{F}$ is a finite set of points such that no pair of points x, y in the set have the relation $x \prec y$. A point y is called a filtered point if any one of the following holds :

- $y \succeq x$ for some $x \in \mathcal{F}$.
- $G(y) \geq G_{max}$, where $G_{max} > 0$ is a maximum allowed value on the constraint violation function $G(x)$.
- $G(y) = 0$, and $f(y) \geq f^F$, where $f^F = f(x^F)$ is the current minimum feasible function value.

Consequently, the set $\bar{\mathcal{F}}$ of all filtered points y is:

$$\bar{\mathcal{F}} = \bigcup_{x \in \mathcal{F}} \{y : y \succeq x\} \cup \{y : y \geq G_{max}\} \cup \left\{y : G(y) = 0, \ \textit{and} \ f(y) \geq f^F\right\}. \qquad (4.2)$$

In order to illustrate how a filter set would operate we can plot f against G in Figure 4.1. In this figure the filter set is given by $\mathcal{F} = \{x_1, x_2, x_3, x_4\}$. Any infeasible point that is generated that lies in the shaded region would be filtered. All infeasible points generated in the unshaded region would be included in $\mathcal{F}$ and would possibly eliminate some points already in $\mathcal{F}$. For example if the point x_5 were to be added to the filter it would eliminate x_4 from the filter set resulting in $\mathcal{F} = \{x_1, x_2, x_3, x_5\}$. This is because $f(x_5) < f(x_4)$ and $G(x_5) < G(x_4)$ giving the relation $x_5 \prec x_4$. For feasible points, any point with an objective function value greater than x_1 such as x_6 will be filtered. However if a feasible point is found with a lower objective function value it will be added to the filter and x_1 will be removed.

The filter set contains two important points, namely the feasible and infeasible incumbents. f_k^F represents the feasible incumbent and is defined as the smallest objective function

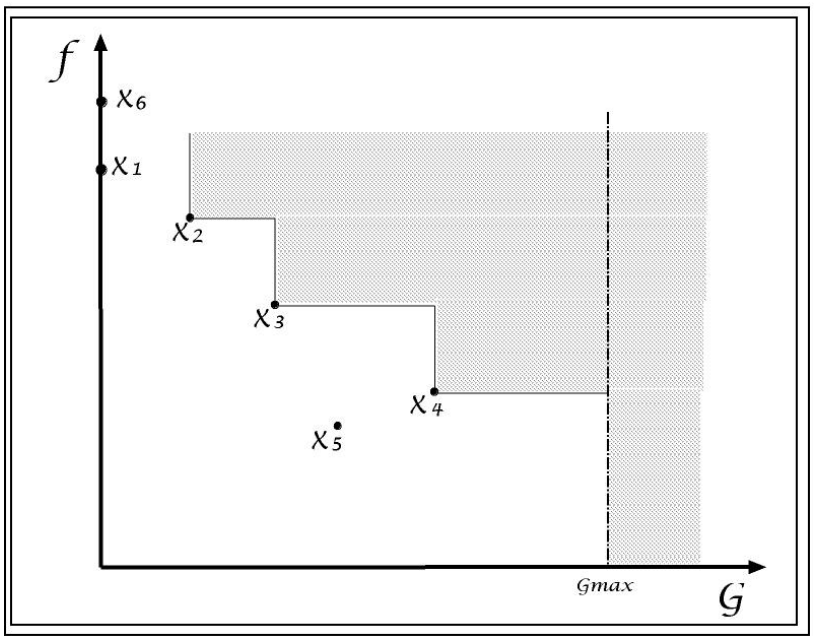

Figure 4.1: *Example of Filter Set*

value for a feasible point found up until iteration k. Let g_k^I be the least positive constraint violation function value found up to iteration k of a filter algorithm. Then f_k^I is the smallest objective function value for a point with its constraint violation equal to g_k^I. In Figure 4.1 the point x_1 would be the feasible incumbent and point x_2 would be the infeasible incumbent. The superscripts F and I signify feasibility and infeasiblity, respectively.

4.1 The filter method for constrained optimization

The use of the filter method in conjunction with evolutionary algorithms has been fairly limited [10, 25]. Next we will discuss two evolutionary methods that employ filters.

In [10] a general evolutionary algorithm is used in conjunction with a filter set. The population is divided into two subsets. One containing feasible points and the other consisting of nondominated infeasible points and the feasible incumbent. The filter is employed implicitly in order to eliminate dominated trial points generated during the search process. The filter concept is used principally as a constraint handling strategy and as an acceptance criteria for

new points. The reader is referred to [10] for further insight. Unfortunately no numerical testing was done for this method, so it would be difficult to judge the accuracy, efficiency or reliability of this method.

Another filter-based global optimization method is the filter simulated annealing (FSA) approach as suggested in [25]. Initially a diverse solution set is generated. The points in the set are then ranked. The ranking procedure used is reliant on a filter set to determine the ranking of points. Based on this ranking process the best point in the set is chosen as a starting point for the annealing process and subsequently when required the annealing process is restarted from this same set. The acceptance criterion is also affected by the filter. All unfiltered points are accepted with a probability of 1. Numerical results are presented in [25] for the FSA algorithm. We have included most of these results in Chapter 7.

Our aim is to take a slightly different approach to the use of the filter set. The filter set will be used in two of the algorithms that will be presented later. We will present our motivation for the proposed use of the filter set below.

Our proposal involves using the mutation process of the DE to explore unfiltered points encountered during the search process. The filter set gives a number of points that have very unique features. These can be considered to be points of interest, since either their objective function value $f(x)$ or constraint value $G(x)$ compares favorably to other infeasible points. When using penalty schemes to handle constraints the schemes focus of minimizing the constraint violation only. The function values are often disregarded completely as is the case for PFP penalty scheme or the penalized constraints overshadow the objective function e.g. in the SFP scheme. By using the filter we want to explore a wider spectrum of points with both high and low objective functions values. We propose using unfiltered points as base vectors during the mutation phase of the DE algorithm. This will allow these points to be better explored, leading to a potentially unexplored feasible region. The mutated vector in DE will thus be generated as follows:

$$\hat{x}_{i,g} = x_{undominated} + F \times (x_{p(2)} - x_{p(3)}) \tag{4.3}$$

where $x_{undominated}$ is an undominated point such that $x_{undominated}$ is selected randomly from $\mathcal{F}$ for each infeasible target point $x_{i,g}$. Trial points generated by the DE algorithm in each generation aim to replace their corresponding target points from the current population. However not all trial points will succeed in replacing their target points. In building our filter set we will consider all trial points irrespective of whether they successfully enter the new population or are rejected. This will create a filter with a substantial number of varying points.

The DE algorithm employs a point to point acceptance rule. This is important since if the target point is feasible the SFP and PFP penalty schemes will ensure that it will only be replaced by a feasible point with a better function value. Hence it is logical to only use the suggested mutation scheme (4.3) when the target points are infeasible. We will further discuss the proposed algorithm in Chapter 6.

Chapter 5

The pattern search method

The pattern search (PS) method falls into the wide class of generalized pattern search (GPS) algorithms. These are derivative free, direct search algorithms for unconstrained optimization. In 1997, Torczon [56] showed that all the existing pattern search algorithms are simply specific implementations of an abstract pattern search scheme. The general scheme involves the construction of a mesh of points, around the current solution. These points are then explored according to some criteria. If the current solution remains unimproved the mesh is refined and the process repeated. Aside from PS other instances of this class include the Hooke and Jeeves method [26], the basic coordinate search method [44] and multidirectional search method [16].

The GPS framework has since been extended to include bound constrained optimization problems [34] and problems with linear inequalities [35]. For problems where f is continuously differentiable, Torzan [56] proved that GPS produces some limit point for which the gradient of the objective function value is zero. It is further proven that for bound constrained [34] and linearly constrained problems [35], the GPS adaptation produces a Karush-Kuhn-Tucker point. For further discussions of these and other similar results refer to [8, 34, 35, 56].

5.1 Description of the PS algorithm

The PS algorithm is an iterative process that aims to generate a sequence of iterates $\{x_k\}$ in $\mathbb{R}^n$ with non-increasing objective function values. This is done by evaluating a finite number of points on a mesh in order to find an improved point. The exploration of the mesh is carried out in either one or two phases. The phases being the SEARCH and POLL steps. In order to better understand these phases we need to formally define important concepts such as positive combination and span [14] and mesh generation [45].

Definition 5.1: Positive combination, Positive Span

1. A positive combination of vectors $\{v_i\}_{i=1}^{p}$ is a linear combination $\sum_{i=1}^{p} \lambda_i v_i$ where $\lambda_i \geq 0, \forall\, i \in \{1, 2, \ldots, p\}\,, \quad n+1 \leq p \leq 2n.$

2. A positive span for a subspace $B \subset \mathbb{R}^n$ is a set of vectors $\{v_i\}_{i=1}^{p}$ such that every $x \in B$ can be expressed as a positive combination of the vectors $\{v_i\}_{i=1}^{p}$. The matrix defined by $V = [v_1, \ldots, v_p]$ is said to be a positive spanning matrix.

3. Let the subspace $B \subset \mathbb{R}^n$ be of dimension m and $V \in \mathbb{R}^{n \times p}$ be a positive spanning matrix for B. If $p = n+1$, then V is said to be a minimal positive spanning matrix for B.

If, for example, $B \subset \mathbb{R}^2$ then $V = [e_1, e_2, -e_1, -e_2]$, where e_1 and e_2 are unit vectors, is a positive spanning matrix. However $V = [e_1, e_2, -(e_1 + e_2)]$ would be a minimal positive spanning matrix for B.

Definition 5.2: Base Direction Matrix

Let $\mathbb{B}$ be the set of all matrices whose columns positively span $\mathbb{R}^n$. Then, the base direction matrix D is any positive spanning matrix satisfying

$$D \in \mathbb{Q}^{n \times p} \cap \mathbb{B}. \tag{5.1}$$

The fact that $\mathbb{Q}^{n\times p}$ is a rational matrix ensures that the matrix D has only rational elements and makes it very easy to establish the minimal distance between distinct mesh points [45].

Definition 5.3: Mesh

$$M(x_k, \Delta_k) = \{x_k + \Delta_k Dm : m \in \mathbb{N}^p\} \tag{5.2}$$

where x_k is the current iterate and $\Delta_k \in \mathbb{R}_+$ is the mesh size parameter. We note that the mesh is not explicitly constructed but is rather a conceptual entity.

SEARCH STEP

A finite subset of mesh points, possibly none, are selected. These points are evaluated to find an improving point. If any of these points improves the current iterate, then x_k is replaced by the improving point. However if this search fails to find an improving point, the next step i.e. the POLL step is invoked. Any strategy such as a heuristic rule may be used to select these candidate mesh points. Consequently, due to the lack of mathematical foundation the SEARCH step does not contribute to the convergence properties of the PS method and is considered by some researchers to be a liability [7, 8]. Most implementations of the PS algorithm do not use this step.

POLL STEP

The POLL step consists of evaluating the function on the set of mesh points neighboring the current iterate x_k. These neighboring points are referred to as the poll set and denoted as follows:

$$P_k = \{x_k + \Delta_k d_i : d_i \in D, i = 1, \ldots, p\}\,. \tag{5.3}$$

Each point in the POLL step is evaluated until an improved mesh point is found. If this step is successful, the iterate is updated to the new improved mesh point.

MESH UPDATE

At each iteration, the SEARCH or POLL steps will either give an improved mesh point or

both will fail. This presents two possible end scenarios. If an iteration fails one can conclude that the current point is locally optimal for the current mesh. Hence the mesh is refined using the following rule:

$$\Delta_{k+1} = \theta_k \Delta_k \tag{5.4}$$

with $0 < \theta_k < 1$. If however the algorithm succeeds in finding an improved mesh point, the mesh is either kept the same or increased via the following rule:

$$\Delta_{k+1} = \theta_k \Delta_k \tag{5.5}$$

with $\theta_k > 1$.

Typical values for the mesh parameter update are $\Delta_{k+1} = \frac{1}{2}\Delta_k$ for when the mesh needs to be refined and $\Delta_{k+1} = 2\Delta_k$ when the mesh needs to be coarsened [8]. Both these processes are implicit. The PS algorithm based on the POLL step is given below.

Algorithm 3 The PS algorithm

1. Set parameters Δ_0, counter $k = 0$, stopping tolerance $\Delta_{tol} > 0$ and x_0, where x_0 is an initial solution
2. **POLL STEP** : Evaluate objective function f at trial points in poll set, $x_k^i = (x_k + \Delta_k d_i)$.
   - **IF** a point say x_k^i in the poll set is found such that $f(x_k^i) \leq f(x_k)$ **THEN**
     * $x_{k+1} = x_k^i$
     * Either increase the mesh size parameter Δ_k or keep it the same using (5.5) and then go to step 3.
   - **IF** $f(x_k^i) \geq f(x_k)$ for all $x_k^i \in P_k$ **THEN**
     * $x_{k+1} = x_k$
     * Decrease the mesh size parameter Δ_k using (5.4) and then go to step 3.
3. **IF** $\Delta_k < \Delta_{tol}$ **THEN** STOP, ELSE $k = k + 1$ and go to step 2.

For illustrative purposes we present a hypothetical example using only the POLL step in Figure 5.1. The current iterate is indicated by a shaded circle, an unsuccessful trial point is indicated by an unshaded circle and a successful trial point is given by a semi shaded circle. We present the trial points in open brackets e.g. $x_1 = (x_1^1, x_1^2)$ and its corresponding function

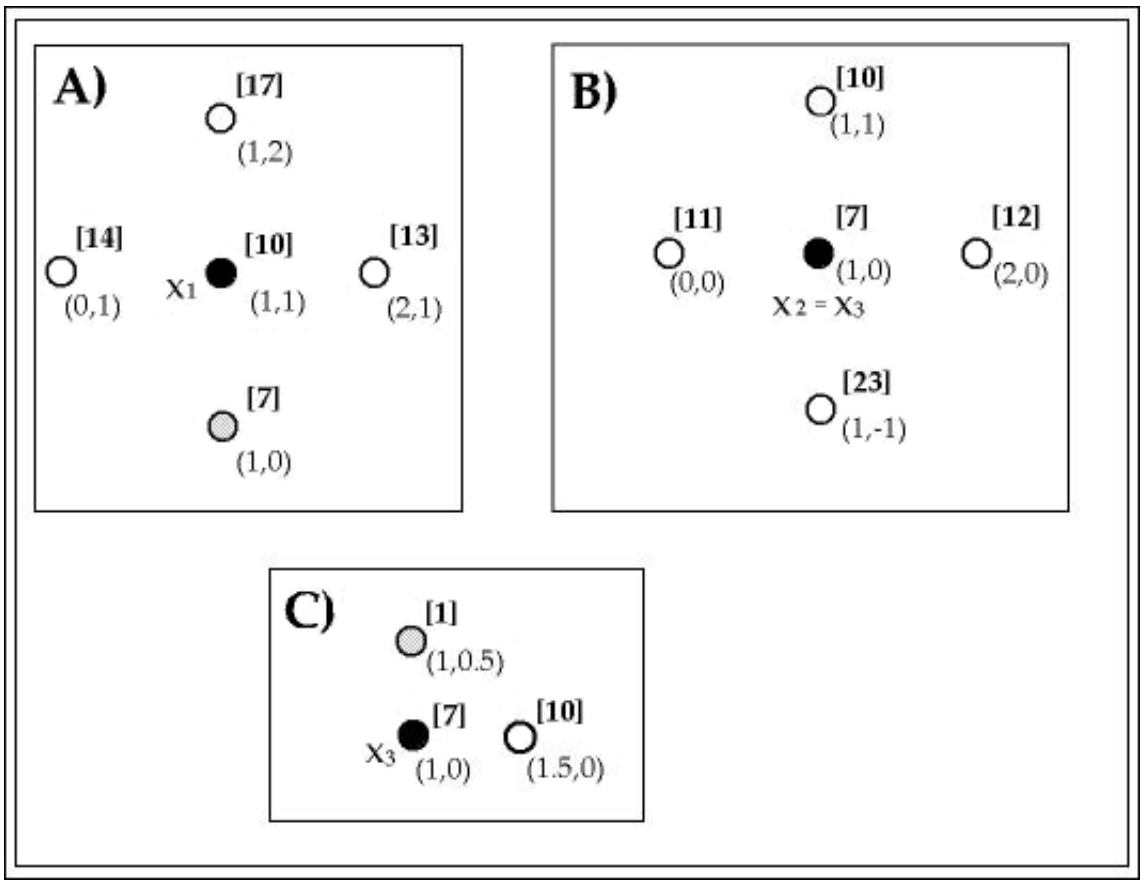

Figure 5.1: *Example of PS*

value in square brackets e.g. $[f(x_1)]$.

In Figure 5.1 A) $x_1 = (1, 1)$ is the current iterate with a function value of 10 and we let $\Delta_1 = 1$. If we poll around x_1 using the spanning matrix $D = \{e_1, e_2, -e_1, -e_2\}$ our first trial point will be the point $x_1 + \Delta_1 \times e_1 = (1, 1) + (1, 0) = (2, 1)$, where the function value is 13. This trial point will therefore not provide an improvement and we proceed to the next trial point, $(1, 2)$. Similarly we are unsuccessful at this trial point, $(1, 2)$, as well as (0,1) where the function values are 17 and 14 respectively. The last trial point $(1, 0)$ however has the a function value of 7 which is lower than that of x_1. Therefore we let $x_2 = (1, 0)$ be our new iterate and poll center. The poll step is successful and so the mesh is kept the same. The order in which the trial points are generated does not matter.

In Figure 5.1 B) once again we poll around x_2 however none of the trial points provides a decrease in the objective function value. Hence for the next iteration the poll center is kept the same and the mesh is refined with $\Delta_3 = \frac{1}{2}$.

Figure 5.1 C) shows how the process is repeated again with $x_3 = (1, 0)$ as our current iterate and $\Delta_3 = \frac{1}{2}$. The second trial point $(1, 0.5)$ provides an improvement and will be set as the new iterate x_4. The POLL process is restarted using $x_4 = (1, 0.5)$ as the new iterate. This process will continue until $\Delta_k < \Delta_{tol}$.

5.2 PS for constrained optimization

The applicability of the PS method to constrained local optimization problems is fairly limited. This is because in order to uphold the convergence guarantees of the PS method, it can only be used on a specific set of problems. Audet and Dennis [8] have proposed a PS filter method that does not require any derivatives. We will provide a brief overview of this method. For the purpose of their algorithm the definitions for a mesh and poll center were extended. The method uses an initial set of solutions, say S_0. The definition of a mesh is thus extending to include the mesh for each initial solution. Hence at any iteration k:

$$M(S_k, \Delta_k) = \bigcup_{x \in S_k} M(x, \Delta_k) \tag{5.6}$$

where $M(x, \Delta_k)$ is defined as in equation (5.2). This allows the SEARCH step to select mesh points around any of the trials points, $x \in S_k$. A filter set is created using the initial set of solutions.

The poll center is chosen from either the feasible or infeasible incumbents. The resulting poll set needs to include the selected poll center $\{p_k\}$ and is thus defined as:

$$P_k = \{p_k\} \cup \{x_k + \Delta_k d_i : d \in D, i = 1, \ldots, p\} \tag{5.7}$$

For this algorithm, the purpose of the exploration is not just to find a decrease in the objective function value but to find unfiltered mesh points. Therefore the SEARCH and POLL steps are considered successful if an unfiltered mesh point is found.

The algorithm generates and evaluates an initial set, S_0, of points. The filter and incumbents are extracted from this set. The SEARCH and POLL steps are carried out until an unfiltered mesh point is found or it is shown that all mesh points are filtered. The filter set, mesh size parameter, Δ_k and trial point are updated accordingly. The process is then repeated. Algorithm 4 fully describes this process. For more details and insight the reader is referred to [8].

Algorithm 4 A pattern search filter algorithm

Initialization: Let x_0 be an undominated point from the initial set of solutions. Include all initial solutions in the filter $\mathcal{F}$ and set $G_{max} > G(x_0)$. Fix $\Delta_0 > 0$ and set $k = 0$.

Definition of Incumbents: Define (if possible) the following:

- f_k^F: Feasible incumbent i.e. smallest feasible objective function found
- G_k^I: Least positive constraint violation found thus far
- f_k^I: Infeasible incumbent i.e. the smallest objective function value for points found thus far whose constraint violation is equal to G_k^I

SEARCH and POLL Steps: Perform SEARCH and possible POLL step until an unfiltered trial point x_{k+1} is found or until it is found that all trial points are filtered.

- SEARCH: Evaluate G and f on a set of trial points on a mesh M_k.
- POLL: Evaluate G and f on the poll set P_k, where $p_k \in P_k$ satisfies either $((G(p_k), f(p_k)) = (0, f_k^F)$ or $((G(p_k), f(p_k)) = ((G_k^I, f_k^I))$

Parameter Update: If the SEARCH or POLL step resulted in an unfiltered mesh point $x_{k+1} \in \mathcal{F}_{\ k+1}$ then declare the iteration successful and and update $\Delta_{k+1} \geq \Delta_k$. Otherwise set $x_{k+1} = x_k$, declare the iteration unsuccessful and set $\Delta_{k+1} < \Delta_k$. Increment $k = k + 1$ and go back to definition of incumbents.

Unlike in the PS filter algorithm discussed above our aim is not to use the PS method as our underlying optimizer. We will rely on DE as a primary exploration tool since it is a global solver. The PS method is ideal for local exploration. We want to exploit this characteristic of the PS by incorporating it into the DE algorithm as a local search mechanism. This will allow us to have a hybrid global method with additional local search properties.

If we look at the mutation process of the DE algorithm, we see that the trial point is a perturbation of one of the parents in a single direction. This direction is not necessarily the descent direction. However the PS will enable us to search in multiple directions until an

improving point is found. With PS, our objective is not to do a complete local search but to simply explore the immediate neighborhood of a particular point. We will incorporate the PS method into two of our proposed approaches. The first will be the DE algorithm for constrained global optimization together with PS for local exploration. The second will be the DE algorithm with PS and filter set combined. We will fully discuss these methods in Chapter 6.

Chapter 6

Differential evolution algorithms for constrained global optimization

The aim of this chapter is to provide a description of algorithms that are proposed for constrained global optimization. Our research objective is to find a constrained global optimization algorithm that can successfully be applied to a wide range of problems without imposing any prerequisites on the problems. We wish to use the DE algorithm as an underlying global optimizer. In this chapter we present four different algorithms based on the DE algorithm and other techniques.

Our first goal would be to provide a modified DE algorithm that can be used for constrained global optimization. We will refer to it as the DE algorithm for constrained global optimization and denote it by DEC. The second algorithm is a filter based DEC algorithm which will be abbreviated as FDEC. It is established on the DEC algorithm but includes a filter set that is used to create a diversification mechanism in the search process. The third algorithm is a PS based DEC or PSDEC. This algorithm is also structured on the DEC algorithm but has an additional localization strategy that is founded on the PS method. The last algorithm presented includes both the diversification process provided by the filter set as well as the local search mechanism provided by the PS method. It is denoted by PSFDEC. Detailed descriptions of all the algorithms are given below.

6.1 Differential evolution for constrained global optimization (DEC)

The DE algorithm for unconstrained optimization has been fully discussed in Chapter 2. The elementary pseudo-code for the DE algorithm has also been presented in Chapter 2 along with a full description of all the steps involved. Here, we briefly summarize the DE algorithm: Firstly the population is initialized as described by equation (2.1). Then a trial population is generated via the process of mutation and crossover. Any of the mutation schemes, such as the examples given in equations (2.2) to (2.4) and the crossover scheme represented by (2.5) can be used. Then the two populations are compared point to point and a new population is formed depending on the acceptance rule as given in (2.6). This process of mutation, crossover and acceptance is repeated until some stopping condition is met.

We now present the changes made in going from the unconstrained to the constrained version of the DE algorithm. Firstly, changes are made to the parameter values of DE. Instead of the fixed scaling parameter in the original algorithm we will use an iteration based scaling parameter F for the mutation process. Since F will be different for each generation we use the notation F_g, where $F_g \in [-1, -0.4] \cup [0.4, 1]$. At each generation a random number $r_g \in (0, 1)$ will be generated. If $r_g \leq 0.5$ then F_g is drawn uniformly from $[0.4, 1]$ else F_g is drawn uniformly from $[-1, -0.4]$.

In order to adapt the DE algorithm for constrained optimization, there are certain changes that need to be made to the algorithm. Since we are using penalty functions as our constraint handling mechanism, the fundamental structure of the DE algorithm remains mostly unchanged. The main differences between DE and DEC center around the evaluation of the fitness function and the acceptance rule.

Secondly, the DEC algorithm has to accommodate the inclusion of the constraint violation. For each member, $x_{i,g}$, in the population two additional values namely the constraint violation $G(x_{i,g})$ and the resulting fitness function value $\hat{f}(x_{i,g})$ for each point need to be stored.

In general, to calculate the fitness value for the i^{th} member of the population, the objective function values $f(x_{i,g})$ and constraint violation $G(x_{i,g})$ for all points in the current population, S_g, needs to be known first. This is because firstly the term Θ_g (see (3.8) and (3.12)) in the penalty function is dependent on the feasibility of the point being evaluated and whether the population has only infeasible points or a combination of both infeasible and feasible points. Secondly the worst feasible and (or) best infeasible point in the population is needed in order to determine the penalty function. We briefly describe how Θ_g and $\hat{f}$ are evaluated using both the SFP and PFP below.

For the SFP scheme the fitness function and constraints are calculated as given by equations (3.7) to (3.9) in Chapter 3. The additional penalty term Θ_g is added to infeasible points if the population has both infeasible and feasible points. Also Θ_g is calculated based on the worst feasible point and best infeasible point. The reader is referred to Chapter 3 for a more detailed look at this scheme.

Equations (3.11) and (3.12) describe the calculation Θ_g and $\hat{f}$ using the PFP scheme. Unlike SFP, here the additional penalty term Θ_g applies to all infeasible points irrespective of the makeup of the population. The worst feasible point in the population is used to calculate the value of Θ_g. For further reading please refer to Chapter 3.

All the above mentioned issues related to the evaluation of $\hat{f}$ becomes relevant from a computational perspective as they will affect the implementation of the DEC algorithm. Any point in the population can only be fully evaluated (i.e. fitness value calculated) after all points have been generated and their relevant objective function value $f(x)$ and constraint violation value $G(x)$ have been determined. Then only can the value for Θ_g in the penalty function for the SFP and PFP schemes be calculated. The DEC algorithm needs to facilitate this.

Finally, the last difference is in the acceptance rule. For the unconstrained optimization trial points are accepted based on the objective function value, $f(x)$, whereas for constrained optimization the acceptance rule is structured on the fitness value, $\hat{f}(x)$. Therefore, the new acceptance rule can be given mathematically as:

$$x_{i,g+1} = \begin{cases} y_{i,g} & if\ \hat{f}(y_{i,g}) < \hat{f}(x_{i,g}), \\ x_{i,g} & otherwise. \end{cases} \tag{6.1}$$

The DEC algorithm is summarized in the next few lines. Firstly the population $S = \{x_{1,0}, x_{2,0}, \ldots, x_{N,0}\}$ is initialized as described by equation (2.1). Then the objective function $f(x_{i,0})$ and constraint $G(x_{i,g})$ for each point are obtained. The penalty Θ_0 is evaluated and the fitness values $\hat{f}(x_{i,g})$ for each point is determined. Next a trial population is generated via the process of mutation and crossover. Any of the mutation schemes such as those given in equations (2.2) to (2.4) and the binomial crossover scheme represented by (2.5) can be used. The trial population is evaluated to obtain the objective function value $f(y_{i,1})$ and constraint value $G(y_{i,1})$. Θ_1 is obtained and the fitness value $\hat{f}(y_{i,1})$ for all trial points is finally calculated. Then the two populations are compared point to point and a new population is formed depending on the acceptance rule as given in (6.1). This process of mutation, crossover, evaluation and acceptance is repeated until some stopping criterion is met. We now present the DEC algorithm below:

Algorithm 5 The DEC algorithm

1. Set control parameters N, c_r and $g = 0$ as well as penalty parameter R for the SFP method.
2. Initialize population $S_0 = \{x_{1,0}, x_{2,0}, \ldots, x_{N,0}\}$ uniformly using (2.1)
3. Evaluate objective function value and constraint violation of each member in the population
4. Determine penalty function Θ_0 and evaluate fitness of initial population using (3.7)-(3.9) for SFP method or (3.11) and (3.12) for PFP method
5. **IF** Stopping Criteria not met
   (a) Generate $F_g \in [-1, -0.4] \cup [0.4, 1]$ uniformly
   (b) **FOR** $i = 1$ **TO** N,
      i. generate trial point $y_{i,g}$ via:
         - **Mutation** using (2.2), (2.3) or (2.4)
         - **Crossover** using (2.5)
      ii. Evaluate $f(y_{i,g})$ and $G(y_{i,g})$
6. Determine penalty function Θ_g and evaluate fitness of trial population using (3.7)-(3.9) for SFP method or (3.11) and (3.12) for PFP method
7. Update population using (6.1)
8. Set $g = g + 1$, calculate stopping criteria and go to 5.

6.1.1 Possible pitfalls of the acceptance rule in DEC

An important aspect to consider when dealing with the DEC algorithm is that during the acceptance phase two different populations are being compared. The value of the additional term Θ_g in the penalty function is totally dependent on the composition of the population. We will discuss this further.

If we consider the case of unconstrained optimization we compare the objective function values of two points, which is straight forward. However for constrained optimization we are comparing the fitness function values of two points from two different populations. This is significant since the additional penalty term in the fitness function is determined by the composition of the population. We will provide an example below to describe this.

If we were to consider two populations e.g. the current population and the trial population at the g^{th} generation. For the sake of explanation, let us consider that the current population contains only infeasible points with the constraint violation in the range of say $0 < G(x) < 10$. While the trial population contains feasible and infeasible points and objective function values of the feasible points are large positive numbers. Table 6.1 and 6.2 present examples of these two populations. We use Problem 1 from Appendix A and the PFP penalty scheme to evaluate the populations. If we were to compare these two populations the acceptance rule, as given by (6.1), will select the infeasible points with small $\hat{f}(x)$ values as given in Table 6.1 over the feasible points with large $\hat{f}(x)$ values as given by x_3 and x_4 in Table 6.2. Also the infeasible points in the trial population will carry an additional penalty while those in the current population will not carry any added penalty since all points are infeasible. So even though the actual constraint violation may be less the resulting fitness value will be higher e.g. x_1 and x_2 in Table 6.2 as compared to x_1 and x_2 in Table 6.1. This can be seen with the infeasible points in the current population where the fitness value is simply the constraint violation and are therefore small values. Whereas in the trial population the infeasible points carry an additional penalty so even though both infeasible points in the trial population have smaller violations their actual fitness value is higher.

i	x_i	$f(x_i)$	$G(x_i)$	$\hat{f}(x_i)$
1	(1422.3 ,4301.4, 7781.5,319.3,361.97, 248.43, 612.76, 624.18)	13505	2.68	2.68
2	(6815.8 , 8448.7, 8654.4 , 133.88 , 234.31 ,7 645.76 , 886.06 , 781.17)	23919	6.8839	6.8839
3	(4216.2 , 2042.3 , 7348.6 , 886.02 , 369.85 , 384.46 , 934.54 , 830.31)	13607	5.8267	5.8267
4	(5227.7 , 2225 , 9401.5 , 302.82, 156.38, 581.87, 617.91, 869.35)	16854	7.5201	7.5201

Table 6.1: Example: current population with infeasible points only

i	x_i	$f(x_i)$	$G(x_i)$	$\hat{f}(x_i)$
1	(6518.4 , 6405.3 , 8989.4, 27.369 , 483.64 , 590.42, 265.17 , 609.15)	1.6032	1.6032	18056.6032
2	(5244.4 , 3398.3 , 4072.5 , 281.06 , 252.03 , 331.99 , 553.75 , 459.47)	1.9188	1.9188	18056.9188
3	(4768.1 , 4482.6 , 8784.6 , 209.68 , 266.59 , 148.32 , 317.27, 353.27)	18035	0	18035
4	(5875.5 , 3822.4 , 8357, 173.29 , 253.29 , 130.09 , 312.43 , 342.13)	18055	0	18055

Table 6.2: Example: trial population with feasible and infeasible points

This scenario is rare and usually occurs when using the PFP method with problems that have positive function values for feasible points. An easy way to overcome this, as we have chosen to do, is to always select feasible points over infeasible points irrespective of the fitness values. Next we look at the DEC implementation utilizing the filter method.

6.2 A filter based DE method for constrained optimization (FDEC)

Our approach for the filter based DE is to maintain the underlying structure of the DEC algorithm but include the filter method as an additional means to explore the infeasible search space. Since we are using penalty schemes as our constraint handling mechanism, the DEC algorithm will in most instances be biased toward feasible points. This is because as soon as a feasible point enters the population, the search will be directed toward that particular feasible region. This could compromise the full exploration of the infeasible search space and possible discovery of additional feasible regions. When using penalty schemes to handle constraints the schemes focus of minimizing the the constraint violation only. The function values are often disregarding or de-emphasized. By using the filter we want to explore a wider spectrum of points with both high and low objective functions values. This is the main

motivation of the FDEC algorithm.

We now use two figures to motivate our use of the filter set. Figures 6.1 and 6.2 give an example of the evolution of population set of DEC and FDEC respectively. The algorithms were applied to problem 20 in Appendix A. Using a population size of $N = 10$ we ran both algorithms for 10 generations. Each figure was drawn using the data from a single run. The points in the figure were taken from the 1^{st}, 5^{th} and 10^{th} generations, i.e. S_1, S_5 and S_{10}. From Figure 6.1 and 6.2 we can see that when compared to DEC, the population set of FDEC maintains a greater diversity amongst its points.

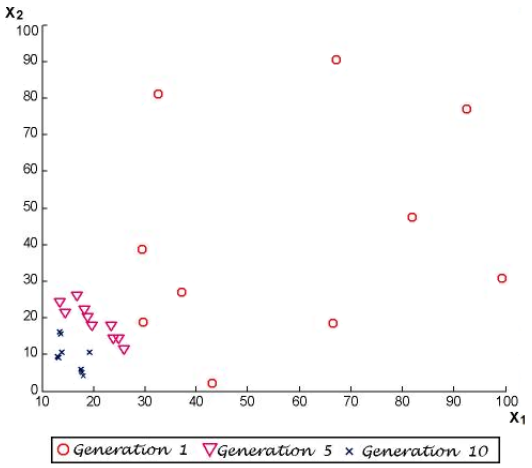

Figure 6.1: *Distribution of points in* $S_g, g = 1, 5, 10$*, DEC*

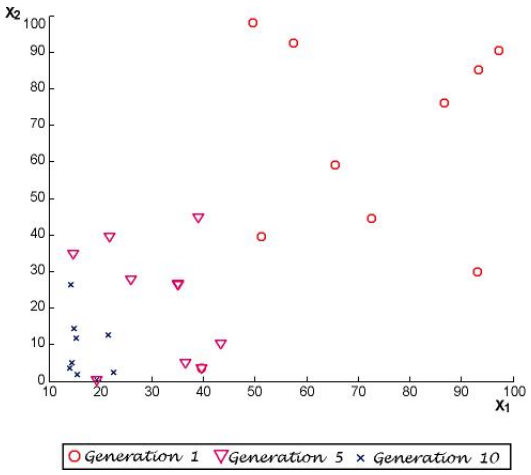

Figure 6.2: *Distribution of points in* $S_g, g = 1, 5, 10$*, FDEC*

Having motivated the use of the filter set, we now present the differences between DEC and FDEC. There are two principal differences between the FDEC algorithm and the DEC algorithm. Firstly the FDEC algorithm will keep an updated filter set. The filter set is basically a list of pairs $[f(x), G(x)]$ where no pair dominates another. The reader is referred to Chapter 4 for full details on the filter set. All new trial points generated during the search will be checked against the current filter set. If a new point is found to dominate any point(s) in the filter it will be added to the filter and the dominated point(s) will be removed.

The second difference is in the mutation scheme. If the target point is infeasible then the filter mutation scheme given by:

$$\hat{x}_{i,g} = x_{undominated} + F_g \times (x_{p(2)} - x_{p(3)}) \tag{6.2}$$

is used to generate a trial point where $F_g \in [-1, -0.4] \cup [0.4, 1]$. When the target point is feasible, the mutated point is calculated as in DEC. Note that (4.3) and (6.2) are equivalent except that the scaling parameter F_g has been randomized in (6.2). The point $x_{undominated}$ is randomly selected from the filter set for each mutation point. This mutation scheme will allow for any regions represented by undominated points encountered during the search to be better explored. Our aim is to explore different regions and locate better infeasible points and possibly also feasible points. If the target point is feasible the penalty function will ensure that any infeasible trial point generated will be discarded. If the target point is feasible then the preselected mutation schemes such as those given by (2.2), (2.3) or (2.4) are used.

The FDEC algorithm will now be described. Once the initial population is generated and evaluated as described for the DEC algorithm, all nondominated pairs $[f(x_{i,0}), G(x_{i,0})]$ will then be used to create a filter $\mathcal{F}$. At the g^{th} generation if the target point $x_{i,g}$ is infeasible a point $x_{undominated}$ is randomly selected from $\mathcal{F}$. Using $x_{undominated}$ as the base point, $\hat{x}_{i,g}$ is generated using the filter mutation scheme described by (6.2). If the target point is feasible the mutation phase is carried as in DEC. The trial point $y_{i,g}$ is then obtained once the crossover is carried out using (2.5). The function value and constraint violation of $y_{i,g}$ is then calculated and the pair $[f(y_{i,g}), G(y_{i,g})]$ is checked for dominance against the current filter $\mathcal{F}$.

If $y_{i,g}$ is dominated by any pair in $\mathcal{F}$, then we proceed to the calculation of the next trial point $y_{i+1,g}$. However if $y_{i,g}$ is undominated, it is added to $\mathcal{F}$ and all pairs that it dominates are removed from $\mathcal{F}$. It is important to note that the objective function value and not the fitness value is used in the filter set. The fitness value for all trial points, $y_{i,g}, i = 1, 2, \ldots, N$, are then calculated once the penalty functions G and Θ_g are evaluated. Then the two populations are compared point to point and a new population is formed depending on the acceptance rule as given in (6.1). This process of mutation, crossover, evaluation and acceptance is repeated until some stopping criteria is met. The FDEC algorithm is given in algorithm 4.

Algorithm 6 The FDEC algorithm

1. Set control parameters N, c_r and $g = 0$ as well as penalty parameter R for SFP method.
2. Initialize population $S_0 = \{x_{1,0}, x_{2,0}, \ldots, x_{N,0}\}$ uniformly using (2.1)
3. Evaluate objective function value and constraint violation of each member in the population
4. Determine penalty function Θ_0 and evaluate fitness of initial population using (3.7)-(3.9) for SFP method or (3.11) and (3.12) for PFP method
5. Generate filter set using initial population
6. **WHILE** Stopping Criteria not met
   (a) Generate $F_g \in [-1, -0.4] \cup [0.4, 1]$ uniformly
   (b) **FOR** $i = 1$ **TO** N,
      i. generate trial point $y_{i,g}$:
         - **IF** $x_{i,g}$ is infeasible then use filter mutation scheme (6.2) **ELSE** use mutation scheme (2.2),(2.3) or (2.4)
         - **Crossover** using (2.5)
      ii. Evaluate $f(y_{i,g})$ and $G(y_{i,g})$
      iii. Evaluate $y_{i,g}$ against filter and update filter set as necessary
7. Determine penalty function Θ_g and evaluate fitness of trial population using (3.7)-(3.9) for SFP method or (3.11) and (3.12) for PFP method
8. Update population using (6.1)
9. Set $g = g + 1$, calculate stopping criteria and go to 6.

6.2.1 Implementational issues

In most of the previous use of the filter set, the filter is simply used to check for dominance [10, 25]. However the FDEC algorithm uses the actual points related to the filter pairs. These points are used for the filter mutation scheme as given in (6.2). Therefore it becomes necessary to store these points separately. Simply indexing these pairs to the actual population is not possible since occasionally points that are in the filter are not present in the current population. This could become an important factor to consider for problems of very high dimensions where the filter could be of a significant computational size. Also since the FDEC algorithm is based on the DEC algorithm all other implementational issues that were relevant for the DEC algorithm apply to it as well.

6.3 A PS based DE method for constrained global optimization (PSDEC)

The aim of the PS based DEC method is to provide a local search mechanism within the scope of the DEC algorithm. This hybrid will have the global properties of the DEC algorithm while the PS method will allow for quicker convergence to local minima.

6.3.1 The PS based local exploration algorithm

We first look at the PS method that will be used for the PSDEC algorithm. Our aim is not to carry out a full local search but just a simple limited local exploration. Hence we will not implement the full PS method but a limited version of it. We denote this method by PPS or partial pattern search. A detailed description of the general PS algorithm is presented in Chapter 5 and by algorithm 3. Here we briefly describe the method: A single point x_k referred to as the current iterate is evaluated, k being the iteration number. Then a set of points that lie on a mesh neighboring the current iterate are considered. These points are referred to as the poll set. The mesh size is dependent on a parameter Δ_k. If an improving point is found in the poll set then the mesh is either coarsened or kept the same. If no improving point is found the current iterate is considered to be locally optimal for that mesh and hence the mesh is refined. The parameter Δ_k is used to either coarsen or refine the mesh. Equations (5.2) to (5.5) give the mathematically formulation for the mesh, poll set and mesh update.

What we propose for the PPS method is a non-iterative process using a single iterate. Hence the PPS method uses a single value for Δ to produce one poll set around the current iterate. If an improving point is found the algorithm is stopped and considered successful. A point will be considered to be improved if a lower fitness function value is found. If none of the points in the poll set improve the current poll center is considered to be a minimum for the current poll set and the PPS algorithm is ended. This will allow us to have a partial local

search within our DEC algorithm. We will fully describe the PPS algorithm below.

Firstly five points are selected randomly from the best $\alpha\%$ of the population[1]. One of the five points is randomly selected to be the current iterate x_k and the poll centre. The most important feature of PPS is the calculation of Δ. The average distance ($AvgDis$) between the five points that have been selected is calculated. Δ will be set as a fraction of the $AvgDis$ i.e. $\Delta = \beta \times AvgDis$, where $\beta < 1$ [2]. Since atleast one of the directions in the poll set is a descent direction, we set Δ as a small value in order to find a better point than x_k. The value for Δ will vary depending on the distance between the points. At early stages the population will be diverse and this will result in larger values for Δ. As the population converges the distance between points will decrease and so will the value for Δ. Clearly, Δ will no longer be a user defined parameter and will adapt with the population. Since the algorithm is to be tested on a wide range of problems whose search areas vary dramatically in size, using a fixed value such as $\Delta = 1$ will results in poor performance. The PPS algorithm is presented below by algorithm 5.

Algorithm 7 The PPS algorithm

1. Randomly select five point within best 10 % of the population.
2. Calculate the average distance, $AvgDis$, between the five points and set $\Delta = 0.1 \times AvgDis$
3. Uniformly select one point say $x_{r,g}$ amongst the five points be the current iterate $x_k = x_{r,g}$
4. POLL STEP: Evaluate objective function value at trial points in poll set, $x_k^i = (x_k + \Delta_k d_i)$.
   - **IF** a point say x_k^i in the poll set is found such that $\hat{f}(x_k^i) \leq \hat{f}(x_k)$ **THEN**
     - $x_{k+1} = x_k^i$
     - STOP
   - **IF** $\hat{f}(x_k^i) \geq \hat{f}(x_k)$ for all $x_k^i \in P_k$ **THEN**
     - $x_{k+1} = x_k$
     - STOP

[1] For our problem set this percentage was determined empirically. We tested a range of values between 5% and 15% and found 10% to be most optimal.

[2] Empirical testing showed that $\beta = 0.1$ is a good choice for the problem sets.

6.3.2 The PSDEC algorithm

To implement the PPS method in DEC, the underlying DEC algorithm need not change. The reader is referred to section 6.1 for a full description of the DEC algorithm. The only change would be to call PPS after the population is updated at the end of an iteration. One of the drawbacks of the PS method is that its efficiency is extremely dependent on the dimension of the problem. Higher dimensions imply larger poll sets and thus a larger number of function evaluations. In order to limit the number of function evaluations for the PPS method we only invoke the method after every 10 iterations. The PPS method will be carried out using a random point selected as the poll center. The poll center is therefore a point within the best 10% of the current population. If the PPS method finds an improving point it will replace the original point with the improved point in the population, otherwise the population remains unchanged.

The most noteworthy feature of this algorithm is that it does not require any additional parameters. The only parameters are those of the DEC algorithm, while the PPS algorithm is self contained and requires no user defined input. The PSDEC algorithm is presented below:

Algorithm 8 The PSDEC algorithm

1. Set control parameters N, c_r and $g = 0$ as well as penalty parameter R for the SFP method.
2. Initialize population $S_0 = \{x_{1,0}, x_{2,0}, \ldots, x_{N,0}\}$ uniformly using (2.1)
3. Evaluate objective function value and constraint violation of each member in the population
4. Determine penalty function Θ_0 and evaluate fitness of initial population using (3.7)-(3.9) for SFP method or (3.11) and (3.12) for PFP method
5. **IF** Stopping Criteria not met
   (a) Generate $F_g \in [-1, -0.4] \cup [0.4, 1]$ uniformly
   (b) **FOR** $i = 1$ **TO** N,
      i. generate trial point $y_{i,g}$ via:
         - **Mutation** using (2.2),(2.3) or (2.4)
         - **Crossover** using (2.5)
      ii. Evaluate $f(y_{i,g})$ and $\sum_{j=1}^{m} \langle g_j(y_{i,g}) \rangle$
6. Determine penalty function Θ_g and evaluate fitness of trial population using (3.7)-(3.9) for SFP method or (3.11) and (3.12) for PFP method
7. Update population using (6.1)
8. If (g mod 10 = 0), then call PPS and update population if required
9. Set $g = g + 1$, calculate stopping criteria and go to 5.

6.3.3 Implementational issues

The only implementational issue to consider for this method is that the PPS method requires additional fitness function evaluations. These must be considered and they should contribute to the total number of fitness function evaluations for the algorithm. The implementational issues of the DEC algorithm are also applicable for the PSDEC algorithm.

6.4 A PS filter-based DE method for constrained global optimization (PSFDEC)

The PSFDEC algorithm is in essence a combination of the previous two algorithms. The objective is to create a hybrid that will include both the local search aspect given by the PPS algorithm as well as the diversification mechanism from the filter mutation scheme.

The PSFDEC algorithm will be based on the FDEC algorithm presented in section 6.2. This will ensure that the filter will be maintained and the filter mutation scheme will be used. The reader is referred back for full details of the FDEC algorithm. The only change will be that the PPS method will be invoked just as in the PSDEC algorithm. Hence at the end of every 10^{th} iteration after the population is updated the PPS algorithm will be called.

The PSFDEC algorithm will now be described. Once the initial population is generated and evaluated as described for the DEC algorithm, all nondominated pairs $[f(x_{i,0}), G(x_{i,0})]$ will then be used to create a filter $\mathcal{F}$. At the g^{th} iteration if the target point $x_{i,g}$ is infeasible a point $x_{undominated}$ is randomly selected from $\mathcal{F}$. Using this point, $\hat{x}_{i,g}$ is generated by the filter mutation scheme described by (6.2). If the target point is feasible the mutation phase is carried as normal i.e. using (2.2), (2.3) or (2.4). The trial point $y_{i,g}$ is then obtained once binomial crossover is carried out using (2.5). The function value and constraint violation of $y_{i,g}$ is then calculated and the pair, $[f(x_{i,g}), G(x_{i,g})]$, is checked for dominance against the current filter $\mathcal{F}$. If $y_{i,g}$ is dominated by any pair in $\mathcal{F}$, then we proceed to the calculation of

the next trial point $y_{i+1,g}$. However if $y_{i,g}$ is undominated, it is added to $\mathcal{F}$ and all pairs that it dominates are removed from $\mathcal{F}$. The fitness value for all trial points, $y_{i,g}, i = 1, 2, \ldots, N$, are then calculated once the penalty functions G and Θ_g are evaluated. Then the two populations are compared point to point and a new population is formed depending on the acceptance rule as given in (6.1). If the current iteration g is a multiple of 10 then the PPS method is called. If the PPS method finds an improved point in the poll set, then the improved point will replace the current iterate in the population, otherwise the population remains unchanged. This process of mutation, crossover, evaluation, acceptance and PPS is repeated until some stopping criteria is met.The algorithm is presented below:

Algorithm 9 The PSFDEC algorithm

---

1. Set control parameters N, c_r and $g = 0$ as well as penalty parameter R for SFP method.
2. Initialize population $S_0 = \{x_{1,0}, x_{2,0}, \ldots, x_{N,0}\}$ uniformly using (2.1)
3. Evaluate objective function value and constraint violation of each member in the population
4. Determine penalty function Θ_0 and evaluate fitness of initial population using (3.7)-(3.9) for SFP method or (3.11) and (3.12) for PFP method
5. Generate filter set using initial population
6. **WHILE** Stopping Criteria not met
   (a) Generate $F_g \in [-1, -0.4] \cup [0.4, 1]$ uniformly
   (b) **FOR** $i = 1$ **TO** N,
      i. generate trial point $y_{i,g}$:
         - **IF** $x_{i,g}$ is infeasible then use filter mutation scheme (6.2) **ELSE** use mutation scheme (2.2),(2.3) or (2.4)
         - **Crossover** using (2.5)
      ii. Evaluate $f(y_{i,g})$ and $G(y_{i,g})$
      iii. Evaluate $y_{i,g}$ against filter and update filter set as necessary
7. Determine penalty function Θ_g and evaluate fitness of trial population using (3.7)-(3.9) for SFP method or (3.11) and (3.12) for PFP method
8. Update population using (6.1)
9. If (g mod 10 = 0) then call PPS method and update population if required
10. Set $g = g + 1$, calculate stopping criteria and go to 6.

6.4.1 Implementational issues

The implementational issues for this algorithm are simply a combination of all the issues that have been previously mentioned for the DEC, FDEC and PSDEC.

Chapter 7

Numerical results

In order to evaluate the performance of DEC, FDEC, PSDEC and PSFDEC introduced in the previous Chapter, we have carried out extensive numerical testing on two sets of test problems. This Chapter presents these results together with comparisons and analysis. The first set of test problems was taken from [20, 28, 36, 37]. We will compare the performance of all the proposed algorithms with GA presented in [38].

The first set, henceforth is referred to as set A, consists of 33 test problems and is summarized in Table 7.1. The table contains all the attributes mentioned below. We denote the problem number by np as has been done in [38]. The dimension, given by n, of each problem is included. The objective function is classified as either linear (lin), nonlinear (nonl) or quadratic (quad). The number of linear equality (LE), linear inequality (LI), nonlinear equality (NE) and nonlinear inequality (NI) constraints are also indicated. The value ρ, expressed as a percentage, is the ratio of the feasible region to the given box constrained area as has been given in [38]. The problem references as well as the best known minimum value for each problem are given. This set of problems contains 21 nonlinear problems, 11 quadratic and a single linear problem. Of the 21 nonlinear problems only 3 contain trigonometric objective functions or constraint functions. A full description of the problem set A is given in Appendix A.

The second set, referred to as set B, of 12 test problems is presented in Table 7.2. It includes problems that are also commonly used to test the performance of constrained global optimization algorithms. Just as with problem set A all the relevant details regarding each problem including their references are presented in Table 7.2. The problems in set B are numbered from 34 to 45. We calculated the value for ρ by generating 1 million random points in the box-constrained area of a given problem. The overall percentage of feasible points generated will determine the value for ρ. We have included the values for ρ in Table 7.2. Of the 12 problems 10 are nonlinear with 4 containing trigonometric objective functions and/or constraints. The problems are fully described in Appendix B.

The algorithms were implemented in Matlab and were tested on a Pentium 4, 2.8 GHz computer. For each problem, 100 independent test runs were carried out. For all the algorithms a run was stopped if a preset stopping condition was met. If a run resulted in any feasible points in the population then the best fitness function value found was recorded and the run was considered successful. We refer to this as feasibility success. The best values from the successful runs were then used to summarize the results for each problem. The following values were extracted for comparison:

mean: The mean value of the best solutions found for all feasible runs of a problem.
min: The best fitness value found amongst all the feasible runs of a problem.
dev: The standard deviation of all the best values found from all feasible runs of a problem.
feas: The percentage of runs that produced feasible points for a problem.
iter: The average number of iterations, where the average is taken over all runs that achieved feasibility success for a problem.
feval: The average number of fitness function evaluations, where the average is taken over all runs that achieved feasibility success for a problem.

The above values will be used to represent the results based on 100 independent runs on each problem. It is also important to note that these results are based on the feasibility success of each problem.

We also present the summarized results of an algorithm on an entire set of test problems. However these summarized results are obtained on problems that are solved at least once out of the 100 runs. Hence we define a second set of criteria to provide a measure of the overall performance of an algorithm on an entire problem set. The following values will be used as the criteria for comparison:

SR: The success rate (SR) will indicate the number of problems for which the algorithm succeeded in finding the best known minimum function value for a problem. For example in problem set A, SR = 33 will imply that an algorithm successfully located the global minimum for each problem at least once in its 100 independent runs i.e. all 33 problems were solved by the algorithm.

TFeas: The total feasible (TFeas) value will give the total number of runs resulting in feasible solutions. This will be used as an indication of the reliability of the algorithm. The TFeas value is a reflection of the feasibility success of an algorithm. In problem set A there are 33 problems and each problem had 100 independent runs. Hence if all runs for all 33 problem resulted in feasible points then TFeas = 3300.

AvgIter: The efficiency will be measured by the average number of iterations (AvgIter) calculated using the iter value for all problems for which the algorithm was successful in locating the best known minimum function value, i.e. the iter average taken over all successful problems.

AvgFe: Another measure of efficiency will be the average number of fitness function evaluations (AvgFe) calculated using the feval value for all problems for which the algorithm was successful in locating the best known function value, i.e. the feval average taken over all successful problems.

AvgSD: We also give the average of the standard deviation (dev) values for problems where the best known function value was successfully located. This value will give a measure as to how accurately the algorithm is able to locate the best known solution. If the AvgSD is high, this will show that the best solutions found for a problem varied widely

over the feasible runs. Conversely, a low AvgSD indicates that the best solutions found had very similar fitness values for each feasible successful run.

The reason there are two measures for efficiency, namely AvgIter and AvgFe, is because the algorithms that have the PPS algorithm incorporated into them will have additional fitness function evaluations for those iterations where the PPS algorithm is invoked. Therefore the AvgIter will not provide an accurate measure of the efficiency. For GA, only the average iteration values have been given in [38]. However it is simple to extract the average number of fitness function evaluations from the average number of iterations. If the population size is N then the AvgFe value is given by:

$$AvgFe = N + AvgIter \times N \tag{7.1}$$

This equation applies to GA, DEC and FDEC.

PARAMETER SELECTION

The following parameters were common amongst all four new algorithms. The population was fixed at $N = 101$. The algorithms were stopped if either one of two stopping criteria were met. The first was a maximum of 500 iterations for each run. The second stopping criteria was set such that if the difference between the best individuals for the last 100 iterations was less than 0.01, the algorithm was stopped. Binomial crossover (2.5) was used for all algorithms. For the SFP scheme a penalty coefficient of $R = 10000$ was used. Equality constraints were transformed into inequality constraints with a tolerance value of 0.01 i.e. $\delta = 0.01$ in (3.3) and (3.4).

The number of independent runs, population size, penalty coefficient, equality constraint transformation tolerance and stopping criteria are the same as were used for the numerical tests by Miettinen et al [38]. By doing this it will allow us to make a fair comparison between GA and the DE based algorithms without creating a bias toward either algorithm. Any other parameters that are specific to a particular algorithm will be discussed in the relevant section.

If an algorithm was successful in finding the best known solution we indicated this by printed the minimum value in bold. However if the problem contained equality constraints the results are often better than the best known solution. This can be attributed to the tolerance value, δ, introduced during the transformation to inequality constraints. These minimum values are therefore underlined. If the algorithm failed in finding any feasible results for all 100 runs of that problem this was indicated by a '-'.

Using these settings we tested all four algorithms with the SFP and PFP constraint handling schemes. When referring to a particular algorithm implemented with one of the schemes we denote it by appending the scheme to the name of the algorithm e.g. DEC-SFP implies the DEC algorithm implemented with the SFP constraint handling scheme. The results for each algorithm are given in separate subsections. The first set of results presented will be those of GA in the next section.

np	n	Obj Func	ρ	LE	LI	NE	NI	Ref.	Best Known
1	8	lin	.000578	0	6	0	0	[20]	7049.25
2	5	quad	26.960078	0	0	0	6	[20]	-30665.5387
3	6	quad	11.312849	0	4	0	2	[20]	-310.0
4	4	nonl	.043394	1	2	0	0	[20]	-4.5142
5	4	nonl	.013552	1	2	0	0	[20]	-2.07
6	6	nonl	.000000	3	3	0	0	[20]	-11.96
7	2	lin	44.200537	0	0	0	2	[20]	-5.5079
8	2	quad	.332226	0	0	1	0	[20]	-16.68
9	4	nonl	.00000	0	2	3	0	[37]	5126.4981
10	50	nonl	100.0	0	0	0	2	[37]	-0.8331937
11	5	nonl	.00001	0	0	3	0	[28]	0.0539498
12	2	nonl	24.99898	0	2	0	0	[36]	-1.0
13	2	nonl	.861168	0	0	0	2	[37]	-0.095825
14	23	nonl	.00000	0	0	1	0	[37]	-1.0
15	10	nonl	.00000	3	0	0	0	[36]	-47.760765
16	2	nonl	7.32900	0	0	0	2	[28]	0.25
17	2	quad	96.644521	0	0	0	2	[28]	5.0
18	7	nonl	.524944	0	0	0	4	[37]	680.6300573
19	13	quad	.00244	0	9	0	0	[37]	-15.0
20	2	nonl	.006711	0	0	0	2	[37]	-6961.81381
21	10	quad	.000110	0	3	0	5	[37]	24.3062091
22	2	quad	37.492715	0	1	0	1	[28]	1.0
23	5	quad	95.256165	0	1	0	0	[20]	-17.0
24	6	quad	23.404995	0	2	0	0	[20]	-213.0
25	13	quad	.237391	0	9	0	0	[20]	-15
26	6	quad	1.827590	0	5	0	0	[20]	-11.005
27	10	quad	.004728	0	11	0	0	[20]	-268.01
28	10	quad	.007350	0	5	0	0	[20]	-39.0
29	20	quad	.00000	0	10	0	0	[20]	-394.7506
30	20	quad	.00000	0	10	0	0	[20]	-884.75058
31	20	quad	.00000	0	10	0	0	[20]	-8695.01193
32	30	nonl	99.999947	0	0	0	2	[37]	-0.8331937
33	70	nonl	100.0	0	0	0	2	[37]	-0.8331937

Table 7.1: Test Problems - Set A

np	n	Obj Func	ρ	LE	LI	NE	NI	Ref.	Best Known
34	6	nonl	0.14	0	0	0	2	[50]	-316.27
35	2	nonl	27.55	0	1	0	2	[50]	0.18
36	2	nonl	18.05	0	1	0	1	[50]	0
37	2	quad	37.23	0	1	0	1	[50]	-195.37
38	2	nonl	73.04	0	2	0	0	[50]	-2.21
39	2	quad	49.93	0	1	0	0	[50]	0.125
40	3	nonl	0.23	0	0	0	5	[11]	0.6164
41	2	nonl	0.63	0	0	1	0	[11]	0.0821
42	2	nonl	62.73	0	0	0	2	[11]	1.5087
43	6	nonl	2.75	0	4	0	2	[11]	0.7593
44	11	nonl	0.00	2	4	0	0	[11]	8827.5977
45	6	nonl	0.00	0	0	4	1	[11]	-0.388811

Table 7.2: Test Problems - Set B

7.1 Genetic algorithm

This section will present and summarize the results for GA as have been given in [38]. The implementation of GA requires a number of associated parameter values to be provided. Miettinen et al [38] used the following parameter values: $crossover\ \ rate\ =\ 0.8$, $elitism\ \ size\ =\ 1$, $tournament\ \ size\ =\ 3$, $mutation\ \ rate\ =\ 0.1$ and $p\ =\ 4$, where p is the mutation exponent. We consider the results for the SFP and PFP constraint handling schemes. The results are given in Table 7.3.

	S F P					**P F P**				
np	mean	min	dev	iter	fea	mean	min	dev	iter	fea
1	7893.74	7116.64	1285.08	416.5	68	8464.55	7292.1	1294.91	101.0	100
2	-30665.53	**-30665.54**	0.06	296.3	100	-30665.53	**-30665.54**	0.01	296.3	100
3	-309.84	**-310.0**	1.6	217.6	100	-308.58	**-310.0**	12.69	224.5	100
4	-4.52	-4.53	0.15	145.8	100	-4.41	-4.53	0.4	148.4	100
5	-3.13	-3.14	0.02	128.9	100	-3.14	-3.14	0.0	129.7	100
6	-13.32	-13.41	0.26	298.8	100	-13.38	-13.41	0.15	300	100
7	-5.51	**-5.51**	0.0	110.6	100	-5.51	**-5.51**	0.0	110.9	100
8	-16.78	**-16.78**	0.0	115.5	100	-16.78	**-16.78**	0.0	116.5	100
9	4239.21	4221.83	62.14	404.6	100	4755.32	4221.83	531.2	102.7	100
10	-0.56	-0.64	0.03	394.7	100	-0.56	-0.64	0.03	394.7	100
11	0.38	0.05	0.29	304.5	100	0.58	0.05	0.35	108.2	100
12	-1.0	**-1.0**	0.0	102.8	100	-1.0	**-1.0**	0.0	102.7	100
13	-0.1	**-0.10**	0.01	106.4	100	-0.1	**-0.1**	0.0	106.7	100
14	-	-	-	-	-	-0.78	-1.01	0.09	461.4	100
15	-47.01	-48.11	0.77	467.2	100	-47.03	-47.97	0.77	446.7	100
16	0.25	**0.25**	0.00	106	100	0.25	**0.25**	0.0	106.0	100
17	5.0	**5.0**	0.0	110.5	100	5.0	**5.0**	0.0	110.0	100
18	681.56	680.81	0.0	143.7	100	682.75	680.75	2.17	225.3	100
19	-14.94	**-15.0**	0.34	323.9	100	-14.98	**-15.0**	0.2	327.1	100
20	-6961.81	**-6961.81**	0.0	143.7	100	-6961.81	**-6961.81**	0.0	143.4	100
21	26.87	24.77	1.37	459.4	100	32.63	25.76	5.64	107.6	100
22	1.0	**1.0**	0.0	106.7	100	1.0	**1.0**	0.0	106.6	100
23	15.98	**-17.00**	1.16	154.4	100	-15.81	**-17.0**	1.54	155.1	100
24	-212.98	**-213.0**	0.08	213.3	100	-212.98	**-213.0**	0.09	216.6	100
25	-15.00	**-15.0**	0.0	313.0	100	-15.0	**-15.0**	0.0	313.4	100
26	-11.00	**-11.0**	0.0	204.3	100	-10.99	**-11.0**	0.05	212.7	100
27	-265.81	**-268.01**	3.21	469.2	100	-265.06	-268.00	3.42	468.6	100
28	-36.66	**-39.0**	5.19	259.4	100	-37.05	**-39.0**	4.73	251.2	100
29	-135.08	-221.11	38.63	494.3	100	-132.02	-247.72	40.54	498.2	100
30	-593.81	-696.76	34.34	463.6	100	-586.45	-698.08	29.27	458.1	100
31	-3043.37	-5374.88	682.9	486.6	100	-3106.12	-5424.69	695.63	490.1	100
32	-0.66	-0.74	0.04	350.1	100	-0.66	-0.74	0.04	350.1	100
33	-0.5	-0.57	0.03	408.3	100	-0.5	-0.57	0.03	408.3	100

Table 7.3: Results for GA on set A

If we look at Table 7.3 we see that both schemes managed to successfully locate the best known minimum for 23 out of the 33 problems. GA-PFP successful located feasible points for all runs of all problems, whereas GA-SFP failed completely on problem 14 and for some runs of problem 1. For those problems that the methods where successful in locating the known minimum the standard deviation was fairly low except for problem 9. We also note that for some of the problems (i.e. problems 7, 8, 12, 16-18, 20, 22 and 25) the mean and min values are equal and the standard dev is 0. We can conclude that for these problems the best known solution was found for all 100 runs.

We now summarize the results for GA in Table 7.4 using the second set of evaluation criteria described at the beginning of this Chapter. We also include the AvgFe as calculated using (7.1).

	SR	**TFeas**	**AvgIter**	**AvgFe**	**AvgSD**
GA-SFP	23	3168	221.89	22 511.9	3.27
GA-PFP	23	3300	199.86	20 286.9	24.01

Table 7.4: Summary of results for GA on set A

From the summary we can see that both algorithms failed to find the best known solution for 10 out of the 33 test problems giving SR=23. GA-PFP method has a greater feasibility success with 132 more feasible runs. This is partially due to the fact that GA-SFP method failed completely on all runs of problem 14. In terms of computational cost GA-SFP required more iterations and had a small average standard deviation amongst the successful problems. This indicates that the accuracy of the method is fairly good but with an added computational cost. GA-PFP method was less accurate but also required fewer iterations.

Next we present the results of GA on test set B. For this we have implemented GA with the same parameters as those used in [38] i.e.
$crossover\ rate = 0.8, elitism\ size = 1, tournament\ size = 3, mutation\ rate = 0.1$ and $p = 4$, where p is the mutation exponent. We tested GA on problem set B using both the SFP and PFP schemes. We have presented the results in Table 7.5.

	S F P					P F P				
np	mean	min	dev	feas	iter	mean	min	dev	feas	iter
34	-295.43	-314.03	13.41	100	313.1	-295.65	-314.39	12.12	100	319.3
35	0.18	**0.18**	0.00	100	101.3	0.18	**0.18**	0.00	100	101.3
36	0	**0**	0.00	100	101.8	0	**0**	0.00	100	101.6
37	-195.37	**-195.37**	0.01	100	123	-195.37	**-195.37**	0.00	100	121.6
38	-2.21	**-2.21**	0.00	100	100.9	-2.21	**-2.21**	0.00	100	101
39	0.13	**0.13**	0.00	100	101.1	0.13	**0.13**	0.00	100	101.1
40	0.64	**0.62**	0.02	100	170.5	0.63	**0.62**	0.02	100	173.3
41	0.08	**0.08**	0.00	100	103.86	0.08	**0.08**	0.00	100	104.75
42	1.51	**1.51**	0.00	100	102.97	1.51	**1.51**	0.00	100	103.10
43	0.97	0.79	0.13	100	216.9	0.97	0.79	0.11	100	201
44	-	-	-	-	-	-	-	-	-	-
45	-0.41	-0.41	0.01	100	134.3	-0.41	-0.41	0	100	136

Table 7.5: Results for GA on set B

Table 7.5 shows that both GA-SFP and GA-PFP located the best known solution for 9 out of the 12 problems. Both schemes failed to find any feasible points for problem 44 but located feasible points for all runs of all other problems. The standard deviation for all problems except for problem 34 is very low. The min and mean values, for many of the successful problems, are equal and the standard deviation is 0. This shows that the best known minimum was located for all 100 runs for each problem. We have summarized the results in Table 7.6.

	SR	**TFeas**	**AvgIter**	**AvgFe**	**AvgSD**
GA-SFP	9	1100	115.53	11 769.53	0.0
GA-PFP	9	1100	115.97	11 813.97	0.0

Table 7.6: Summary of results for GA on set B

Table 7.6 shows that SR and TFeas values for both schemes are the same. Both schemes failed on two problems i.e. on problems 34 and 44, neither scheme was able to find any feasible points for problem 44. The AvgSD for both methods was zero however GA-PFP had a slightly higher AvgIter value. This indicates that GA-SFP performed slightly better than GA-PFP.

In the sections that follow we present the results for the four algorithms introduced in Chapter 6 on both problem sets. We will provide a comparisons of these algorithms with GA

[38] using both sets of test problems.

7.2 Results for the DEC algorithm

In this section we present the results for the DEC algorithm. We firstly study the effects of two different mutation schemes on the DEC algorithm. We set the crossover parameter c_r equal to 0.5 and implement the mutation schemes given by equations (2.2) and (2.3). Equation (2.2) gives a mutation scheme where the base vector is randomly selected while for (2.3) the base vector is the point with the lowest fitness value in the current population. These two settings can be denoted by rand/1/bin and best/1/bin respectively. The results for the DEC-SFP algorithm on test set A are presented in Table 7.7.

	rand/1/bin					**best/1/bin**				
np	mean	min	dev	feas	iter	mean	min	dev	feas	iter
1	8741.89	7467.11	1011.20	100	366.86	7298.59	7114.99	130.47	100	455.20
2	-30665.39	-30665.48	0.05	100	500.00	-30665.53	**-30665.54**	0.01	100	458.64
3	-310.00	**-310.00**	0.00	100	386.89	-310.00	**-310.00**	0.00	100	261.44
4	-4.53	-4.53	0.00	100	324.43	-4.53	-4.53	0.00	100	217.68
5	-3.13	-3.14	0.00	100	238.60	-3.14	-3.14	0.00	100	170.15
6	-10.05	-12.45	1.72	88	363.45	-13.10	-13.40	0.41	100	432.12
7	-5.51	**-5.51**	0.00	100	131.31	-5.51	**-5.51**	0.00	100	116.85
8	-16.78	**-16.78**	0.00	100	144.04	-16.78	**-16.78**	0.00	100	112.52
9	-	-	-	-	-	5225.38	5126.48	151.80	100	481.03
10	-0.25	-0.32	0.02	100	218.55	-0.54	-0.71	0.11	100	454.52
11	0.94	0.59	0.12	11	448.73	0.53	0.10	0.33	99	415.28
12	-1.00	**-1.00**	0.00	100	104.91	-1.00	**-1.00**	0.00	100	102.51
13	-0.10	**-0.10**	0.00	100	109.19	-0.10	**-0.10**	0.00	100	105.55
14	-	-	-	-	-	-	-	-	-	-
15	-42.75	-45.36	1.51	39	278.69	-44.50	-47.41	1.47	100	361.34
16	0.25	**0.25**	0.00	100	117.15	0.25	**0.25**	0.00	100	109.04
17	5.00	**5.00**	0.00	100	130.55	5.00	**5.00**	0.00	100	116.26
18	680.74	680.66	0.07	100	462.37	680.65	**680.63**	0.02	100	350.88
19	-14.90	-14.94	0.02	100	500.00	-14.99	**-15.00**	0.00	100	364.48
20	-6961.81	**-6961.81**	0.00	100	267.73	-6961.81	**-6961.81**	0.00	100	186.07
21	25.74	25.07	0.38	100	481.64	24.62	24.39	0.19	100	467.06
22	1.00	**1.00**	0.00	100	123.84	1.00	**1.00**	0.00	100	111.85
23	-16.85	**-17.00**	0.60	100	405.79	-16.38	**-17.00**	0.57	100	197.39
24	-213.00	**-213.00**	0.00	100	354.80	-213.00	**-213.00**	0.00	100	244.50
25	-14.92	-14.96	0.02	100	500.00	-14.99	**-15.00**	0.00	100	354.08
26	-10.99	**-11.00**	0.00	100	323.02	-11.00	**-11.00**	0.00	100	207.43
27	-262.52	-265.17	1.71	100	495.90	-267.17	-267.69	0.42	100	494.96
28	-38.99	**-39.00**	0.00	100	491.77	-38.82	**-39.00**	0.72	100	310.19
29	-27.38	-47.88	7.11	88	500.00	-163.50	-275.05	40.88	100	490.08
30	-513.46	-537.94	9.65	89	500.00	-604.20	-701.19	35.57	100	476.57
31	-1323.32	-1744.87	164.46	79	499.97	-3419.02	-4949.56	626.20	100	481.26
32	-0.40	-0.47	0.04	100	342.12	-0.76	-0.82	0.04	100	416.94
33	-0.20	-0.23	0.01	100	152.90	-0.33	-0.56	0.10	100	315.83

Table 7.7: Results for DEC-SFP on set A, $c_r = 0.5$

Looking at Table 7.7 we see that firstly both settings successful located the best known minimum for 15 common problems and the setting best/1/bin was successful on an additional 6 problems. Both the settings failed to locate any feasible points for problem 14 and the setting rand/1/bin failed to locate feasible points for problem 9 as well. We also note that the setting rand/1/bin had far more runs where it failed to locate any feasible solutions than the setting best/1/bin. For those problems where both the settings were successful in locating the known minimum the standard deviation was fairly low except the setting rand/1/bin on problem 9. We also note that for problems 10, 29, 30 and 32 the minimum values found by the setting rand/1/bin were even better than those found by GA in Table 7.3. However the minimum value found by GA for problem 31 and 33 are slightly better. Aside from problem 14, these problems have the highest dimensions in the problem set. We summarize these results in Table 7.8.

	SR	**TFeas**	**AvgIter**	**AvgFe**	**AvgSD**
rand/1/bin	15	2894	243.6	24 704.6	0.04
best/1/bin	21	3199	238.6	24 199.6	7.31

Table 7.8: Summary of results for DEC-SFP on Set A, $c_r = 0.5$

Table 7.8 shows that the setting best/1/bin performed better than the setting rand/1/bin on all aspects except the standard deviation. The rand/1/bin setting was successful in locating the minimum for only 15 of the problems while the setting best/1/bin was successful on 21 problems. What is notable about the setting rand/1/bin is that the AvgIter value is slighter higher than the setting best/1/bin however the AvgSD value is very small. We can conclude that for those problems where the setting rand/1/bin successfully locates the best known minimum value, the best values found for each run vary fairly little from the best known minimum. With regard to the total number of feasible runs the setting best/1/bin leads the setting rand/1/bin. This can be partially attributed to problem 9 and 14 where the setting rand/1/bin failed to locate any feasible solutions. For some other problems such as 6, 11 and 15 the setting rand/1/bin had far fewer feasible runs. All these factors indicate that the setting best/1/bin is better than the setting rand/1/bin. Hence the rest of our numerical results are based on the implementation of best/1/bin.

For constrained global optimization we would like to study the effects of c_r on the performance of the DEC algorithm, as to the best of our knowledge, this has not been done before. For unconstrained optimization $c_r = 0.5$ is recommended for most problems and is the best value for using on a large set of test problems [2, 6]. However for constrained optimization we have found some interesting results. We carried out a number of test runs on problem set A using various values such as $c_r = 0.7$, 0.9 and 0.95 . The DEC algorithm produced superior results with $c_r = 0.9$ for both penalty schemes. We present the results in Table 7.9 for best/1/bin with $c_r = 0.9$ using both the SFP and PFP constraint handling schemes.

	S F P					P F P				
np	mean	min	dev	feas	iter	mean	min	dev	feas	iter
1	7049.29	**7049.25**	0.06	100	472.82	7049.39	**7049.25**	0.82	100	484.96
2	-30665.54	**-30665.54**	0.00	100	245.87	-30665.54	**-30665.54**	0.00	100	250.75
3	-310.00	**-310.00**	0.00	100	230.61	-310.00	**-310.00**	0.00	100	233.33
4	-4.53	-4.53	0.00	100	145.80	-4.53	-4.53	0.00	100	156.30
5	-3.14	-3.14	0.00	100	137.47	-3.14	-3.14	0.00	100	149.50
6	-13.41	-13.41	0.00	100	176.47	-13.41	-13.41	0.00	100	190.77
7	-5.51	**-5.51**	0.00	100	113.05	-5.51	**-5.51**	0.00	100	113.82
8	-16.78	**-16.78**	0.00	100	110.62	-16.78	**-16.78**	0.00	100	117.53
9	5210.17	5126.48	142.27	100	388.35	5201.06	5126.48	138.03	100	397.56
10	-0.38	-0.49	0.04	100	201.97	-0.38	-0.48	0.05	100	194.17
11	0.33	0.05	0.23	100	186.56	0.34	0.05	0.24	100	188.01
12	-1.00	**-1.00**	0.00	100	102.43	-1.00	**-1.00**	0.00	100	102.31
13	-0.10	**-0.10**	0.00	100	104.94	-0.10	**-0.10**	0.00	100	104.91
14	-	-	-	-	-	-1.11	-1.12	0.00	100	240.59
15	-48.07	-48.14	0.24	100	296.08	-48.07	-48.14	0.21	100	336.41
16	0.25	**0.25**	0.00	100	107.56	0.25	**0.25**	0.00	100	108.13
17	5.00	**5.00**	0.00	100	113.97	5.00	**5.00**	0.00	100	113.24
18	680.63	**680.63**	0.00	100	181.07	680.63	**680.63**	0.00	100	189.19
19	-14.64	**-15.00**	0.59	100	309.70	-14.48	**-15.00**	0.73	100	315.46
20	-6961.81	**-6961.81**	0.00	100	161.44	-6961.81	**-6961.81**	0.00	100	193.27
21	24.31	**24.31**	0.01	100	281.93	24.31	**24.31**	0.01	100	297.10
22	1.00	**1.00**	0.00	100	109.96	1.00	**1.00**	0.00	100	109.57
23	-16.04	**-17.00**	1.21	100	193.53	-16.03	**-17.00**	1.29	100	191.41
24	-213.00	**-213.00**	0.00	100	192.45	-213.00	**-213.00**	0.00	100	193.36
25	-14.70	**-15.00**	0.54	100	300.64	-14.60	**-15.00**	0.63	100	297.71
26	-11.00	**-11.00**	0.00	100	169.93	-11.00	**-11.00**	0.00	100	168.35
27	-268.01	**-268.01**	0.03	100	374.81	-268.01	**-268.01**	0.00	100	396.37
28	-36.94	**-39.00**	4.94	100	304.24	-36.70	**-39.00**	5.92	100	316.92
29	-216.62	-383.13	75.40	100	500.00	-204.69	-364.69	60.62	100	500.00
30	-688.27	-869.20	74.32	100	500.00	-678.76	-841.80	69.48	100	500.00
31	-5228.99	-8534.11	1519.82	100	500.00	-5021.31	-8101.83	1454.09	100	494.06
32	-0.46	-0.70	0.07	100	194.53	-0.46	-0.70	0.07	100	193.27
33	-0.34	-0.43	0.04	100	190.87	-0.33	-0.41	0.06	100	187.59

Table 7.9: Results for DEC on set A: $c_r = 0.9$, setting best/1/bin.

Table 7.9 shows that DEC-SFP and DEC-PFP located the known global minimum for 26 common problems. The DEC-PFP method also located the minimum for problem 14 but the DEC-SFP method failed to locate any feasible points for this problem. For both methods the standard deviation is fairly low for all problems that were successful except problem 9. We also note that for many of the problems (i.e. problems 2-8, 12,13, 16-18, 20-22, 24, 26 and 27) the mean and min values are equal and the standard dev is 0. We can conclude that for these problems the best known solution was found for all 100 runs. With regard to feasibility, DEC-SFP had all feasible runs except for problem 14 while DEC-PFP managed to find feasible points for all runs on all problems. For problems 29 and 30 we note that the average number of iterations for both methods are equal to 500. The same is true for DEC-SFP on problem 31. This is important since we set a maximum of 500 iterations as one of our stopping criterion. This indicates that for all runs on these problems the methods did not get trapped in any local minima but the search process was stopped due to the preset stopping criterion.

Using the second set of evaluation criteria, as discussed previously, we now summarize the results for Table 7.9 in Table 7.10. We have included the summarized results in Table 7.10 for the best/1/bin setting with $c_r = 0.5$, as given in Table 7.8, as well as the results for GA as summarized in Table 7.4 in the previous section for comparison.

	SR	**TFeas**	**AvgIter**	**AvgFe**	**AvgSD**
DEC-SFP $c_r = 0.5$	21	3199	238.6	24 199.6	7.31
DEC-SFP $c_r = 0.9$	26	3200	212.01	21 514.0	5.77
DEC-PFP $c_r = 0.9$	27	3300	220.6	22 381.6	5.48
GA-SFP	23	3168	221.89	22 511.9	3.27
GA-PFP	23	3300	199.86	20 286.9	24.01

Table 7.10: Summary of results for DEC on set A

To see the effect of c_r in DEC we firstly compare DEC-SFP for $c_r = 0.5$ and $c_r = 0.9$. Table 7.10 shows that the overall performance of the DEC-SFP algorithm has improved with a higher crossover rate i.e. $c_r = 0.9$. For instance, the total number of problems for which the best known solutions were found increased from 21 for $c_r = 0.5$ to 26 for $c_r = 0.9$. We also

note that the average number of iterations as well as the average standard deviation is lower for the DEC-SFP algorithm using $c_r = 0.9$. There is also one additional feasible run for the DEC-SFP algorithm with $c_r = 0.9$. All these factors indicate that the accuracy, efficiency, reliability and success rate of the DEC method is much better with the higher crossover rate.

Next we compare DEC-SFP and DEC-PFP for $c_r = 0.9$ using the summarized results in Table 7.10. Table 7.10 shows firstly that the DEC-PFP has a higher success rate than DEC-SFP. Secondly since DEC-SFP fails to find any feasible points for problem 14 its total number of feasible runs is less than that for the DEC-PFP method. Lastly, the DEC-PFP method does prove to be more expensive in terms of the average number of iterations. On the other hand, the average standard deviation on successful problems is also slightly lower for DEC-PFP.

From the above discussion it is clear that the DEC algorithm produced superior results using $c_r = 0.9$. Therefore, next we compare our results of the DEC algorithm where $c_r = 0.9$ with those of GA. Firstly, Table 7.10 shows that both DEC-SFP and DEC-PFP attained a higher SR than GA. The DEC-SFP algorithm was successful in locating the global minimum for 26 problems out of a total of 33 problems. On the other hand, GA failed on a total of 10 problems. This difference in the success rate is very significant as it shows the most important improvement of DEC over GA.

A look at the TFeas values in Table 7.10 reveals that, DEC-SFP has more feasible runs than GA-SFP. Both DEC-PFP and GA-PFP found feasible points for all 100 runs for all 33 problems. This shows that for both DEC and GA the PFP scheme is more reliable than the SFP scheme. Furthermore GA-SFP proved to be the least reliable with respect to TFeas.

Finally, we compare DEC and GA with respect to accuracy (AvgSD) and efficiency (AvgIter). Table 7.10 shows that GA-SFP is the best performer with respect to AvgSD and DEC-PFP is the runner-up with 3.27 and 5.48 respectively. A similar comparison using AvgIter shows that GA-PFP is the best performer followed by DEC-SFP. For all these methods the accuracy seems to share an inversely proportional relationship with the efficiency of the method. For instance, GA-SFP method which has the smallest standard deviation is also

the most expensive whereas GA-PFP method is the least accurate requires the fewest number of average iterations.

Next we present the results for the DEC algorithm using the setting best/1/bin and $c_r = 0.9$ for the second set of test problems. These results are presented in Table 7.11.

	S F P					**P F P**				
np	mean	min	dev	feas	iter	mean	min	dev	feas	iter
34	-307.80	**-316.27**	20.26	100	251.51	-309.49	**-316.27**	18.44	100	266.08
35	0.18	**0.18**	0.00	100	101.49	0.18	**0.18**	0.00	100	101.46
36	0.00	**0.00**	0.00	100	102.66	0.00	**0.00**	0.00	100	102.51
37	-195.37	**-195.37**	0.00	100	129.31	-195.37	**-195.37**	0.00	100	129.57
38	-2.21	**-2.21**	0.00	100	100.83	-2.21	**-2.21**	0.00	100	100.85
39	0.13	**0.13**	0.00	100	101.16	0.13	**0.13**	0.00	100	101.40
40	0.62	**0.62**	0.00	100	136.99	0.62	**0.62**	0.00	100	135.93
41	0.08	**0.08**	0.00	100	107.46	0.08	**0.08**	0.00	100	109.06
42	1.51	**1.51**	0.00	100	104.15	1.51	**1.51**	0.00	100	104.33
43	0.76	**0.76**	0.00	100	163.80	0.76	**0.76**	0.00	100	164.00
44	8900.70	8840.31	42.30	100	500.00	8909.57	8828.20	46.09	100	498.70
45	-0.41	-0.42	0.00	100	124.60	-0.41	-0.42	0.00	100	125.80

Table 7.11: Results for DEC on set B

Table 7.11 shows that both schemes were successful in locating the global minimum for all problems except number 44. DEC-SFP and DEC-PFP show good reliability with all runs resulting in feasible points. For the successful problems the standard deviation is low for all problems except problem 1. In fact for problems 35 to 43 and problem 45 the standard deviation is 0 and the min and mean values are equal. This indicates that the best known minimum was found in all runs for these problems. We summarize the results for set B in Table 7.12 where we have also included the summarized results of GA from Table 7.6.

	SR	**TFeas**	**AvgIter**	**AvgFe**	**AvgSD**
DEC-SFP	11	1200	129.45	13 175.45	1.84
DEC-PFP	11	1200	131.0	13 332.0	1.68
GA-SFP	9	1100	115.53	11 769.53	0.0
GA-PFP	9	1100	115.97	11 813.97	0.0

Table 7.12: Summary of results for DEC on set B

The summarized results in Table 7.12 show that when compared to GA the DEC algorithm has a higher success rate. DEC-SFP, with a lower AvgIter value, has a better effi-

ciency when compared to DEC-PFP but with respect to AvgSD, DEC-SFP is slightly superior. When compared to GA both DEC-SFP and DEC-PFP have a higher AvgIter and AvgSD value. This is because the results of DEC are based on 11 problems while those of GA are based on 9 problems only. We therefore compare DEC and GA on the 9 common problems i.e. 35-42 and 45, and present the summarized results in Table 7.13.

	SR	**TFeas**	**AvgIter**	**AvgFe**	**AvgSD**
DEC-SFP	9	900	112.07	11 420.07	0.0
DEC-PFP	9	900	112.32	11 445.32	0.0
GA-SFP	9	900	115.53	11 769.53	0.0
GA-PFP	9	900	115.97	11 813.97	0.0

Table 7.13: Summary of results for DEC for 9 common problems on set B

Table 7.13 shows that DEC and GA performed equally well on the 9 problems with respect to SR, TFeas and AvgSD. However, DEC is superior to GA with respect to AvgIter. This shows that for the 9 common problems the performance of DEC is better than GA.

To sum up, DEC is superior to GA with respect to SR on problem set A and B (see Tables 7.10 and 7.12). The TFeas values for GA-PFP and DEC-PFP are equal for set A but DEC-SFP was superior to GA-SFP. DEC performed better on feasibility on set B. DEC is comparable to GA-SFP with regard to AvgSD on both A and B. However DEC is superior to GA-PFP with respect to AvgSD. For problem set A, the AvgIter values for DEC-SFP and DEC-PFP are comparable to GA-SFP but are slightly inferior to GA-PFP. For problem set B the AvgIter values for DEC are superior than GA (see Tables 7.13). It is therefore clear that DEC performs better than GA on most criteria.

7.3 Results for the FDEC algorithm

This section contains the results for the FDEC algorithm using both the SFP and PFP penalty schemes. We use the best parameter values as found in the previous section e.g. $c_r = 0.9$ and the setting best/1/bin. The FDEC algorithm does not require any additional parameters when compared to the DEC algorithm. The results for the algorithm are given in Table 7.14.

	S F P					**P F P**				
np	mean	min	dev	feas	iter	mean	min	dev	feas	iter
1	7049.33	**7049.25**	0.30	100	485.77	7049.38	**7049.25**	0.49	100	484.65
2	-30665.54	**-30665.54**	0.00	100	250.09	-30665.54	**-30665.54**	0.00	100	248.14
3	-310.00	**-310.00**	0.00	100	234.55	-310.00	**-310.00**	0.00	100	236.67
4	-4.53	-4.53	0.00	100	147.14	-4.52	-4.53	0.15	100	151.49
5	-3.14	-3.14	0.00	100	142.09	-3.14	-3.14	0.00	100	145.93
6	-13.41	-13.41	0.00	100	178.99	-13.40	-13.41	0.09	100	190.05
7	-5.51	**-5.51**	0.00	100	113.26	-5.51	**-5.51**	0.00	100	114.12
8	-16.78	**-16.78**	0.00	100	110.56	-16.78	**-16.78**	0.00	100	115.78
9	5143.65	5126.48	39.50	100	390.23	5191.18	5126.48	107.53	100	396.73
10	-0.38	-0.46	0.04	100	199.46	-0.38	-0.45	0.03	100	200.36
11	0.36	0.05	0.24	100	181.56	0.33	0.05	0.22	100	193.03
12	-1.00	**-1.00**	0.00	100	102.71	-1.00	**-1.00**	0.00	100	102.26
13	-0.10	**-0.10**	0.00	100	104.81	-0.10	**-0.10**	0.00	100	104.81
14	-	-	-	-	-	-1.11	-1.12	0.00	100	241.35
15	-48.08	-48.14	0.17	100	295.18	-48.08	-48.14	0.14	100	307.81
16	0.25	**0.25**	0.00	100	107.72	0.25	**0.25**	0.00	100	108.77
17	5.00	**5.00**	0.00	100	113.74	5.00	**5.00**	0.00	100	113.69
18	680.63	**680.63**	0.00	100	185.19	680.63	**680.63**	0.00	100	186.24
19	-14.46	**-15.00**	0.75	100	319.67	-14.74	**-15.00**	0.50	100	319.70
20	-6961.81	**-6961.81**	0.00	100	169.53	-6961.81	**-6961.81**	0.00	100	178.29
21	24.31	**24.31**	0.01	100	284.93	24.31	**24.31**	0.01	100	289.19
22	1.00	**1.00**	0.00	100	110.40	1.00	**1.00**	0.00	100	110.19
23	-15.94	**-17.00**	1.46	100	188.87	-16.03	**-17.00**	1.34	100	187.75
24	-213.00	**-213.00**	0.00	100	190.52	-213.00	**-213.00**	0.00	100	195.43
25	-14.68	**-15.00**	0.63	100	300.82	-14.83	**-15.00**	0.47	100	308.98
26	-11.00	**-11.00**	0.00	100	169.43	-11.00	**-11.00**	0.00	100	170.90
27	-268.01	**-268.01**	0.01	100	380.93	-268.01	**-268.01**	0.01	100	389.09
28	-36.31	**-39.00**	6.24	100	311.33	-37.25	**-39.00**	5.13	100	316.04
29	-216.54	-383.17	66.52	100	500.00	-216.26	-380.89	70.97	100	500.00
30	-691.29	-831.76	53.39	100	500.00	-699.63	-868.69	78.69	100	500.00
31	-5190.01	-8333.66	1417.70	100	500.00	-5071.04	-8162.21	1324.25	100	500.00
32	-0.46	-0.59	0.06	100	194.54	-0.46	-0.66	0.06	100	188.34
33	-0.33	-0.41	0.05	100	186.59	-0.33	-0.41	0.05	100	189.72

Table 7.14: Results for FDEC on set A

The results given in Table 7.14 show that FDEC-SFP and FDEC-PFP located the best known global minimum for 26 and 27 problems respectively. Both were successful on 26 common problems. The FDEC-PFP method was successful on problem 14 where FDEC-SFP failed. Aside from problem 9, the standard deviations for the successful problems are relatively low for both schemes. For many of the problems (i.e. problems 2-8, 12,13, 16-18, 20-22, 24, 26 and 27) the mean and min values are equal and the standard deviation is 0. We can conclude for these problems the best known solution was found for all 100 runs. The FDEC-PFP method had feasible runs for all 33 problems while FDEC-SFP had feasible runs for all problems except problem 14 where it failed completely.

Just as with the DEC algorithm discussed in the previous section we summarize the results for FDEC-SFP and FDEC-PFP in Table 7.15 where we have included the summary of GA results as has been given in Table 7.4.

	SR	**TFeas**	**AvgIter**	**AvgFe**	**AvgSD**
FDEC-SFP	26	3200	214.23	21 738.2	1.9
FDEC-PFP	27	3300	218.78	22 197.8	4.3
GA-SFP	23	3168	221.89	22 511.9	3.27
GA-PFP	23	3300	199.86	20 286.9	24.01

Table 7.15: Summary of results for FDEC on set A

The summary in Table 7.15 shows that (when compared to GA) FDEC-SFP is successful for 3 more problems than GA-SFP. Also the FDEC-PFP method is successful on 4 more problems than GA-PFP method. Hence FDEC has a better success rate than GA for both schemes.

The TFeas values for both GA-PFP and FDEC-PFP indicate that all 100 runs for the 33 problems were successful in locating feasible points. Both GA-SFP and FDEC-SFP failed to locate any feasible points for problem 14. However FDEC-SFP found feasible points for all the remaining 32 problems.

We now compare FDEC and GA with respect to accuracy and efficiency. GA-PFP has the highest AvgSD and the lowest AvgFe value and is not comparable to the other methods.

From the remaining methods FDEC-SFP clearly performed the best with the lowest AvgFe and lowest AvgSD values. FDEC-PFP and GA-SFP are comparable with regard to both AvgIter and AvgSD values.

We now present the results for FDEC on problem set B in Table 7.16.

	S F P					**P F P**				
np	mean	min	dev	feas	iter	mean	min	dev	feas	iter
34	-308.36	**-316.27**	19.69	100	258.28	-306.1	**-316.27**	21.8	100	262.18
35	0.18	**0.18**	0.00	100	101.39	0.18	**0.18**	0.00	100	101.42
36	0	**0**	0.00	100	102.47	0	**0**	0.00	100	102.53
37	-195.37	**-195.37**	0.00	100	129.32	-195.37	**-195.37**	0.00	100	129.81
38	-2.21	**-2.21**	0.00	100	100.84	-2.21	**-2.21**	0.00	100	100.84
39	0.13	**0.13**	0.00	100	101.09	0.13	**0.13**	0.00	100	101.19
40	0.62	**0.62**	0.00	100	135.63	0.62	**0.62**	0.00	100	139.42
41	0.08	**0.08**	0.00	100	107.05	0.08	**0.08**	0.00	100	109.10
42	1.51	**1.51**	0.00	100	104.30	1.51	**1.51**	0.00	100	104.81
43	0.77	**0.76**	0.04	100	174.00	0.76	**0.76**	0.00	100	168.00
44	8903.02	8832.96	47.64	100	499.70	8915.20	8827.23	50.33	100	500.00
45	-0.41	-0.42	0.00	100	127.90	-0.41	-0.42	0.00	100	127.60

Table 7.16: Results for FDEC on set B

Looking at Table 7.16 we can see that FDEC-PFP successfully located the global minimum for all 12 problems whereas FDEC-SFP located the global minimum for 11 problems. Both have feasible runs for all problems. With regard to the standard deviation for the common successful problems, both FDEC-SFP and FDEC-PFP have low values for all problems except problem 34. FDEC-PFP also has a high standard deviation for problem 44. For many of the problems in this set, the min and mean values are equal and the standard deviation is zero. This indicates that the best known value was found in all runs. We have summarized the results in Table 7.17 where we have included the summary of GA as given in Table 7.6.

We now compare FDEC and GA using the results in Table 7.17. The table shows that FDEC has a higher SR than GA. In addition it is interesting to note that FDEC-PFP is the only method thus far to find the best known solution for problem 44. FDEC-SFP and FDEC-PFP were both successful in locating feasible points for all 100 runs of all 12 problems. GA however has 100 less feasible runs. The AvgIter and AvgSD values for FDEC-SFP is

	SR	TFeas	AvgIter	AvgFe	AvgSD
FDEC-SFP	11	1200	131.12	13 344.12	1.79
FDEC-PFP	12	1200	162.24	16 487.24	6.01
GA-SFP	9	1100	115.53	11 769.53	0.0
GA-PFP	9	1100	115.97	11 813.97	0.0

Table 7.17: Summary of FDEC on set B

much lower than those for FDEC-PFP. This can mainly be attributed to problem 44 where the FDEC-PFP was successful but the average number of iterations required and the standard deviation were both high. When compared to GA, we can see that the FDEC schemes have a much higher AvgIter and AvgSD value. This is due to the additional problems where the FDEC method was successful. Therefore for a fair comparison, we now present the summary using only the 9 problems (i.e. problems 35-42 and 45) that were solved by all methods. The summarized results are presented in Table 7.18.

	SR	TFeas	AvgIter	AvgFe	AvgSD
FDEC-SFP	9	900	112.22	11 435.22	0.0
FDEC-PFP	9	900	112.97	11 510.97	0.0
GA-SFP	9	900	115.53	11 769.53	0.0
GA-PFP	9	900	115.97	11 813.97	0.0

Table 7.18: Summary of results for FDEC on 9 common problems from set B

Table 7.18 shows that FDEC and GA performed equally well on the 9 problems with respect to SR, TFeas and AvgSD. The AvgIter values for FDEC however are lower than those for GA. We can conclude that the performance of FDEC is better than GA.

In conclusion, FDEC is better than GA with respect to SR on problem set A and B (see Table 7.15 and 7.17). Also the FDEC-PFP method has the highest SR for both problem set A and B. FDEC-PFP and GA-PFP are comparable with respect to TFeas for problem set A but FDEC-SFP is superior to GA-SFP. On set B FDEC is superior to GA on TFeas. FDEC-SFP and FDEC-PFP are comparable to GA-SFP with regard to AvgSD on set A and B. However both methods are superior to GA-PFP with respect to AvgSD on set A. For problem set A, the AvgIter values for FDEC-SFP and FDEC-PFP are comparable to those for GA-SFP but

are slightly inferior to GA-PFP. For problem set B the AvgIter values for DEC are superior than GA (see Table 7.18). Overall we can conclude that the FDEC method performed better than GA and that FDEC-PFP was the best overall performer.

7.3.1 A study on the filter set

We now make some observations about the size of the filter and the variations it experiences during a single independent run. We selected 3 problems whose ρ values vary. We then carried out a single run with the FDEC-SFP algorithm on each problem. We use all the settings and parameter values mentioned previously. We store the size of the filter at every 10 generations. We now plot the the filter size against the number of iterations. The graphs are presented in Figures 7.1 to 7.3, where the x axis gives the number of iterations and the y axis the size of the filter.

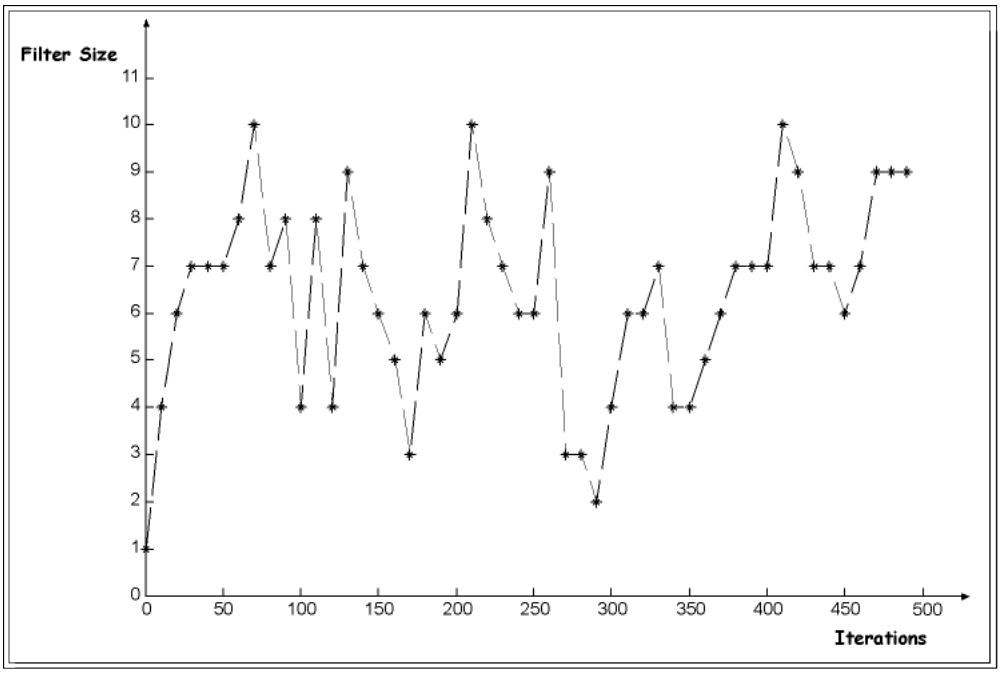

Figure 7.1: *Example of filter size, Problem 1,* $\rho = 0.0006$

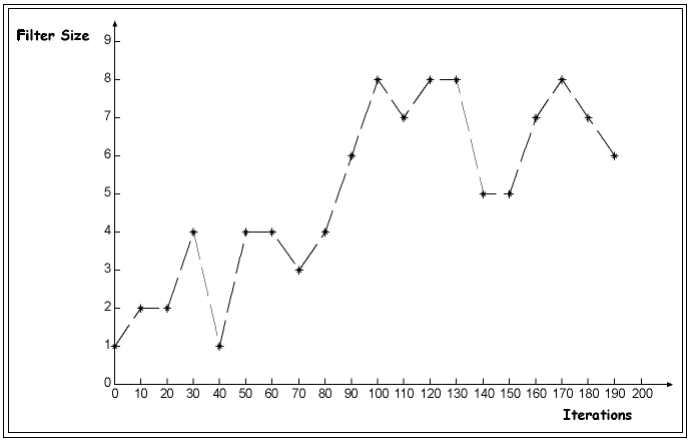

Figure 7.2: *Example of filter size, Problem 24,* $\rho = 23.405$

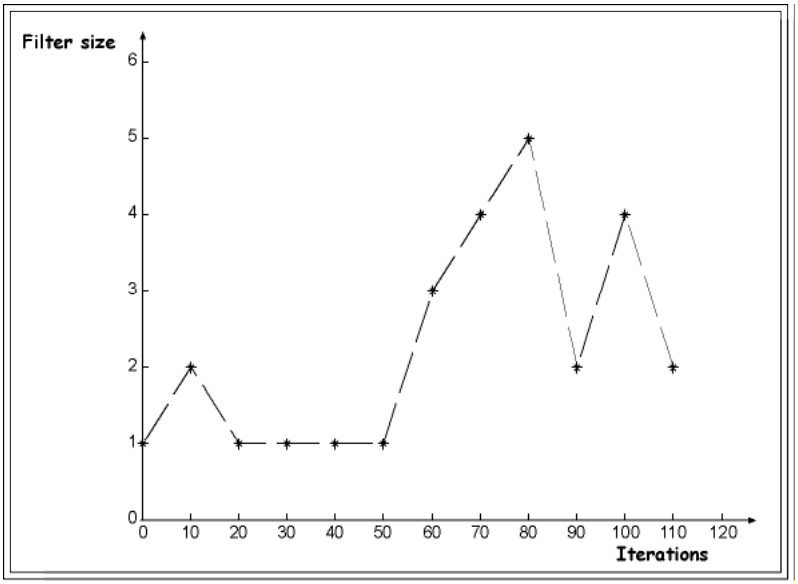

Figure 7.3: *Example of filter size, Problem 17,* $\rho = 96.645$

The most important distinction between these three problems is the amount of variation the filter experiences. We note for problem 1 the filter set is most active. This is because the feasible region is very small, i.e. ρ is small, and there will naturally be more infeasible points in the populations. For problem 24 where the feasible region is larger than problem 1, the filter set varies less and the maximum number of points in the filter is lower than problem 1. Problem 17 has the largest feasible region and hence has the least active and smallest filter set. We can conclude that for problems where the feasible region is small, the filter is more active. A larger filter set with varied points will certainly provide an advantage in terms of exploration for problems with small feasible regions. It is therefore clear that the filter set has an important role to play for constrained global optimization.

7.4 Results for the PSDEC algorithm

In this section we present the numerical results for the PSDEC algorithm. The PSDEC algorithm is based on the DEC algorithm except that it performs a local technique periodically using PPS. Hence the number of fitness calls in each iteration is not fixed i.e. the number of fitness evaluations for the iteration where PPS is invoked is greater than the population size. Therefore we have ignored the average iterations for each problem since they do not accurately reflect the efficiency of the method. Instead we use the average number of fitness evaluations. The parameters for PSDEC are the same as for DEC. We note that $c_r = 0.9$ and the setting best/1/bin are used. The PPS technique has been fully discussed in Chapter 6. PPS does not have any additional parameters. The results for PSDEC on test set A are presented in Table 7.19.

	S F P					P F P				
np	mean	min	dev	feas	feval	mean	min	dev	feas	feval
1	7049.29	**7049.25**	0.16	100	48519.14	7049.41	**7049.25**	0.62	100	49996.45
2	-30665.54	**-30665.54**	0.00	100	25128.58	-30665.54	**-30665.54**	0.00	100	25412.16
3	-310.00	**-310.00**	0.00	100	23413.34	-310.00	**-310.00**	0.00	100	24002.27
4	-4.52	-4.53	0.15	100	14874.54	-4.50	-4.53	0.21	100	16045.68
5	-3.14	-3.14	0.00	100	14410.21	-3.14	-3.14	0.00	100	15206.35
6	-13.40	-13.41	0.09	100	18178.26	-13.41	-13.41	0.00	100	20042.75
7	-5.51	**-5.51**	0.00	100	11614.40	-5.51	**-5.51**	0.00	100	11593.45
8	-16.78	**-16.78**	0.00	100	11294.08	-16.78	**-16.78**	0.00	100	11767.87
9	5193.77	5126.48	133.46	100	39462.19	5191.61	**5126.48**	111.52	100	41200.79
10	-0.38	-0.53	0.05	100	20886.46	-0.37	-0.47	0.04	100	19831.61
11	0.36	0.05	0.21	100	19124.77	0.37	0.05	0.23	100	19243.74
12	-1.00	**-1.00**	0.00	100	10474.91	-1.00	**-1.00**	0.00	100	10499.32
13	-0.10	**-0.10**	0.00	100	10725.76	-0.10	**-0.10**	0.00	100	10727.77
14	-	-	-	-	-	-1.11	-1.12	0.00	100	25282.51
15	-48.09	-48.14	0.12	100	29907.30	-48.03	-48.14	0.39	100	32689.79
16	0.25	**0.25**	0.00	100	10961.82	0.25	**0.25**	0.00	100	11087.29
17	5.00	**5.00**	0.00	100	11643.63	5.00	**5.00**	0.00	100	11612.17
18	680.63	**680.63**	0.00	100	18699.87	680.63	**680.63**	0.00	100	19289.99
19	-14.55	**-15.00**	0.60	100	31455.83	-14.49	**-15.00**	0.77	100	31989.23
20	-6961.81	**-6961.81**	0.00	100	16534.48	-6961.81	**-6961.81**	0.00	100	19450.32
21	24.31	**24.31**	0.01	100	29197.20	24.31	**24.31**	0.00	100	30158.60
22	1.00	**1.00**	0.00	100	11155.53	1.00	**1.00**	0.00	100	11179.70
23	-16.00	**-17.00**	1.29	100	19220.89	-15.78	**-17.00**	1.54	100	19462.46
24	-213.00	**-213.00**	0.00	100	19695.99	-213.00	**-213.00**	0.00	100	19662.66
25	-14.72	**-15.00**	0.58	100	30480.10	-14.72	**-15.00**	0.55	100	31043.53
26	-11.00	**-11.00**	0.00	100	17368.44	-11.00	**-11.00**	0.00	100	17213.75
27	-268.01	**-268.01**	0.01	100	37950.09	-268.01	**-268.01**	0.01	100	40301.44
28	-37.66	**-39.00**	4.47	100	31137.22	-37.38	**-39.00**	4.88	100	31651.00
29	-218.30	-365.16	67.11	100	51106.84	-217.70	-385.35	72.32	100	51137.89
30	-693.02	-853.97	69.27	100	51316.80	-678.21	-866.61	64.40	100	51377.30
31	-5201.58	-8207.83	1286.38	100	51147.41	-5147.22	-8379.78	1560.19	100	49947.02
32	-0.46	-0.62	0.06	100	19442.54	-0.47	-0.67	0.06	100	20079.24
33	-0.32	-0.43	0.06	100	18278.86	-0.34	-0.41	0.04	100	19212.46

Table 7.19: Results for PSDEC on set A

Table 7.19 shows that PSDEC-SFP and PSDEC-PFP were successful in finding the best known minimum for 26 and 27 problems respectively. PSDEC-PFP located feasible points for all runs of all problems but PSDEC-SFP failed completely on problem 14 where no feasible points were located for any of the runs. The standard deviation for PSDEC was low for all successful problems except problem 9. Some of the problems (i.e. problems 2,3,5,7,8,12,13,16-18,20, 24 and 26) the mean and min values are equal and the standard deviation is 0. We can conclude for these problems the best known solution was found for all 100 runs. Table 7.20 presents a summary of the results for PSDEC as given in Table 7.19. We have also included the results for GA as has been summarized in Table 7.4.

	SR	**TFeas**	**AvgFe**	**AvgSD**
PSDEC-SFP	26	3200	21 639.6	5.42
PSDEC-PFP	27	3300	22 511.6	4.47
GA-SFP	23	3168	22 511.9	3.27
GA-PFP	23	3300	20 286.9	24.01

Table 7.20: Summary of results for PSDEC on set A

A comparison of PSDEC and GA shows that PSDEC-SFP is successful on 3 more problems than GA while PSDEC-PFP is successful on 4 more problems than GA. Clearly, PSDEC performed better than GA with respect to SR. Table 7.20 shows that PSDEC-PFP is comparable to GA-PFP with respect to TFeas but PSDEC-SFP is superior to GA-SFP.

We now compare GA and PSDEC with respect to AvgSD and AvgFe. Table 7.20 shows us that GA-PFP has the best AvgFe and worst AvgSD. From the remaining three methods PSDEC is better with respect to AvgFe and GA-SFP is better with regard to AvgSD.

Next we present the results for the PSDEC method on problem set B. The results are presented in Table 7.21.

Table 7.21 shows that PSDEC located the global minimum for 11 common problems. For all the problems the schemes, SFP and PFP, managed to locate feasible points on all runs. The standard deviation for all successful problems except problem 34 are very low. We have summarized the results in Table 7.22 where we have also included the summary of GA

Superiority of Feasible Points						Parameter Free Penalties				
np	mean	min	dev	feas	func eval	mean	min	dev	feas	func eval
34	-310.05	**-316.27**	17.76	100	26029.7	-307.8	**-316.27**	20.26	100	27137.01
35	0.18	**0.18**	0	100	10377.07	0.18	**0.18**	0	100	10404.72
36	0	**0**	0	100	10481.89	0	**0**	0	100	10504.99
37	-195.37	**-195.37**	0	100	13157.11	-195.37	**-195.37**	0	100	13252.96
38	-2.21	**-2.21**	0	100	10331.63	-2.21	**-2.21**	0	100	10314.72
39	0.13	**0.1**3	0	100	10357.18	0.13	**0.13**	0	100	10350.12
40	0.62	**0.62**	0	100	13835.87	0.62	**0.62**	0	100	13949.85
41	0.08	**0.08**	0.00	100	10958.83	0.08	**0.08**	0.00	100	11033.13
42	1.51	**1.51**	0.00	100	10700.91	1.51	**1.51**	0.00	100	10687.70
43	0.76	**0.76**	0.00	100	16771.55	0.76	**0.76**	0.00	100	16986.55
44	8828.75	8894.81	43.66	100	51626.63	8839.64	8905.88	47.31	100	51612.49
45	-0.41	-0.42	0.00	100	12922.71	-0.41	-0.42	0.00	100	12941.93

Table 7.21: Results for PSDEC on set B

results on set B from Table 7.6.

	SR	**TFeas**	**AvgFe**	**AvgSD**
PSDEC-SFP	11	1200	13 265.86	1.65
PSDEC-PFP	11	1200	13 414.88	1.84
GA-SFP	9	1100	11 769.53	0.0
GA-PFP	9	1100	11 813.97	0.0

Table 7.22: Summary of results for PSDEC on set B

From Table 7.22 we see that when compared to GA the SR of the PSDEC algorithm is much better. Both PSDEC-SFP and PSDEC-PFP were successful in finding feasible points for all runs of all 12 problems wheres GA was successful for only 11 problems. PSDEC-SFP has lower AvgSD and AvgIter values than PSDEC-PFP. Both PSDEC-SFP and PSDEC-PFP have higher AvgIter and AvgSD values than GA. This is due to problems 34 and 43 where the dev and iter values are high. For a fair comparison, we now compare GA and PSDEC on 9 common problems that were solved by both methods. The results are presented in Table 7.23.

Table 7.23 shows that PSDEC and GA performed equally well on the 9 problems with respect to SR, TFeas and AvgSD. The AvgIter values for PSDEC however are lower than those for GA. We can conclude that the overall performance of PSDEC is better than GA.

To sum up, PSDEC-SFP and PSDEC-PFP are better than GA with respect to SR on

	SR	TFeas	AvgFe	AvgSD
PSDEC-SFP	9	900	11 458.13	0.0
PSDEC-PFP	9	900	11 493.35	0.0
GA-SFP	9	900	11 769.53	0.0
GA-PFP	9	900	11 813.97	0.0

Table 7.23: Summary of results for PSDEC on 9 common problems from set B

problem set A and B (see Table 7.20 and 7.22). Both PFP schemes for PSDEC and GA perform equally with respect to TFeas for problem set A but PSDEC-SFP was superior to GA-SFP. On problem set B PSDEC was better than GA with respect to TFeas. PSDEC is comparable to GA-SFP with regard to AvgSD on both problem set A and B. However PSDEC is superior to GA-PFP on set A with respect to AvgSD. For problem set A, the AvgIter values for PSDEC are better than GA-SFP but are slightly inferior to GA-PFP. For problem set B the AvgIter values for PSDEC are superior than GA (see Table 7.23). Overall we can conclude that PSDEC performed better than GA.

7.5 Results for the PSFDEC algorithm

In this section we present the results for the PSFDEC algorithm. The PSFDEC algorithm is implemented using $c_r = 0.9$ and the setting best/1/bin. Also as with the PSDEC algorithm, we present the average number of function evaluations, feval, instead of the average number of iterations, iter. The results for problem set A are presented in Table 7.24.

	S F P					**S F P**				
np	mean	min	dev	feas	feval	mean	min	dev	feas	feval
1	7049.32	**7049.25**	0.17	100	49618.16	7049.29	**7049.25**	0.07	100	50121.48
2	-30665.54	**-30665.54**	0.00	100	25619.79	-30665.54	**-30665.54**	0.00	100	25613.25
3	-310.00	**-310.00**	0.00	100	23653.67	-310.00	**-310.00**	0.00	100	23996.65
4	-4.52	-4.53	0.15	100	14989.61	-4.53	-4.53	0.00	100	15563.60
5	-3.14	-3.14	0.00	100	14403.05	-3.14	-3.14	0.00	100	14946.62
6	-13.41	-13.41	0.00	100	18453.86	-13.41	-13.41	0.00	100	19100.10
7	-5.51	**-5.51**	0.00	100	11583.18	-5.51	**-5.51**	0.00	100	11569.12
8	-16.78	**-16.78**	0.00	100	11303.81	-16.78	**-16.78**	0.00	100	11582.60
9	5148.52	5126.48	43.88	100	39665.95	5196.02	5126.48	113.53	100	42023.24
10	-0.38	-0.48	0.05	100	20283.30	-0.38	-0.45	0.04	100	20427.70
11	0.35	0.05	0.24	100	18427.04	0.36	0.05	0.25	100	19381.10
12	-1.00	**-1.00**	0.00	100	10484.97	-1.00	**-1.00**	0.00	100	10478.32
13	-0.10	**-0.10**	0.00	100	10721.84	-0.10	**-0.10**	0.00	100	10732.13
14	-	-	-	-	-	-1.11	**-1.12**	0.00	100	25511.55
15	-48.09	-48.14	0.11	100	31413.38	-48.09	-48.14	0.11	100	30983.37
16	0.25	**0.25**	0.00	100	10973.25	0.25	**0.25**	0.00	100	11165.48
17	5.00	**5.00**	0.00	100	11604.03	5.00	**5.00**	0.00	100	11578.70
18	680.63	**680.63**	0.00	100	18954.71	680.63	**680.63**	0.00	100	19227.43
19	-14.60	**-15.00**	0.64	100	31627.53	-14.74	**-15.00**	0.52	100	32748.47
20	-6961.81	**-6961.81**	0.00	100	17337.83	-6961.81	**-6961.81**	0.00	100	18497.09
21	24.31	**24.31**	0.00	100	29362.44	24.31	**24.31**	0.01	100	29621.18
22	1.00	**1.00**	0.00	100	11217.33	1.00	**1.00**	0.00	100	11194.87
23	-16.22	**-17.00**	1.05	100	19514.40	-16.01	**-17.00**	1.12	100	19284.90
24	-213.00	**-213.00**	0.00	100	19514.24	-213.00	**-213.00**	0.00	100	19856.68
25	-14.69	**-15.00**	0.57	100	30852.66	-14.80	**-15.00**	0.49	100	31360.86
26	-11.00	**-11.00**	0.00	100	17302.24	-11.00	**-11.00**	0.00	100	17588.81
27	-268.01	**-268.01**	0.01	100	38826.12	-268.01	**-268.01**	0.02	100	39852.48
28	-36.61	**-39.00**	5.41	100	31828.79	-36.65	**-39.00**	5.55	100	32144.37
29	-224.47	-384.69	74.08	100	51102.93	-216.85	-378.93	67.73	100	51138.93
30	-687.49	-867.93	68.65	100	51303.64	-695.85	-858.73	74.84	100	51344.88
31	-4942.59	-8435.19	1452.69	100	51132.15	-5051.79	-8115.77	1374.30	100	51148.04
32	-0.45	-0.61	0.06	100	19664.79	-0.45	-0.73	0.07	100	19454.87
33	-0.31	-0.38	0.06	100	17789.10	-0.34	-0.41	0.04	100	19331.30

Table 7.24: Results for PSFDEC on set A

Tables 7.24 show that PSFDEC-SFP and PSFDEC-PFP successful located the best known local minimum for 26 and 27 problems respectively. PSFDEC-PFP found feasible points for all runs while PSFDEC-SFP only failed to find any feasible points for problem 14. The standard deviation for both methods was low for successful problems except for problem 9. Some of the problems (i.e. problems 2,3,5-8,12,13,16-18,20-22,24 and 26) the mean and min values are equal and the standard dev is 0. We can conclude for these problems the best known solution was found for all 100 runs. We now summarize the results for the PSFDEC algorithms in Table 7.25 where we have also included the summary for GA as given in Table 7.4.

	SR	**TFeas**	**AvgFe**	**AvgSD**
PSFDEC-SFP	26	3200	21 894.4	2.01
PSFDEC-PFP	27	3300	22 434.2	4.51
GA-SFP	23	3168	22 511.9	3.27
GA-PFP	23	3300	20 286.9	24.01

Table 7.25: Summary of results for PSFDEC on set A

From Table 7.25 we see that PSFDEC-SFP is successful for 3 more problems than GA-SFP and PSFDEC-PFP is successful on 4 more problems than GA-PFP . Hence both PSFDEC-SFP and PSFDEC-PFP have a better SR than GA.

Next we compare TFeas for GA and PSFDEC. The TFeas values for both GA-PFP and PSFDEC-PFP indicate that all 100 runs for the 33 problems were successful in locating feasible points. Both GA-SFP and PSFDEC-SFP failed to locate any feasible points for problem 14.

Finally, we compare GA and PSFDEC with respect to AvgSD and AvgFe. GA-PFP method again has the lowest AvgFe and highest AvgFe. From the remaining three methods PSFDEC-SFP has the lowest AvgFe and AvgSD values. We can say that the results for the PSFDEC-SFP are superior. PSFDEC-PFP and GA-SFP are comparable.

We now present results for PSFDEC on set B in Table 7.26.

From Table 7.26 we see that both schemes, SFP and PFP, fail to find the best known

	S F P					P F P				
np	mean	min	dev	feas	feval	mean	min	dev	feas	feval
34	-308.93	**-316.27**	19.08	100	26174.26	-306.1	**-316.27**	21.8	100	26842.56
35	0.18	**0.18**	0.00	100	10402.5	0.18	**0.18**	0.00	100	10382.38
36	0	**0**	0.00	100	10466.78	0	**0**	0.00	100	10487.01
37	-195.37	**-195.37**	0.00	100	13175.25	-195.37	**-195.37**	0.00	100	13242.92
38	-2.21	**-2.21**	0.00	100	10333.72	-2.21	**-2.21**	0.00	100	10313.9
39	0.13	**0.13**	0.00	100	10358.33	0.13	**0.13**	0.00	100	10336.98
40	0.62	**0.62**	0.00	100	13921.01	0.62	**0.62**	0.00	100	14288.91
41	0.08	**0.08**	0.00	100	11033.71	0.08	**0.08**	0.00	100	11064.98
42	1.51	**1.51**	0.00	100	10714.13	1.51	**1.51**	0.00	100	10702.80
43	0.77	**0.76**	0.04	100	18513.29	0.76	**0.76**	0.00	100	17210.56
44	8904.02	8839.28	42.36	100	51624.75	8906.09	8836.32	43.79	100	51599.30
45	-0.41	-0.42	0.00	100	12920.13	-0.41	-0.42	0.01	100	18256.78

Table 7.26: Results for PSFDEC on set B

minimum for a single problem only i.e. problem 44. All runs for both schemes managed to locate feasible points for all the problems. Aside from problem 34, the standard deviation for the successful problems is very low. Also for many problems the mean and min values are equal indicating that the best known solution was located for all runs. We summarize the results for PSFDEC on set B in Table 7.27 and include the summarized results of GA as given in Table 7.6.

	SR	**TFeas**	**AvgFe**	**AvgSD**
PSFDEC-SFP	11	1200	13 455.74	1.74
PSFDEC-PFP	11	1200	13 920.89	1.99
GA-SFP	9	1100	11 769.53	0.0
GA-PFP	9	1100	11 813.97	0.0

Table 7.27: Summary of results for PSFDEC on set B

Table 7.27 shows that PSFDEC performed better than GA with respect to SR. The TFeas value indicates that PSFDEC located feasible points for all 100 runs of all 12 problems while GA also failed on 100 runs. When we compare AvgSD and AvgIter of GA to PSFDEC we find that PSFDEC is inferior with regard to both criteria. This is due to problems 34 and 43 where the PSFDEC methods were successful and GA was unsuccessful. We therefore compare the results of GA and PSFDEC on the 9 successful problems that are common. The results are presented in Table 7.28

	SR	TFeas	AvgFe	AvgSD
PSFDEC-SFP	9	900	11 480.62	0.0
PSFDEC-PFP	9	900	12 119.63	0.0
GA-SFP	9	900	11 769.53	0.0
GA-PFP	9	900	11 813.97	0.0

Table 7.28: Summary of results for PSFDEC on 9 common problems from set B

From Table 7.28 we see that SR, TFeas and AvgSD are equal for PSDEC and GA. The AvgIter values of PSDEC-SFP are lower than those of GA. The AvgIter values of PSFDEC-PFP however are higher than GA. We can conclude that the performance of the PSDEC-SFP is slightly better than GA while PSFDEC-PFP is slightly worst.

To sum up, PSFDEC is better than GA with respect to SR on problem set A and B (see Table 7.25 and 7.26). PSFDEC-PFP and GA-PFP are comparable with respect to TFeas for problem set A but PSFDEC-SFP is superior to GA-SFP. On set B PSFDEC is superior to GA on TFeas. PSFDEC is comparable to GA-SFP with regard to AvgSD on both A and B. However PSFDEC is superior to GA-PFP on set A. For problem set A, the AvgIter values for PSFDEC are comparable to GA-SFP but are slightly inferior to GA-PFP. For problem set B the AvgIter values for PSFDEC-SFP are superior than GA while those of PSFDEC-PFP are inferior (see Table 7.28). Overall we can conclude that PSFDEC performed better than GA in most regards.

7.6 Comparisons with other methods

Thus far we have compared the new algorithms with GA presented in [38]. In this section we compare the new algorithms with some other algorithms presented in literature. In particular, we look at results presented in [23], [25], [30], [39] and [51]. The results for the algorithms are presented in Table 7.29 and the algorithms considered are:

- Homomorphous Mappings (HM) method [30],
- Stochastic Ranking (SRA) method [51],
- Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA) [23],
- Simple Multimember Evolutionary Strategy (SMES) method [39], and
- Filter Simulated Annealing (FSA) method [25].

The results we present are from a set of 13 benchmark problems. We have only included the results for those problems that are also a part of our set of test problems. For example, in Table 7.29 G1 corresponds to problem 19, G4 to 2, G5 to 9, G6 to 20 , G7 to 21 , G8 to 13, G9 to 18, G10 to 1 and G13 corresponds to 11. This gives us a total of 9 problems. Since the above algorithms are all implemented differently with different accuracies, termination criteria and parameter values, it would be unfair and difficult to draw any conclusions from a direct comparisons of these results. However we wish to highlight and comment on certain aspects.

Table 7.29 shows that none of the algorithms was successful in locating the minimum for all the above mentioned problems. However all four of our algorithms presented were successful for all these problems. This indicates that the proposed algorithms are more reliable than some recent algorithms presented in literature.

Next we consider the efficiency of these methods. The computation cost for HM, SRA, ASCHEA and SMES are fixed for each algorithm. The respective algorithms were terminated after 1400000, 350000, 1500000 and 250000 fitness function evaluations. A fitness

np		HM	SRA	ASCHEA	SMES	FSA
19 (G1)	Best	-14.79	**-15.00**	**-15.00**	**-15.00**	**-15.00**
	Av.	-14.71	-15.00	-14.84	-15.00	-14.99
2 (G4)	Best	-30664.50	**-30665.54**	-30665.50	**-30665.54**	**-30665.54**
	Av.	-30655.30	-30665.54	-30665.50	-30665.54	-30665.47
9 (G5)	Best	-	5126.50	5126.50	5126.60	5126.50
	Av.	-	5128.88	5141.65	5174.49	5126.50
20 (G6)	Best	-6952.10	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**
	Av.	-6342.60	-6875.94	-6961.81	-6961.28	-6961.81
21 (G7)	Best	24.62	**24.31**	24.33	24.33	**24.31**
	Av.	24.83	24.37	24.66	24.47	24.38
13 (G8)	Best	**0.10**	**0.10**	**0.10**	**0.10**	**0.10**
	Av.	0.09	0.10	0.10	0.10	0.10
18 (G9)	Best	680.91	**680.63**	**680.63**	**680.63**	**680.63**
	Av.	681.16	680.66	680.64	680.64	680.64
1 (G10)	Best	7147.90	7054.32	7061.13	7051.90	7059.86
	Av.	8163.60	7559.19	7497.43	7253.05	7509.32
11 (G13)	Best	N.A.	**0.05**	N.A.	**0.05**	**0.05**
	Av.	N.A.	0.06	N.A.	0.17	0.3

Table 7.29: Miscellaneous results

function evaluation includes the evaluation of the objective function and evaluation of the constraints. This is the case for the new algorithms as well. For the FSA method the function evaluations and constraint evaluations are considered separately. For this algorithm, the function evaluations range from 44538 to 324569, while the constraint evaluation range from 15817 to 171299. For all of the new algorithms we limited the number of iterations to 500. This would result in a maximum of 50601 fitness function evaluations. From all the results we presented the most expensive algorithm was the DEC algorithm with a random mutation scheme. The PSDEC-PFP method required an average of 22511.6 fitness function evaluations for a successful run. This value is an average for those problems where the best known minimum was located. Even though this is the most expensive of all the algorithms presented, the average number of fitness evaluations required is significantly lower than all the values given for the above algorithms listed in Table 7.29. We can surmise that the new algorithms have a better success rate and are also more efficient than the algorithms presented in Table 7.29.

7.7 Overall comparison of the new algorithms with GA

In this section we provide a summary of all the results for both problem sets presented in the previous sections. We first summarize all the results for each problem set into four tables. Each table will give the results of all five algorithms. The first table will give the best minimum values found by each algorithm for each problem. The next table will give the mean values for each problem. The third table will give the standard deviation values and the last table will give the average fitness function evaluations for each problem. In each table we indicate the best results found amongst all the algorithms in bold. We present the results for problem set A first.

7.7.1 Results on problem set A

	GA		DEC		FDEC		PSDEC		PSFDEC	
np	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP
1	7116.64	7292.1	**7049.25**	**7049.25**	**7049.25**	**7049.25**	**7049.25**	**7049.25**	**7049.25**	**7049.25**
2	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**
3	**-310**	**-310**	**-310**	**-310**	**-310**	**-310**	**-310**	**-310**	**-310**	**-310**
4	**-4.53**	**-4.53**	**-4.53**	**-4.53**	**-4.53**	**-4.53**	**-4.53**	**-4.53**	**-4.53**	**-4.53**
5	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**
6	**-13.41**	**-13.41**	**-13.41**	**-13.41**	**-13.41**	**-13.41**	**-13.41**	**-13.41**	**-13.41**	**-13.41**
7	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**
8	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**
9	**4221.83**	**4221.83**	5126.48	5126.48	5126.48	5126.48	5126.48	5126.48	5126.48	5126.48
10	-0.64	-0.64	-0.49	-0.48	-0.46	-0.45	-0.53	-0.47	-0.48	-0.45
11	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**
12	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**
13	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**
14	-	-1.01	-	**-1.12**	-	**-1.12**	-	**-1.12**	-	**-1.12**
15	-48.11	-47.97	**-48.14**	**-48.14**	**-48.14**	**-48.14**	**-48.14**	**-48.14**	**-48.14**	**-48.14**
16	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**
17	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
18	680.81	680.75	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**
19	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**
20	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**
21	24.77	25.76	**24.31**	**24.31**	**24.31**	**24.31**	**24.31**	**24.31**	**24.31**	**24.31**
22	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
23	**-17**	**-17**	**-17**	**-17**	**-17**	**-17**	**-17**	**-17**	**-17**	**-17**
24	**-213**	**-213**	**-213**	**-213**	**-213**	**-213**	**-213**	**-213**	**-213**	**-213**
25	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**
26	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**
27	**-268.01**	-268	**-268.01**	**-268.01**	**-268.01**	**-268.01**	**-268.01**	**-268.01**	**-268.01**	**-268.01**
28	**-39**	**-39**	**-39**	**-39**	**-39**	**-39**	**-39**	**-39**	**-39**	**-39**
29	-221.11	-247.72	-383.13	-364.69	-383.17	-380.89	-365.16	-385.35	-384.69	-378.93
30	-696.76	-698.08	-869.2	-841.8	-831.76	-868.69	-853.97	-866.61	-867.93	-858.73
31	-5374.88	-5424.69	-8534.11	-8101.83	-8333.66	-8162.21	-8207.83	-8379.78	-8435.19	-8115.77
32	-0.74	-0.74	-0.7	-0.7	-0.59	-0.66	-0.62	-0.67	-0.61	-0.73
33	-0.57	-0.57	-0.43	-0.41	-0.41	-0.41	-0.43	-0.41	-0.38	-0.41

Table 7.30: Set A: Best minimum values (min)

	GA		DEC		FDEC		PSDEC		PSFDEC	
np	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP
1	7893.74	8464.55	**7049.29**	7049.39	7049.33	7049.38	**7049.29**	7049.41	7049.32	**7049.29**
2	-30665.53	-30665.53	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**	**-30665.54**
3	-309.84	-308.58	**-310**	**-310**	**-310**	**-310**	**-310**	**-310**	**-310**	**-310**
4	-4.52	-4.41	**-4.53**	**-4.53**	**-4.53**	-4.52	-4.52	-4.5	-4.52	**-4.53**
5	-3.13	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**
6	-13.32	-13.38	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**	**-3.14**
7	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**	**-5.51**
8	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**	**-16.78**
9	**4239.21**	4755.32	5210.17	5201.06	5143.65	5191.18	5193.77	5191.61	5148.52	5196.02
10	-0.56	-0.56	-0.38	-0.38	-0.38	-0.38	-0.38	-0.37	-0.38	-0.38
11	0.38	0.58	**0.33**	0.34	0.36	**0.33**	0.36	0.37	0.35	0.36
12	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**
13	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**
14	-	-0.78	-	**-1.11**	-	**-1.11**	-	**-1.11**	-	**-1.11**
15	-47.01	-47.03	-48.07	-48.07	-48.08	-48.08	**-48.09**	-48.03	**-48.09**	**-48.09**
16	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**	**0.25**
17	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
18	681.56	682.75	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**
19	**-14.94**	**-14.98**	-14.64	-14.48	-14.46	-14.74	-14.55	-14.49	-14.6	-14.74
20	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**	**-6961.81**
21	26.87	32.63	**24.31**	**24.31**	**24.31**	**24.31**	**24.31**	**24.31**	**24.31**	
22	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
23	15.98	-15.81	-16.04	-16.03	-15.94	-16.03	-16	-15.78	**-16.22**	-16.01
24	-212.98	-212.98	**-213**	**-213**	**-213**	**-213**	**-213**	**-213**	**-213**	**-213**
25	**-15**	**-15**	-14.7	-14.6	-14.68	-14.83	-14.72	-14.72	-14.69	-14.8
26	**-11**	-10.99	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**	**-11**
27	-265.81	-265.06	**-268.01**	**-268.01**	**-268.01**	**-268.01**	**-268.01**	**-268.01**	**-268.01**	**-268.01**
28	-36.66	-37.05	-36.94	-36.7	-36.31	-37.25	-37.66	-37.38	-36.61	-36.65
29	-135.08	-132.02	-216.62	-204.69	-216.54	-216.26	-218.3	-217.7	-224.47	-216.85
30	-593.81	-586.45	-688.27	-678.76	-691.29	-699.63	-693.02	-678.21	-687.49	-695.85
31	-3043.37	-3106.12	-5228.99	-5021.31	-5190.01	-5071.04	-5201.58	-5147.22	-4942.59	-5051.79
32	-0.66	-0.66	-0.46	-0.46	-0.46	-0.46	-0.46	-0.47	-0.45	-0.45
33	-0.5	-0.5	-0.34	-0.33	-0.33	-0.33	-0.32	-0.34	-0.31	-0.34

Table 7.31: Set A: Mean values (mean)

	GA		DEC		FDEC		PSDEC		PSFDEC	
np	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP
1	1285.08	1294.91	**0.06**	0.82	0.3	0.49	0.16	0.62	0.17	0.07
2	0.06	0.01	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
3	1.6	12.69	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
4	0.15	0.4	**0**	**0**	**0**	0.15	0.15	0.21	0.15	**0**
5	0.02	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
6	0.26	0.15	**0**	**0**	**0**	0.09	0.09	**0**	**0**	**0**
7	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
8	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
9	62.14	531.2	142.27	138.03	**39.5**	107.53	133.46	111.52	43.88	113.53
10	0.03	0.03	0.04	0.05	0.04	0.03	0.05	0.04	0.05	0.04
11	0.29	0.35	0.23	0.24	0.24	0.22	**0.21**	0.23	0.24	0.25
12	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
13	0.01	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
14	-	0.09	-	**0**	-	**0**	-	**0**	-	**0**
15	0.77	0.77	0.24	0.21	0.17	0.14	0.12	0.39	**0.11**	**0.11**
16	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
17	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
18	**0**	2.17	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
19	0.34	**0.2**	0.59	0.73	0.75	0.5	0.6	0.77	0.64	0.52
20	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
21	1.37	5.64	0.01	0.01	0.01	0.01	0.01	**0**	**0**	0.01
22	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
23	1.16	1.54	1.21	1.29	1.46	1.34	1.29	1.54	**1.05**	1.12
24	0.08	0.09	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
25	**0**	**0**	0.54	0.63	0.63	0.47	0.58	0.55	0.57	0.49
26	**0**	0.05	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
27	3.21	3.42	0.03	**0**	0.01	0.01	0.01	0.01	0.01	0.02
28	5.19	4.73	4.94	5.92	6.24	5.13	**4.47**	4.88	5.41	5.55
29	38.63	40.54	75.4	60.62	66.52	70.97	67.11	72.32	74.08	67.73
30	34.34	29.27	74.32	69.48	53.39	78.69	69.27	64.4	68.65	74.84
31	682.9	695.63	1519.82	1454.09	1417.7	1324.25	1286.38	1560.19	1452.69	1374.3
32	0.04	0.04	0.07	0.07	0.06	0.06	0.06	0.06	0.06	0.07
33	0.03	0.03	0.04	0.06	0.05	0.05	0.06	0.04	0.06	0.04

Table 7.32: Set A: Standard deviation values (dev)

	GA		DEC		FDEC		PSDEC		PSFDEC	
np	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP
1	42167.5	**10302**	47855.82	49081.96	49163.77	49050.65	48519.14	49996.45	49618.16	50121.48
2	30027.3	30027.3	**24933.87**	25426.75	25360.09	25163.14	25128.58	25412.16	25619.79	25613.25
3	**22078.6**	22775.5	23392.61	23667.33	23790.55	24004.67	23413.34	24002.27	23653.67	23996.65
4	**14826.8**	15089.4	**14826.8**	15887.3	14962.14	15401.49	14874.54	16045.68	14989.61	15563.6
5	**13119.9**	13200.7	13985.47	15200.5	14452.09	14839.93	14410.21	15206.35	14403.05	14946.62
6	30279.8	30401	**17924.47**	19368.77	18178.99	19296.05	18178.26	20042.75	18453.86	19100.1
7	**11271.6**	11301.9	11519.05	11596.82	11540.26	11627.12	11614.4	11593.45	11583.18	11569.12
8	11766.5	11867.5	11273.62	11971.53	**11267.56**	11794.78	11294.08	11767.87	11303.81	11582.6
9	40965.6	**10473.7**	39324.35	40254.56	39514.23	40170.73	39462.19	41200.79	39665.95	42023.24
10	39965.7	39965.7	20499.97	19712.17	20246.46	20337.36	20886.46	19831.61	20283.3	20427.7
11	30855.5	**11029.2**	18943.56	19090.01	18438.56	19597.03	19124.77	19243.74	18427.04	19381.1
12	10483.8	10473.7	10446.43	10434.31	10474.71	**10429.26**	10474.91	10499.32	10484.97	10478.32
13	10847.4	10877.7	10699.94	10696.91	10686.81	10686.81	10725.76	10727.77	**10721.84**	10732.13
14	-	46702.4	-	**24400.59**	-	24477.35	-	25282.51	-	25511.55
15	47288.2	45217.7	**30005.08**	34078.41	29914.18	31189.81	29907.3	32689.79	31413.38	30983.37
16	**10807**	**10807**	10964.56	11022.13	10980.72	11086.77	10961.82	11087.29	10973.25	11165.48
17	11261.5	**11211**	11611.97	11538.24	11588.74	11583.69	11643.63	11612.17	11604.03	11578.7
18	**14614.7**	22856.3	18389.07	19209.19	18805.19	18911.24	18699.87	19289.99	18954.71	19227.43
19	32814.9	33138.1	**31380.7**	31962.46	32387.67	32390.7	31455.83	31989.23	31627.53	32748.47
20	14614.7	**14584.4**	16406.44	19621.27	17223.53	18108.29	16534.48	19450.32	17337.83	18497.09
21	46500.4	**10968.6**	28575.93	30108.1	28878.93	29309.19	29197.2	30158.6	29362.44	29621.18
22	10877.7	**10867.6**	11206.96	11167.57	11251.4	11230.19	11155.53	11179.7	11217.33	11194.87
23	**15695.4**	15766.1	19647.53	19433.41	19176.87	19063.75	19220.89	19462.46	19514.4	19284.9
24	21644.3	21977.6	19538.45	19630.36	**19343.52**	19839.43	19695.99	19662.66	19514.24	19856.68
25	31714	31754.4	30465.64	**30169.71**	30483.82	31307.98	30480.1	31043.53	30852.66	31360.86
26	20735.3	21583.7	17263.93	**17104.35**	17213.43	17361.9	17368.44	17213.75	17302.24	17588.81
27	47490.2	47429.6	37956.81	40134.37	38574.93	39399.09	**37950.09**	40301.44	38826.12	39852.48
28	26300.4	**25472.2**	30829.24	32109.92	31545.33	32021.04	31137.22	31651	31828.79	32144.37
29	50025.3	50419.2	50601	50601	50601	50601	51106.84	51137.89	51102.93	51138.93
30	46924.6	46369.1	50601	50601	50601	50601	51316.8	51377.3	51303.64	51344.88
31	49247.6	49601.1	50601	50001.06	50601	50601	51147.41	49947.02	51132.15	51148.04
32	35461.1	35461.1	19748.53	19621.27	19749.54	19123.34	19442.54	20079.24	19664.79	19454.87
33	41339.3	41339.3	19378.87	19047.59	18946.59	19262.72	18278.86	19212.46	17789.1	19331.3

Table 7.33: Set A: Fitness function evaluation values (feval)

We now summarize the results for problem set A in Table 7.34. We have also included a graphical representation of the results in Figure 7.4.

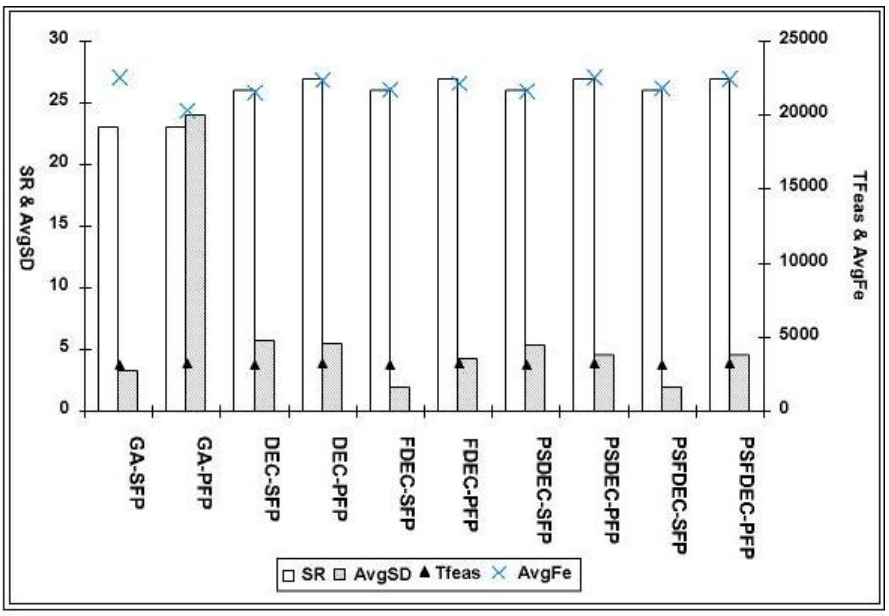

Figure 7.4: *Summary of Results for set A*

	SR	**TFeas**	**AvgFe**	**AvgSD**
GA-SFP	23	3168	22 511.9	3.27
GA-PFP	23	3300	20 286.9	24.01
DEC-SFP	26	3200	21 514.0	5.77
DEC-PFP	27	3300	22 381.6	5.48
FDEC-SFP	26	3200	21 738.2	1.9
FDEC-PFP	27	3300	22 197.8	4.3
PSDEC-SFP	26	3200	21 639.6	5.42
PSDEC-PFP	27	3300	22 511.6	4.47
PSFDEC-SFP	26	3200	21 894.4	2.01
PSFDEC-PFP	27	3300	22 434.2	4.51

Table 7.34: Summary of all results for problem set A

A comparison of the algorithms using Table 7.34 will now be presented. Firstly, when we consider the success rate of the methods, the DEC based methods using the PFP scheme prove to be the most successful. They only fail on 6 out of the 33 problems. The DEC based methods using the SFP scheme are the next most successful with a total of 26 problems for which the best known minimum was located. GA-SFP and GA-PFP methods only succeed in locating the minimum for 23 problems and thus have the lowest overall success rate. On

this criterion the DEC based methods performed better than GA.

The TFeas value for the all methods using the PFP scheme is 3300,indicating that all runs for all 33 problems located feasible points. On the other hand the methods using the SFP scheme are slightly inferior. We can conclude that the PFP scheme performed the best and was the most reliable for all methods. The DEC and GA methods were evenly rated for the PFP scheme but for the SFP scheme all new methods performed better than GA.

Next we consider the average number of fitness function evaluations. For the new methods, the SFP scheme prove to be computationally less expensive than their respective PFP counterparts. GA-PFP method however requires the least number of average fitness function evaluations while GA-SFP is the most costly with the highest average.

If we look at the accuracy (AvgSD) of the methods, the FDEC-SFP method is the most accurate followed by PSFDEC-SFP and then GA-SFP. GA-PFP method proves to be the least accurate with an AvgSD value much higher than all the other methods. Overall, the new methods again show much better performance than GA on this criterion.

An important aspect to consider is that the DEC based methods using the SFP scheme are successful on 3 additional problems (i.e problems 1, 18 and 21) than GA-SFP. If we only consider the results for problems where all methods, using SFP, were successful in locating the best known minimum we will have a total of 23 common problems (i.e. 2-9, 11-13, 15-17, 19, 20, 22-28). Similarly, the new methods using PFP where successful on 4 more problems than GA-PFP. Those problems being problem 1,18, 21 and 27. Again if we only consider the results for problems where all methods, using PFP, were successful in locating the best known minimum we will have a total of 23 common problems (i.e. 2-9, 11-17, 19, 20, 22-26, 28). It is important to note that the 23 common problems for SFP and PFP are not the same.

Using only these 23 common problems we provide a summary of the results for GA and DEC based methods. Table 7.35 gives the results for all methods using the SFP scheme on the 23 common problems and Table 7.36 gives the results for the 23 problems common for

all methods using the PFP scheme.

	TFeas	AvgFe	AvgSD
GA-SFP	2300	22 511.9	3.27
DEC-SFP	2300	20 197.98	6.52
FDEC-SFP	2300	20 362.6	2.13
PSDEC-SFP	2300	20 270.1	6.13
PSFDEC-SFP	2300	20 492.11	2.18

Table 7.35: Summary of results for 23 common problems on set A, SFP scheme

	TFeas	AvgFe	AvgSD
GA-PFP	2300	20 286.9	24.01
DEC-PFP	2300	20 253.53	6.39
FDEC-PFP	2300	20 116.17	5.02
PSDEC-PFP	2300	20 350.72	5.22
PSFDEC-PFP	2300	20 300.08	5.29

Table 7.36: Summary of results for 23 common problems on set B, PFP scheme

From Table 7.35 above we see that for the 23 common problems GA and the new methods have the same TFeas values. All new methods have lower AvgFe values than GA. Also if we compare AvgFe for these 23 problems to those of the full problem set as given in Table 7.34 we can see a significant decrease in the average number of fitness function evaluations. This shows that the 3 additional problems that the new methods were successful on resulted in a much higher AvgFe value. If we consider the AvgSD values in Table 7.35 we see that FDEC has the lowest value followed by PSFDEC and then GA. We also note that AvgSD for the DEC based methods is slightly higher for the 23 common problems than for the full problem set (see Table 7.34).

Next we look at the results for the methods using the PFP scheme. From Table 7.36 we firstly see that all algorithms have the same TFeas value. Secondly, DEC and FDEC have AvgFe values that are lower than GA while the AvgFe values for PSDEC and PSFDEC are higher than GA. All new methods have lower AvgFe values when compared to the full problem set (see Table 7.34). The same behaviour is exhibted by the methods using SFP. We do however note that the difference in the AvgFe values for all methods is fairly small. Lastly

when we look at the AvgSD values, all new methods have lower values than GA. We also note that AvgSD for the DEC based methods is slightly higher for the 23 common problems than for the full problem set (see Table 7.34).

This exercise has shown us that the DEC methods have performed better in most regards when compared to GA. We can also see that amongst the DEC based methods the overall performance of FDEC seems to be slightly superior to the other methods. Next we look at problem set B.

7.7.2 Results on problem set B

	GA		DEC		FDEC		PSDEC		PSFDEC	
np	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP
34	-314.03	-314.39	**-316.27**	**-316.27**	**-316.27**	**-316.27**	**-316.27**	**-316.27**	**-316.27**	**-316.27**
35	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**
36	**0**	**0**	**0**	**0**	**0**	**0**	0	0	0	0
37	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**
38	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**
39	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**
40	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**
41	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**
42	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**
43	0.79	0.79	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**
44	-	-	8840.31	8828.2	8832.96	**8827.23**	8894.81	8905.88	8839.28	8836.32
45	-0.41	-0.41	**-0.42**	**-0.42**	**-0.42**	**-0.42**	**-0.42**	**-0.42**	**-0.42**	**-0.42**

Table 7.37: Set B: Minimum values (min)

	GA		DEC		FDEC		PSDEC		PSFDEC	
np	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP
34	-295.43	-295.65	-307.8	-309.49	-308.36	-306.1	**-310.05**	-307.8	-308.93	-306.1
35	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**
36	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
37	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**	**-195.37**
38	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**	**-2.21**
39	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**	**0.13**
40	0.64	0.63	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**
41	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**
42	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**	**1.51**
43	0.97	0.97	**0.76**	**0.76**	0.77	**0.76**	**0.76**	**0.76**	0.77	**0.76**
44	-	-	8900.7	8909.57	8903.02	8915.2	8828.75	8839.64	8904.02	8906.09
45	**-0.41**	**-0.41**	**-0.41**	**-0.41**	**-0.41**	**-0.41**	**-0.41**	**-0.41**	**-0.41**	**-0.41**

Table 7.38: Set B: Mean values (mean)

	GA		DEC		FDEC		PSDEC		PSFDEC	
np	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP
34	13.41	12.12	20.26	18.44	19.69	21.8	17.76	20.26	19.08	21.8
35	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
36	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
37	0.01	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
38	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
39	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
40	0.02	0.02	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
41	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
42	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
43	0.13	0.11	**0**	**0**	0.04	**0**	**0**	**0**	0.04	**0**
44	-	-	42.3	46.09	47.64	50.33	43.66	47.31	42.36	43.79
45	0.01	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0.01

Table 7.39: Set B: Standard Deviation values (dev)

	GA		DEC		FDEC		PSDEC		PSFDEC	
np	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP	SFP	PFP
34	31724.1	32350.3	**25503.51**	26975.08	26187.28	26581.18	26029.7	27137.01	26174.26	26842.56
35	10332.3	**10332.3**	10351.49	10348.46	10341.39	10344.42	10377.07	10404.72	10402.5	10382.38
36	10382.8	**10362.6**	10469.66	10454.51	10450.47	10456.53	10481.89	10504.99	10466.78	10487.01
37	12524	**12382.6**	13161.31	13187.57	13162.32	13211.81	13157.11	13252.96	13175.25	13242.92
38	10291.9	10302	**10284.83**	10286.85	10285.84	10285.84	10331.63	10314.72	10333.72	10313.9
39	10312.1	10312.1	10318.16	10342.4	**10311.09**	10321.19	10357.18	10350.12	10358.33	10336.98
40	17321.5	17604.3	13936.99	**13829.93**	13799.63	14182.42	13835.87	13949.85	13921.01	14288.91
41	**10590.86**	10680.75	10954.46	11116.06	10913.05	11120.1	10958.83	11033.13	11033.71	11064.98
42	**10500.97**	10514.1	10620.15	10638.33	10635.3	10686.81	10700.91	10687.7	10714.13	10702.8
43	22007.9	20402	**16644.8**	16665	17675	17069	16771.55	16986.55	18513.29	17210.56
44	101	101	50601	50469.7	50570.7	50601	51626.63	51612.49	51624.75	51599.3
45	13665.3	13837	**12685.6**	12806.8	13018.9	12988.6	12922.71	12941.93	12920.13	18256.78

Table 7.40: Set B: Fitness Function Evaluation values (feval)

Next we summarize the results for problem set B in Table 7.41 and present a graphical representation of the results in Figure 7.5.

	SR	TFeas	AvgFe	AvgSD
GA-SFP	9	1100	11 769.53	0.0
GA-PFP	9	1100	11 813.97	0.0
DEC-SFP	11	1200	13 175.45	1.84
DEC-PFP	11	1200	13 332.0	1.68
FDEC-SFP	11	1200	13 344.12	1.79
FDEC-PFP	12	1200	16 487.24	6.01
PSDEC-SFP	11	1200	13 265.86	1.65
PSDEC-PFP	11	1200	13 414.88	1.84
PSFDEC-SFP	11	1200	13 455.74	1.74
PSFDEC-PFP	11	1200	13 920.89	1.99

Table 7.41: Summary of results on set B

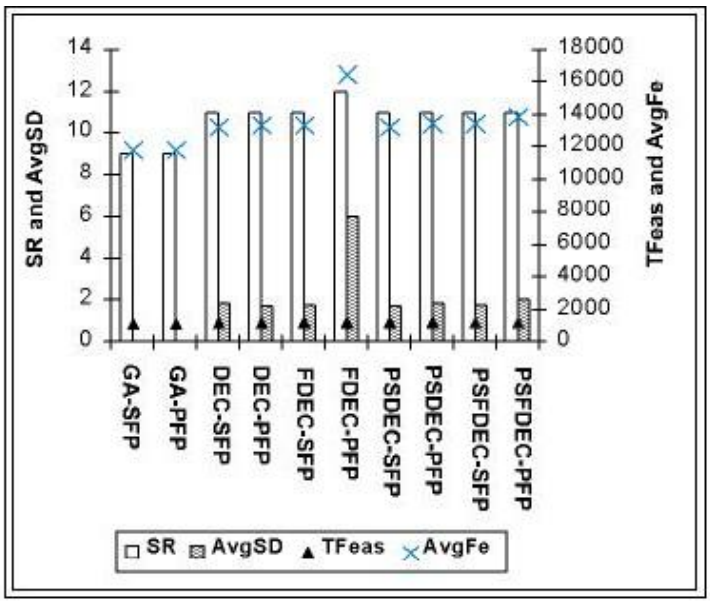

Figure 7.5: *Summary of Results for set B*

Table 7.41 shows that the FDEC-PFP method is the only method successful on all 12 problems in this set. Also other new methods still performed better than GA which was only successful on 9 out of the 12 problems. Hence the new methods have again shown their superior performance on this criterion.

The TFeas value shows that the DEC methods found feasible points for all runs, whereas GA methods had fewer feasible runs. Clearly the DEC methods performed better than GA.

We know from the results on problem set A that the AvgSD and AvgFe values for the new methods are inflated due to the additional problems the methods were successful on. Therefore we will now present the summary of results for all methods on the 9 problems that were solved by all methods. The results are presented in Table 7.42.

	SR	**TFeas**	**AvgFe**	**AvgSD**
GA-SFP	9	900	11 769.53	0.0
GA-PFP	9	900	11 813.97	0.0
DEC-SFP	9	900	11 420.07	0.0
DEC-PFP	9	900	11 445.32	0.0
FDEC-SFP	9	900	11 435.22	0.0
FDEC-PFP	9	900	11 510.97	0.0
PSDEC-SFP	9	900	11 458.13	0.0
PSDEC-PFP	9	900	11 493.35	0.0
PSFDEC-SFP	9	900	11 480.62	0.0
PSFDEC-PFP	9	900	12 119.63	0.0

Table 7.42: Summary of results for set B on 9 common problems

Table 7.42 shows that for these 9 problems all methods were equally matched with respect to SR, TFeas and AvgSD. The AvgFe values for all DEC based methods, except PSFDEC-PFP, are lower than GA. This shows that the performance of the DEC-based method is superior to GA on set B.

Chapter 8

Conclusion

Our objective in this thesis was to design a general purpose algorithm for solving constrained global optimization problems. We wanted to create a solver that could easily and effectively deal with a wide array of problems without imposing any restrictions on the problems. To achieve our objective we propose four new algorithms for constrained global optimization and conducted extensive numerical testing using two sets of test problems.

We firstly proposed the DEC algorithm, that is based on the differential evolution algorithm and reliant on the penalty function approach for constraint handling. The SFP and PFP penalty schemes were employed. We also proposed three additional algorithms that are based on DEC, namely FDEC, PSDEC and PSFDEC. These algorithms include features such as a filter set for diversification and a local technique based on PS. One of the salient features of all these algorithms is that aside from the standard parameters required for DE and the penalty coefficient in the SFP scheme no additional user input is required.

Our first phase of the testing process was to obtain suitable parameter values for the new algorithms by empirical testing. In the second phase, we tested all algorithms on two sets of test problems. The first set contains 33 test problems and the second set contains 12 problems giving a total of 45 test problems. Each algorithm was tested with both the SFP and PFP constraint handling schemes. The results for all algorithms were extremely promising with

the filter based DEC, FDEC, showing a slight dominance over the other proposed algorithms. We compared the results of the new algorithms with those of GA. Results have shown that, on average, GA is worse than even the worst performing DEC algorithm on both test sets.

When compared the new methods with other methods including GA. The comparisons have shown that the new methods are extremely reliable and efficient solvers for constrained optimization problems. The most important finding was that the DE based methods found the best known solution for more problems than GA. We hope that more research will be done to fully expose the potential of the DE algorithm in solving diverse problems.

The new algorithms introduced provide promising results. The approach we adopted in designing the algorithms is relatively new and will possibly provide new research interests. The use of filter sets and the PS algorithm for evolutionary algorithms provide many possibilities for further exploration.

Bibliography

[1] Al-Sultan, K.S. and Al-Fawzan, M.A., 1997. A tabu search Hooke and Jeeves algorithm for unconstrained optimization. *European Journal of Operations Research*, **103**, pp. 198 - 208.

[2] Ali, M.M and Kaelo, P., 2006. Numerical study of some modified differential evolution algorithms. *European Journal of Operational Research*, **169**(3), pp. 1176 - 1184.

[3] Ali, M. M and Storey, C., 1994. Topographical Multilevel Single Linkage. *Journal of Global Optimization*, **5**, pp. 349 - 358.

[4] Ali, M. M and Storey, C., 1994. Modified controlled random search algorithms. *International Journal of Computer Mathematics*, **53**, pp. 229 - 235.

[5] Ali, M.M and Torn, A., 2002. Topographical differential evolution using pre-calculated differentials. In Dzemyda G., Saltenis V. and Zilinskas A. eds. *Stochastic and Global Optimization*. London: Kluwer Academic Publishers, pp. 1 -17.

[6] Ali, M.M and Torn, A., 2004. Population set based global optimization algorithms: Some modifications and numerical studies. *Computers and Operations Research*, **11**, pp. 1703-1725.

[7] Audet, C., 2004. Convergence results for pattern search algorithms are tight. *Optimization and Engineering*, **5**(2), pp. 101-122.

[8] Audet, C and Dennis Jr, J. E., 2004. A pattern search filter method for non-linear programming withought derivatives. *SIAM Journal on Optimization*, **14**(4), pp. 980 - 1010.

[9] Chelouah, R. and Siarry, P., 2005. A hybrid continuous tabu search and Nelder-Mead simplex algorithms for global optimization of multiminima functions. *European Journal of Operations Research* **161**, 3, pp. 636 - 654.

[10] Clevenger, L. , Ferguson, L. and Hart, W.E., 2005. A Filter-Based Evolutionary Algorithm for Constrained Optimization. *Evolutionary Computation*, **13**(3), pp. 329 - 352.

[11] Coconut Benchmark Problems, [online]. *http://www.mat.univie.ac.at/ neum/glopt/coconut/Benchmark/Benchmark.html*

[12] Coello Coello C.A., 2002. 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), pp. 1245 - 1247.

[13] Coello Coello C.A., Mezura-Montesy E. and Tun-Moralesz E.I., 2004. Simple Feasibility Rules and Differential Evolution for Constrained Optimization. *Lecture Notes in Computer Science*, **2972**, pp. 707 - 716.

[14] Davis, C., 1954. Theory of positive linear dependence. *American Journal of Mathematics*, **76**(4), pp. 733 - 746.

[15] Deb, K., 2002. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, **186**, pp. 311 -338.

[16] Dennis Jr, J.E. and Torczon V., 1991. Direct search methods on parallel machines. *SIAM Journal on Optimization*, **1**(4), pp. 448-474.

[17] Dorigo, M. and Di Caro, G., 1999. The ant colony optimization metaheuristic. In D. Corne, M. Dorigo and F. Glover Eds., *New Ideas in Optimization*, Cambridge: McGraw-Hill, pp. 11- 32.

[18] Fan, H.Y. and Lampinen, J.A., 2003. A trignometric mutation operator to differential evolution. *Journal of Global Optimization*, **27**, 1(2003), pp. 105 - 129.

[19] Fletcher, R. and Layffer, S., 2002. Nonlinear Programming without a penalty function. *Mathematical Programming*, **A 91**, pp. 239 - 269.

[20] Floudas, C.A. and Pardalos, P.M., 1987. *A collection of test problems for constrained global optimization algorithms*. Berlin: Springer.

[21] Glover, F. and Laguna, M., 1997. *Tabu Search*. Dordrecht: Kluwer Academic Publisers.

[22] Goldberg, D.E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading: Addison-Wesley Publishing Company.

[23] Hamida, S.B. and Schoenauer, M. 2002. ASCHEA: New results using adaptive segregational constraint handling, *Proceedings of the Congress of Evolutionary Computation*(CEC2002), pp. 884-889.

[24] Hansen, E. and Walster G.W., 2003. *Global optimization using interval analysis,* 2^{nd} *edition*. Marcel Dekker Inc.

[25] Hedar, A.R. and Fukushima, M., 2006. Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimization. *Journal of Global Optimization*, **35**(4), pp. 521 - 549.

[26] Hooke, R. and Jeeves, T.A., 1961. 'Direct search' solution of numerical and statistical problems, *Journal of the Association of Computing Machinery*, **8**, pp. 212-229.

[27] Jones, D. S., Pertunen, C. D and Stuckman, B. E., 1993. Lipschitz optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, **79** (1), pp. 157-181.

[28] Kim, J.H. and Myung, H. , 1997. Evolutionary programming techniques for constrained optimization problems, *IEEE Transactions on Evolutionary Computation*, **1**(2) , pp. 129 - 140.

[29] Kolda , T. G., Lewis, R. M. and Torezon, V., 2003. Optimization by direct search: New perspective on classical and moderm methods. *SIAM Review*, **45**(3), pp. 385 - 482.

[30] Koziel, S. and Michalewicz, Z., 1999. Evolutionary algorithms, homomorphous mappings and constrained parameter optimization. *Evolutionary Computation*, **7**(1), pp. 19 - 44.

[31] Lampinen, J. and Zelinka, I., 1999. Mechanical Engineering design optimization by differential evolution. In D. Corne, M. Dorigo, F. Glover, Eds. *New ideas in optimization*. Cambridge: McGrawHill, pp. 127- 146.

[32] Lawler, E.L. and Wood, D.E., 1966. Branch-and-bound methods: A survey. *Operations Research*, **14**, pp. 699 - 719.

[33] Lasserre, J. B., 2001. Global Optimization of polynomials and problem of moments. *SIAM Journal of Optimization*, **II** , pp. 796 - 817.

[34] Lewis R.M. and Torczon V., 1999. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, **9**(4), pp. 1082 - 1099.

[35] Lewis R.M. and Torczon V., 2000. Pattern search methods for linearly constrained minimization, *SIAM Journal on Optimization*, **10**(3), pp. 917 - 941.

[36] Michalewicz Z. , Logan T.D. and Swaminathan S., 1994. Evolutionary operators for continuous convex parameter spaces. In Sebald A.V. and Fogel L.J. Eds. *Proceedings of the 4th Annual Conference on Evolutionary Computation*, World Scientific, pp. 84 - 97.

[37] Michalewicz Z. and Schoenauer M., 1996. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, **4**(1), pp. 1 - 32.

[38] Miettinen, K., Makela, M. and Toivanen, J., 2003. Numerical comparison of some penalty based constraint handling techniques in genetic algorithm. *Journal of Global Optimization*, **27**, pp. 427 - 446.

[39] Montes, E. M. and Coello Coello, C. A., 2003. A simple multimembered evolution strategy to solve constrained optimization problems, Technical Report EVOCINV-04-2003, Evolutionary Computation Group at CINVESTAV, Seccion de Computacion, Departa- mento de Ingenieria Electrica, CINVESTAV-IPN, Mexico D.F., Mexico.

[40] Neumaier, A., 2004. Complete search in continuous global optimization and constraint satisfaction. *Acta numerica*, Cambridge University Press, pp. 271 - 369.

[41] Nocedal, J. and Wright, S.J., 1999. Numerical Optimization. *Springer Series in Operations Research*, New York: Springer.

[42] Price K., 1999. An introduction to differential evolution. In Corne D. ,Dorigo M. and Glover F. Eds. *New Ideas in Optimization*, London: McGraw-Hill, pp. 79 - 108.

[43] Pinter , J.D., 1996. Global Optimization in action: Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications. *Nonconvex Optimization and Its Applications*, **6** .

[44] Polak E., 1971. Computational Methods in Optimization: a Unified Approach. *Mathematics in Science and Engineering*, **77**, New York: Academic Press.

[45] Polak E. and Wetter M., 2003. *Generalized Pattern Search Algorithms with Adaptive Precision Function Evaluations*, Technical Report LBNL-52629, Lawrence Berkeley National Laboratory, Berkeley, CA.

[46] Powell D and Skolnick M.M., 1993. Using genetic algorithms in engineering design optimization with non-linear constraints. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 424-430.

[47] Renders, J.M. and Bersini, H., 1994. Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways. *Proceedings of The First IEEE Conference on Evolutionary Computation*, pp. 312-317. IEEE Press, Piscataway, New Jersey.

[48] Rinnoy Kan, A. H. G and Timmer, G. T., 1987. Stochastic global optimization methods, Part I: Clustering methods. *Mathematical Programming*, **39**, pp. 27 - 56.

[49] Rinnoy Kan, A. H. G and Timmer, G. T., 1987. Stochastic global optimization methods, Part II. *Mathematical Programming*, **39**, pp. 57 - 78.

[50] Romeijn, H. E. and Smith, R. L., 1994. Simulated annealing for constrained global optimization. *Journal of Global Optimization*, **5**, pp. 101 - 126.

[51] Runarsson, T. P. and Yao, X., 2000. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, **4**(3), pp. 284 -294.

[52] Sarimveis, S. and Nikolakopoulos, A., 2005. A line up evolutionary algorithm for solving nonlinear constrained optimization problems. *Computer & Operations Research*, **32**, pp. 1499 - 1514.

[53] Smith, A. and Tate, D., 1993. Genetic optimization using a penalty function. In Forrest S. Eds. *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 499 - 503.

[54] Snyman J.A. and Fatti L.P., 1987, A multi-start global optimization algorithm with dynamic search trajectories. *Journal of Optimization Theory and Application*, **54** 1, pp. 121 - 141.

[55] Storn, R. and Price, K., 1997. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, **2**, pp. 341 - 359.

[56] Torczon V., 1997. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, **7**(1), pp. 1 - 25.

[57] Torn, A. and Zilinskas, A., 1989. *Global optimization*, Berlin: Springer-Verlage.

[58] Trelea, I.C., 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, **85**, 6, pp. 317 - 325.

[59] Wah, B.W. and Chen Y., 2001. Hybrid Constrained Simulated Annealing and Genetic Algorithms for Nonlinar Constrained Optimization. *Proc. IEEE Congress on Evolutionary Computation (CEC 01)*, pp. 925-932.

[60] Zabinsky Z.B., 2003, *Stochastic Adaptive Search for Global Optimization*, London: Kluwer Academic Publishers.

[61] Zhang, B. ,Wood G.R. and Baritompa, W.P., 1993. Multidimensional bisection: The performance and the context. *Journal of Global Optimization*, **3**(3), pp. 337-358.

Appendix A

Test Problems A

Problem 1

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_1 + x_2 + x_3, \quad \text{subject to}$$

$$-1 + 0.0025(x_4 + x_6) \leq 0$$

$$-1 + 0.0025(-x_4 + x_5 + x_7) \leq 0$$

$$-1 + 0.01(-x_5 + x_8) \leq 0$$

$$100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \leq 0$$

$$x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \leq 0$$

$$x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \leq 0$$

$$100 \leq x_1 \leq 10000$$

$$1000 \leq x_2 \leq 10000$$

$$1000 \leq x_3 \leq 10000$$

$$10 \leq x_4 \leq 1000$$

$$10 \leq x_5 \leq 1000$$

$$10 \leq x_6 \leq 1000$$

$$10 \leq x_7 \leq 1000$$

$$10 \leq x_8 \leq 1000$$

Global Solution:

- Objective Function: 7049.25
- Continuous variables: **x=**

$$(580.595, 1359.178, 5109.477, 182.125,$$
$$295.621, 217.875, 286.504, 395.621)^T$$

Problem 2

$$\min_{\mathbf{x}} f(\mathbf{x}) = 37.293239x_1 + 0.8356891x_1x_5 + 5.3578547{x_3}^2 - 40792.141, \quad \text{subject to}$$

$$-0.0022053x_3x_5 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 6.665593 \leq 0$$
$$0.0022053x_3x_5 - 0.0056858x_2x_5 - 0.0006262x_1x_4 - 85.334407 \leq 0$$
$$0.0071317x_2x_5 + 0.0021813{x_3}^2 + 0.0029955x_1x_2 - 29.48751 \leq 0$$
$$-0.0071317x_2x_5 + 0.0021813{x_3}^2 - 0.0029955x_1x_2 + 9.48751 \leq 0$$
$$0.0047026x_3x_5 + 0.0019085x_3x_4 + 0.0012547x_1x_3 - 15.699039 \leq 0$$
$$-0.0047026x_3x_5 - 0.0019085x_3x_4 - 0.0012547x_1x_3 + 10.699039 \leq 0$$
$$78 \leq x_1 \leq 102$$
$$33 \leq x_2 \leq 45$$
$$27 \leq x_3 \leq 45$$
$$27 \leq x_4 \leq 45$$
$$27 \leq x_5 \leq 45$$

Global Solution:

- Objective Function: -30665.5387
- Continuous variables: $\mathbf{x} = (78, 33, 29.9953, 45, 36.7758)^T$

Problem 3

$$\min_{\mathbf{x}} f(\mathbf{x}) = -25(x_1-2)^2-(x_2-2)^2-(x_3-1)^2-(x_4-4)^2-(x_5-1)^2-(x_6-4)^2, \quad \text{subject to}$$

$$\begin{aligned}
(x_3-3)^2 + x_4 &\geq 4 \\
(x_5-3)^2 + x_6 &\geq 4 \\
x_1 - 3x_2 &\leq 2 \\
-x_1 + x_2 &\leq 2 \\
x_1 + x_2 &\leq 6 \\
x_1 + x_2 &\geq 2 \\
0 \leq x_1 &\leq 6 \\
0 \leq x_2 &\leq 6 \\
1 \leq x_3 &\leq 5 \\
0 \leq x_4 &\leq 6 \\
1 \leq x_5 &\leq 5 \\
0 \leq x_6 &\leq 10
\end{aligned}$$

Global Solution:

- Objective Function: -310
- Continuous variables: $\mathbf{x} = (5, 1, 5, 0, 5, 10)^T$

Problem 4

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_1^{0.6} + x_2^{0.6} - 6x_1 - 4x_3 + 3x_4, \quad \text{subject to}$$

$$\begin{aligned}
x_2 - 3x_1 - 3x_3 &= 0 \\
x_1 + 2x_3 - 4 &\leq 0 \\
x_2 + 2x_4 - 4 &\leq 0
\end{aligned}$$

$$0 \leq x_1 \leq 3$$

$$0 \leq x_2 \leq 4$$

$$0 \leq x_3 \leq 2$$

$$0 \leq x_4 \leq 1$$

Global Solution:

- Objective Function: -4.5142
- Continuous variables: $\mathbf{x} = (1.333333, 4.0, 0.0, 0.0)^T$

Problem 5

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_1^{0.6} + 2x_2^{0.6} + 2x_3 - 2x_2 - x_4, \quad \text{subject to}$$

$$x_2 - 3x_1 - 3x_3 = 0$$

$$x_1 + 2x_3 - 4 \leq 0$$

$$x_2 + 2x_4 - 4 \leq 0$$

$$0 \leq x_1 \leq 3$$

$$0 \leq x_2 \leq 4$$

$$0 \leq x_3 \leq 2$$

$$0 \leq x_4 \leq 2$$

Global Solution:

- Objective Function: -3.13
- Continuous variables: $\mathbf{x} = (0, 3, 0, 1)^T$

Problem 6

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_1^{0.6} + x_2^{0.6} + x_3^{0.4} + 2x_4 + 5x_5 - 4x_3 - x_6, \quad \text{subject to}$$

$$\begin{aligned}
x_2 - 3x_1 - 3x_4 &= 0 \\
x_3 - 2x_2 - 3x_5 &= 0 \\
4x_4 - x_6 &= 0 \\
x_1 + 2x_4 - 4 &\leq 0 \\
x_2 + x_5 - 4 &\leq 0 \\
x_3 + x_6 - 6 &\leq 0 \\
0 \leq x_1 &\leq 3 \\
0 \leq x_2 &\leq 4 \\
0 \leq x_3 &\leq 4 \\
0 \leq x_4 &\leq 2 \\
0 \leq x_5 &\leq 2 \\
0 \leq x_6 &\leq 6
\end{aligned}$$

Global Solution:

- Objective Function: -11.96
- Continuous variables: $\mathbf{x} = (0.67, 2.0, 4.0, 0.0, 0.0, 0.0)^T$

Problem 7

$$\min_{\mathbf{x}} f(\mathbf{x}) = -x_1 - x_2, \quad \text{subject to}$$

$$\begin{aligned}
x_2 &\leq 2 + 2{x_1}^4 - 8{x_1}^3 + 8{x_1}^2 \\
x_2 &\leq 4{x_1}^4 - 32{x_1}^3 + 88{x_1}^2 - 96x_1 + 36
\end{aligned}$$

$$0 \leq x_1 \leq 3$$

$$0 \leq x_2 \leq 4$$

Global Solution:

- Objective Function: -5.50796
- Continuous variables: $\mathbf{x} = (2.3295, 3.17846)^T$

Problem 8

$$\min_{\mathbf{x}} f(\mathbf{x}) = -12x_1 - 7x_2 + x_2^2, \quad \text{subject to}$$

$$-2x_1^4 - x_2 + 2 = 0$$

$$0 \leq x_1 \leq 2$$

$$0 \leq x_2 \leq 3$$

Global Solution:

- Objective Function: -16.78
- Continuous variables: $\mathbf{x} = (0.7175, 1.4787)^T$

Problem 9

$$\min_{\mathbf{x}} f(\mathbf{x}) = 3x_1 + 0.000001{x_1}^3 + 2x_2 + (0.000002/3){x_2}^3, \quad \text{subject to}$$

$$x_4 - x_3 + 0.55 \geq 0$$

$$x_3 - x_4 + 0.55 \geq 0$$

$$1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

$$0 \leq x_i \leq 1200, \quad i = 1, 2$$

$$-0.55 \leq x_i \leq 0.55, \quad i = 3, 4$$

Global Solution:

- Objective Function: 5126.4981
- Continuous variables: $\mathbf{x} = (679.9453, 1026.067, 0.1188764, -0.3962336)^T$

Problem 10,32,33

$$\max_{\mathbf{x}} f(\mathbf{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4 x_i - 2\prod_{i=1}^{n} \cos^2 x_i}{\sqrt{\sum_{i=1}^{n} i{x_i}^2}} \right|, \quad \text{subject to}$$

$$\prod_{i=1}^{n} x_i \geq 0.75$$
$$\sum_{i=1}^{n} x_i \leq 7.5n$$
$$0 \leq x_i \leq 10, \quad i = 1, \ldots, n$$

Global Solution:

- Objective Function: 0.8331937

Problem 11

$$\min_{\mathbf{x}} f(\mathbf{x}) = \exp^{x_1 x_2 x_3 x_4 x_5}, \quad \text{subject to}$$

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$
$$x_2 x_3 - 5x_4 x_5 = 0$$
$$x_1^3 + x_2^3 + 1 = 0$$

$$-2.3 \leq x_1 \leq 2.3$$

$$-2.3 \leq x_2 \leq 2.3$$

$$-3.2 \leq x_3 \leq 3.2$$

$$-3.2 \leq x_4 \leq 3.2$$

$$-3.2 \leq x_5 \leq 3.2$$

Global Solution:

- Objective Function: 0.0539498473
- Continuous variables: $\mathbf{x} = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.7636450)^T$

Problem 12

$$\min_{\mathbf{x}} f(\mathbf{x}) = \left\{ \begin{array}{lll} f_1 = x_2 + 10^{-5}(x_2 - x_1)^2 - 1 & if & 0 \leq x_1 < 2 \\ f_2 = \frac{1}{27\sqrt{3}}((x_1 - 3)^2 - 9)x_2^3 & if & 2 \leq x_1 < 4 \\ f_3 = \frac{1}{3}(x_1 - 2)^3 + x_2 - \frac{11}{3} & if & 4 \leq x_1 \leq 6 \end{array} \right\}$$

subject to ,

$$\frac{x_1}{\sqrt{3}} - x_2 \geq 0$$
$$-x_1 - \sqrt{3}x_2 + 6 \geq 0$$

$$0 \leq x_1 \leq 6$$

$$x_2 \geq 0$$

Global Solution:

- Objective Function: -1.0

- Continuous variables: $\mathbf{x} = (0, 0)$ or $(4, 0)$ or $(3, \sqrt{3})$

Problem 13

$$\max_{\mathbf{x}} f(\mathbf{x}) = \frac{\sin^3 2\pi x_1 \sin 2\pi x_2}{x_1^3(x_1 + x_2)}, \quad \text{subject to}$$

$$\begin{aligned} x_1^2 - x_2 + 10 &\leq 0 \\ 1 - x_1 + (x_2 - 4)^2 &\leq 0 \\ 0 \leq x_1 &\leq 10 \\ 0 \leq x_2 &\leq 10 \end{aligned}$$

Global Solution:

- Objective Function: 0.1
- Continuous variables: $\mathbf{x} = (1.228, 4.245)^T$

Problem 14

$$\max_{\mathbf{x}} f(\mathbf{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i, \quad \text{subject to}$$

$$\begin{aligned} \prod_{i=1}^{n} x_i^2 &= 1 \\ 0 \leq x_i \leq 1, i &= 1, \ldots, n \end{aligned}$$

Global Solution:

- Objective Function: 1.0
- Continuous variables: $x_1, \ldots, x_n = (\frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}})^T$

Problem 15

$$\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^{10} x_j (c_i + ln \frac{x_j}{x_1 + \ldots + x_{10}}), \quad \text{subject to}$$

$$x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$$

$$x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$

$$x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$$

$$0.000001 \leq x_i \leq 2, i = 1, \ldots, 10$$

$\mathbf{c} = (-6.089, -17.164, -34.054, -5.519, -24.721, -14.986, -24.1, -10.708, -26.662, -22.179)^T$

Global Solution:

- Objective Function: -47.760765
- Continuous variables: **x=**

$$(.04034785, .15386976, .77497089, .00167479, .48468539,$$
$$0.00068965, 0.2826479, 0.1849179, 0.03849563, .10128126)^T$$

Problem 16

$$\min_{\mathbf{x}} f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \quad \text{subject to}$$

$$-x_1 - x_2^2 \leq 0$$

$$-x_1^2 - x_2 \leq 0$$

$$-0.5 \leq x_1 \leq 0.5$$

$$-10 \leq x_2 \leq 1$$

Global Solution:

- Objective Function: 0.25
- Continuous variables: $\mathbf{x} = (0.5, 0.25)^T$

Problem 17

$$\min_{\mathbf{x}} f(\mathbf{x}) = 0.01x_1^2 + x_2^2, \quad \text{subject to}$$

$$\begin{aligned} -x_1x_2 + 25 &\leq 0 \\ -x_1^2 - x_2^2 + 25 &\leq 0 \\ 2 \leq x_1 &\leq 50 \\ 0 \leq x_2 &\leq 50 \end{aligned}$$

Global Solution:

- Objective Function: 5.0
- Continuous variables: $\mathbf{x} = (15.8114, 1.58114)^T$

Problem 18

$$\begin{aligned} \min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 \\ +7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7, \quad \text{subject to} \end{aligned}$$

$$\begin{aligned} 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 &\geq 0 \\ 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 &\geq 0 \\ 196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 &\geq 0 \\ -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 - 11x_7 &\geq 0 \\ -10 \leq x_i \leq 10, i = 1, \ldots, 7 & \end{aligned}$$

Global Solution:

- Objective Function: 680.6300573
- Continuous variables:

  $\mathbf{x} = (2.330499, 1.951372, -0.4775414, 4.365726, -0.624487, 1.038131, 1.594227)^T$

Problem 19

$$\min_{\mathbf{x}} f(\mathbf{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} {x_i}^2 - \sum_{i=5}^{13} x_i, \quad \text{subject to}$$

$$\begin{aligned}
2x_1 + 2x_2 + x_{10} + x_{11} &\leq 10 \\
2x_1 + 2x_3 + x_{10} + x_{12} &\leq 10 \\
2x_2 + 2x_3 + x_{11} + x_{12} &\leq 10 \\
-8x_1 + x_{10} &\leq 0 \\
-8x_2 + x_{11} &\leq 0 \\
-8x_3 + x_{12} &\leq 0 \\
-2x_4 - x_5 + x_{10} &\leq 0 \\
-2x_6 - x_7 + x_{11} &\leq 0 \\
-2x_8 - x_9 + x_{12} &\leq 0 \\
0 \leq x_i \leq 1, &\quad i = 1, \ldots, 9 \\
0 \leq x_i \leq 100, &\quad i = 10, 11, 12 \\
0 \leq x_{13} &\leq 1
\end{aligned}$$

Global Solution:

- Objective Function: -15
- Continuous variables: $\mathbf{x} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)^T$

Problem 20

$$\min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3, \quad \text{subject to}$$

$$\begin{aligned}
(x_1 - 5)^2 + (x_2 - 5)^2 - 100 &\geq 0 \\
-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.82 &\geq 0 \\
13 \leq x_1 &\leq 100 \\
0 \leq x_2 &\leq 100
\end{aligned}$$

Global Solution:

- Objective Function: -6961.81381
- Continuous variables: $\mathbf{x} = (14.095, 0.84296)^T$

Problem 21

$$\begin{aligned}
\min_{\mathbf{x}} f(\mathbf{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\
+2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45,
\end{aligned}$$

subject to:

$$\begin{aligned}
105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 &\geq 0 \\
-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 &\geq 0 \\
-10x_1 + 8x_2 + 17x_7 - 2x_8 &\geq 0 \\
-x_1^2 - 2(x_2 - 2)^2 + 2x_1 x_2 - 14x_5 + 6x_6 &\geq 0 \\
8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 &\geq 0 \\
-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 &\geq 0
\end{aligned}$$

$$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0$$

$$-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0$$

$$-10 \leq x_i \leq 10, \quad i = 1, \ldots, 10$$

Global Solution:

- Objective Function: 24.3062091
- Continuous variables: **x=**

$$(2.171996, 2.363683, 8.773926, 5.095984, 0.9906548,$$
$$1.430574, 1.321644, 9.828726, 8.280092, 8.375927)^T$$

Problem 22

$$\min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2, \quad \text{subject to}$$

$$x_1^2 - x_2 \leq 0$$

$$x_1 + x_2 - 2 \leq 0$$

$$-2 \leq x_1 \leq 1$$

$$0 \leq x_2 \leq 4$$

Global Solution:

- Objective Function: 1
- Continuous variables: $\mathbf{x} = (1, 1)^T$

Problem 23

$$\min_{\mathbf{x}} f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} - 0.5\mathbf{x}^T\mathbf{Q}\mathbf{x}, \quad \text{subject to}$$

$$20x_1 + 12x_2 + 11x_3 + 7x_4 + 4x_5 \leq 40$$

$$0 \leq \mathbf{x} \leq 1$$

$$\mathbf{c} = (42, 44, 45, 47, 47.5)^T$$

$$\mathbf{Q} = 100\mathbf{I}$$

Global Solution:

- Objective Function: -17
- Continuous variables: $\mathbf{x} = (1, 1, 0, 1, 0)^T$

Problem 24

$$\min_{\mathbf{x}} f(\mathbf{x}) = -10.5x_1 - 7.5x_2 - 3.5x_3 - 2.5x_4 - 1.5x_5 \\ -0.5(x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2) - 10x_6$$

subject to:

$$6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 - 6.5 \leq 0$$

$$10x_1 + 10x_3 + x_6 - 20 \leq 0$$

$$0 \leq x_i \leq 1, \quad i = 1, \ldots, 5$$

$$0 \leq x_6 \leq 20$$

Global Solution:

- Objective Function: -213
- Continuous variables: $\mathbf{x} = (0, 1, 0, 1, 1, 20)^T$

Problem 25

$$\min_{\mathbf{x}} f(\mathbf{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i, \quad \text{subject to}$$

$$\begin{aligned}
2x_1 + 2x_2 + x_{10} + x_{11} - 10 &\leq 0 \\
2x_1 + 2x_3 + x_{10} + x_{12} - 10 &\leq 0 \\
2x_2 + 2x_3 + x_{11} + x_{12} - 10 &\leq 0 \\
-8x_1 + x_{10} &\leq 0 \\
-8x_2 + x_{11} &\leq 0 \\
-8x_3 + x_{12} &\leq 0 \\
-2x_4 - x_5 + x_{10} &\leq 0 \\
-2x_6 - x_7 + x_{11} &\leq 0 \\
-2x_8 - x_9 + x_{12} &\leq 0 \\
0 \leq x_i \leq 1, &\quad i = 1, \ldots, 9 \\
0 \leq x_i \leq 100, &\quad i = 10, 11, 12 \\
0 \leq x_{13} &\leq 1
\end{aligned}$$

Global Solution:

- Objective Function: -15
- Continuous variables: $\mathbf{x} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)^T$

Problem 26

$$\min_{x,\mathbf{y}} f(x, \mathbf{y}) = 6.5x - 0.5x^2 - y_1 - 2y_2 - 3y_3 - 2y_4 - y_5, \quad \text{subject to}$$

$$\mathbf{Az} \leq \mathbf{b}$$

$$z = (x, \mathbf{y})^T$$

$$0 \leq x \leq 1$$

$$0 \leq y_1 \leq 6$$

$$0 \leq y_2 \leq 8$$

$$0 \leq y_i \leq 1, \quad i = 3, 4$$

$$0 \leq y_5 \leq 2$$

$$\mathbf{A} = \begin{Bmatrix} 1 & 2 & 8 & 1 & 3 & 5 \\ -8 & -4 & -2 & 2 & 4 & -1 \\ 2 & 0.5 & 0.2 & -3 & -1 & -4 \\ 0.2 & 2 & 0.1 & -4 & 2 & 2 \\ -0.1 & -0.5 & 2 & 5 & -5 & 3 \end{Bmatrix}$$

$$\mathbf{b} = (16, -1, 24, 12, 3)^T$$

Global Solution:

- Objective Function: -11.005
- Continuous variables: $x = 0, \mathbf{y} = (6, 0, 1, 1, 0)^T$

Problem 27

$$\min_{\mathbf{x,y}} f(\mathbf{x,y}) = \mathbf{c}^T\mathbf{x} - 0.5\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{d}^T\mathbf{y}, \quad \text{subject to}$$

$$\mathbf{Az} \leq \mathbf{b}$$

$$z = (\mathbf{x}, \mathbf{y})^T$$

$$0 \leq \mathbf{z} \leq 1$$

$$\mathbf{A} = \left\{ \begin{array}{cccccccccc} -2 & -6 & -1 & 0 & -3 & -3 & -2 & -6 & -2 & -2 \\ 6 & -5 & 8 & -3 & 0 & 1 & 3 & 8 & 9 & -3 \\ -5 & 6 & 5 & 3 & 8 & -8 & 9 & 2 & 0 & -9 \\ 9 & 5 & 0 & -9 & 1 & -8 & 3 & -9 & -9 & -3 \\ -8 & 7 & -4 & -5 & -9 & 1 & -7 & -1 & 3 & -2 \\ -7 & -5 & -2 & 0 & -6 & -6 & -7 & -6 & 7 & 7 \\ 1 & -3 & -3 & -4 & -1 & 0 & -4 & 1 & 6 & 0 \\ 1 & -2 & 6 & 9 & 0 & -7 & 9 & -9 & -6 & 4 \\ -4 & 6 & 7 & 2 & 2 & 0 & 6 & 6 & -7 & 4 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{array} \right\}$$

$$\mathbf{b} = (-4, 22, -6, -23, -12, -3, 1, 12, 15, 9, -1)^T$$

$$\mathbf{d} = (10, 10, 10)^T$$

$$\mathbf{c} = (-20, -80, -20, -50, -60, -90, 0)^T$$

$$\mathbf{Q} = 10\mathbf{I}$$

where **I** is an identity matrix.

Global Solution:

- Objective Function: -268.0164
- Continuous variables: $\mathbf{x} = (1, 0.90755, 0, 1, 0.71509, 1, 0)^T, \mathbf{y} = (0.9168, 1, 1)^T$

Problem 28

$$\min_{\mathbf{x}} f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} - 0.5\mathbf{x}^T\mathbf{Q}\mathbf{x}, \quad \text{subject to}$$

$$\begin{aligned} \mathbf{Az} &\leq \mathbf{b} \\ \mathbf{x} &\in \mathbb{R}^{10} \end{aligned}$$

$$0 \leq \mathbf{x} \leq 1$$

$$\mathbf{A} = \begin{Bmatrix} -2 & -6 & -1 & 0 & -3 & -3 & -2 & -6 & -2 & -2 \\ 6 & -5 & 8 & -3 & 0 & 1 & 3 & 8 & 9 & -3 \\ -5 & 6 & 5 & 3 & 8 & -8 & 9 & 2 & 0 & -9 \\ 9 & 5 & 0 & -9 & 1 & -8 & 3 & -9 & -9 & -3 \\ -8 & 7 & -4 & -5 & -9 & 1 & -7 & -1 & 3 & -2 \end{Bmatrix}$$

$$\mathbf{b} = (-4, 22, -6, -23, -12)^T$$

$$\mathbf{c} = (48, 42, 48, 45, 44, 41, 47, 42, 45, 46)^T$$

$$\mathbf{Q} = 100\mathbf{I}$$

where $\mathbf{I}$ is an identity matrix.

Global Solution:

- Objective Function: -39
- Continuous variables: $\mathbf{x} = (1, 0, 0, 1, 1, 1, 0, 1, 1, 1)^T$

Problem 29,30,31

$$\min_{\mathbf{x}} f(\mathbf{x}) = -0.5 \sum_i \lambda_i (x_i - \alpha_i)^2, \quad \text{subject to}$$

$$\begin{aligned} \mathbf{Ax} &\leq \mathbf{b} \\ \mathbf{x} &\in \mathbb{R}^{20} \end{aligned}$$

$$0 \leq x_i \leq 40, i = 1, \ldots, 20$$

$$\mathbf{A}^T = \left\{ \begin{array}{cccccccccc} -3 & 7 & 0 & -5 & 1 & 1 & 0 & 2 & -1 & 1 \\ 7 & 0 & -5 & 1 & 1 & 0 & 2 & -1 & -1 & 1 \\ 0 & -5 & 1 & 1 & 0 & 2 & -1 & -1 & -9 & 1 \\ -5 & 1 & 1 & 0 & 2 & -1 & -1 & -9 & 3 & 1 \\ 1 & 1 & 0 & 2 & -1 & -1 & -9 & 3 & 5 & 1 \\ 1 & 0 & 2 & -1 & -1 & -9 & 3 & 5 & 0 & 1 \\ 0 & 2 & -1 & -1 & -9 & 3 & 5 & 0 & 0 & 1 \\ 2 & -1 & -1 & -9 & 3 & 5 & 0 & 0 & 1 & 1 \\ -1 & -1 & -9 & 3 & 5 & 0 & 0 & 1 & 7 & 1 \\ -1 & -9 & 3 & 5 & 0 & 0 & 1 & 7 & -7 & 1 \\ -9 & 3 & 5 & 0 & 0 & 1 & 7 & -7 & -4 & 1 \\ 3 & 5 & 0 & 0 & 1 & 7 & -7 & -4 & -6 & 1 \\ 5 & 0 & 0 & 1 & 7 & -7 & -4 & -6 & -3 & 1 \\ 0 & 0 & 1 & 7 & -7 & -4 & -6 & -3 & 7 & 1 \\ 0 & 1 & 7 & -7 & -4 & -6 & -3 & 7 & 0 & 1 \\ 1 & 7 & -7 & -4 & -6 & -3 & 7 & 0 & -5 & 1 \\ 7 & -7 & -4 & -6 & -3 & 7 & 0 & -5 & 1 & 1 \\ -7 & -4 & -6 & -3 & 7 & 0 & -5 & 1 & 1 & 1 \\ -4 & -6 & -3 & 7 & 0 & -5 & 1 & 1 & 0 & 1 \\ -6 & -3 & 7 & 0 & -5 & 1 & 1 & 0 & 2 & 1 \end{array} \right\}$$

$$\mathbf{b} = (-5, 2, -1, -3, 5, 4, -1, 0, 9, 40)^T$$

Global Solution:

Problem 29: $\lambda_i = 1, \alpha_i = 1$

- Objective Function: -394.7506
- Continuous variables:

  $\mathbf{x} = (0, 0, 28.8024, 0, 0, 4.1792, 0, 0, 0, 0, 0, 0, 0, 0, 0.6188, 4.0933, 0, 2.3064, 0, 0)^T$

Problem 30: $\lambda_i = 1, \alpha_i = -5$

- Objective Function: -884.75058
- Continuous variables:

  $\mathbf{x} = (0, 0, 28.8024, 0, 0, 4.1792, 0, 0, 0, 0, 0, 0, 0, 0.6188, 4.0933, 0, 2.3064, 0, 0)^T$

Problem 31: $\lambda_i = 20, \alpha_i = 0$

- Objective Function: -8695.01193
- Continuous variables:

  $\mathbf{x} = (0, 0, 28.8024, 0, 0, 4.1792, 0, 0, 0, 0, 0, 0, 0, 0.6188, 4.0933, 0, 2.3064, 0, 0)^T$

Appendix B

Test Problems B

Problem 34

$$\min_{\mathbf{x}} f(\mathbf{x}) = -(0.0204 + 0.0607x_5^2)x_1x_4(x_1 + x_2 + x_3) - (0.0187 + 0.0437x_6^2)x_2x_3(x_1 + 1.57x_2 + x_4),$$

subject to:

$$\frac{2070}{x_1x_2x_3x_4x_5x_6} - 1 \leq 0$$

$$0.00062x_1x_4x_5^2(x_1 + x_2 + x_3) + 0.00058x_2x_3x_6^2(x_1 + 1.57x_2 + x_4) - 1 \leq 0$$

$$0 \leq x_1 \leq 10$$

$$0 \leq x_2 \leq 10$$

$$0 \leq x_3 \leq 15$$

$$0 \leq x_4 \leq 15$$

$$0 \leq x_5 \leq 1$$

$$0 \leq x_6 \leq 1$$

Global Solution:

- Objective Function: -316.27
- Continuous variables: $\mathbf{x} = (10, 10, 15, 4.609, 0.78511, 0.3814)^T$

Problem 35

$$\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^{5} \frac{1}{a_i(\mathbf{x} - p_i)(\mathbf{x} - p_i) + c_i}, \quad \text{subject to}$$

$$x_1 + x_2 - 5 \leq 0$$

$$x_1 - x_2^2 \leq 0$$

$$5x_1^3 - \frac{8}{5}x_2^2 \leq 0$$

$$-3 \leq x_1 \leq 10$$

$$-4 \leq x_2 \leq 7$$

i	a_i	p_i		c_i
1	0.5	0	5	0.125
2	0.25	2	5	0.25
3	1	3	2	0.1
4	$\frac{1}{12}$	4	4	0.2
5	2	5	1	$\frac{1}{12}$

Global Solution:

- Objective Function: 0.18301
- Continuous variables: $\mathbf{x} = (-3, -4)^T$

Problem 36

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_1^2 + x_2^2, \quad \text{subject to}$$

$$x_1 + x_2 - 2 \leq 0$$

$$x_1^2 - x_2 \leq 0$$

$$-3 \leq x_1 \leq 2$$

$$0 \leq x_2 \leq 5$$

Global Solution:

- Objective Function: 0.0
- Continuous variables: $\mathbf{x} = (0, 0)^T$

Problem 37

$$\min_{\mathbf{x}} f(\mathbf{x}) = -(x_2 - 1.275x_1^2 + 5x_1 - 6)^2 - 10(1 - \frac{1}{8\pi})\cos(\pi x_1) - 10, \quad \text{subject to}$$

$$-\pi x_1 - x_2 \leq 0$$

$$-\pi^2 x_1^2 + 4x_2 \leq 0$$

$$-1.5 \leq x_1 \leq 3.5$$

$$0 \leq x_2 \leq 15$$

Global Solution:

- Objective Function: -195.37
- Continuous variables: $\mathbf{x} = (2.4656, 15)^T$

Problem 38

$$\min_{\mathbf{x}} f(\mathbf{x}) = -2x_1 - 6x_2 + x_1^3 + 8x_2^2, \quad \text{subject to}$$

$$x_1 + 6x_2 - 6 \leq 0$$

$$5x_1 + 4x_2 - 10 \leq 0$$

$$0 \leq x_1 \leq 2$$

$$0 \leq x_2 \leq 1$$

Global Solution:

- Objective Function: -2.2137
- Continuous variables: $\mathbf{x} = (0.8165, 0.375)^T$

Problem 39

$$\min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 0.75)^2 + (0.5x_2 - 0.75)^2, \quad \text{subject to}$$

$$\begin{aligned} x_1 + 0.5x_2 - 1 &\leq 0 \\ 0 \leq x_1 &\leq 1 \\ 0 \leq x_2 &\leq 2 \end{aligned}$$

Global Solution:

- Objective Function: 0.125
- Continuous variables: $\mathbf{x} = (0.5, 1)^T$

Problem 40 (madsen)

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_3, \quad \text{subject to}$$

$$\begin{aligned} -\cos x_2 + x_3 &\leq 0 \\ x_1^2 + x_2^2 + x_1 x_2 - x_3 &\leq 0 \\ -\sin x_1 - x_3 &\leq 0 \\ -(x_3 + x_1^2 + x_2^2 + x_1 x_2) &\leq 0 \\ \sin x_1 - x_3 &\leq 0 \\ -100 \leq x_i &\leq 100, \quad i = 1, \ldots, 3 \end{aligned}$$

Global Solution:

- Objective Function: 0.6164
- Continuous variables: $\mathbf{x} = (0.453275, -0.906592, 0.616432)^T$

Problem 41 (alsotame)

$$\min_{\mathbf{x}} f(\mathbf{x}) = \exp(x_1 - 2x_2), \quad \text{subject to}$$

$$\begin{aligned} \sin(-x_1 + x_2 - 1) &= 0 \\ -2 \leq x_1 &\leq 2 \\ -1.5 \leq x_2 &\leq 1.5 \end{aligned}$$

Global Solution:

- Objective Function: 0.0821
- Continuous variables: $\mathbf{x} = (0.5, 1.5)^T$

Problem 42 (twobars)

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_1 \sqrt[2]{1 + x_2^2}, \quad \text{subject to}$$

$$\begin{aligned} 0.124\sqrt[2]{(1 + x_2^2)} \times (\frac{8}{x_1} + \frac{1}{x_1 x_2}) - 1 &\leq 0 \\ 0.124\sqrt[2]{(1 + x_2^2)} \times (\frac{8}{x_1} - \frac{1}{x_1 x_2}) - 1 &\leq 0 \\ 0.2 \leq x_1 &\leq 4 \\ 0.1 \leq x_2 &\leq 1.6 \end{aligned}$$

Global Solution:

- Objective Function: 1.5087
- Continuous variables: $\mathbf{x} = (1.41163, 0.377072)^T$

Problem 43 (synthes1)

$$\min_{\mathbf{x}} f(\mathbf{x}) = -18\log(x_2+1)-19.2\log(x_1-x_2+1)+5x_4+6x_5+8x_6+10x_1-7x_3+10, \quad \text{subject to}$$

$$\begin{aligned}
-(0.8\log(x_2+1)+0.96\log(x_1-x_2+1)-0.8x_3) &\leq 0\\
-(\log(x_2+1)+1.2\log(x_1-x_2+1)-x_3-2x_6+2) &\leq 0\\
x_2-x_1 &\leq 0\\
x_2-2x_4 &\leq 0\\
-x_2+x_1-2x_5 &\leq 0\\
x_4+x_5-1 &\leq 0\\
0 \leq x_1 &\leq 2\\
0 \leq x_2 &\leq 2\\
0 \leq x_3 &\leq 1\\
0 \leq x_4 &\leq 1\\
0 \leq x_5 &\leq 1\\
0 \leq x_6 &\leq 1
\end{aligned}$$

Global Solution:

- Objective Function: 0.7593

- Continuous variables: $\mathbf{x} = (1.1465150499, 0.5465962726, 10, 0.2732981363, 0.2999593887, 0)^T$

Problem 44 (hs087)

$$\min_{\mathbf{x}} f(\mathbf{x}) = 30x_7 + 31x_8 + 28x_9 + 29x_{10} + 30x_{11}, \quad \text{subject to}$$

$$\begin{aligned}
x_5 - x_9 - x_{10} - x_{11} &= 0 \\
x_4 - x_7 - x_8 &= 0 \\
-0.007629 \sin(-x_3 + 1.4847699) x_1 x_2 + 0.006895843 x_1^2 + 200 &= 0 \\
0.007629 \sin(x_3 + 1.4847699) x_1 x_2 + x_6 - 0.00689584 \times x_2^2 &= 0 \\
0.007629 \cos(-x_3 + 1.4847699) x_1 x_2 + x_4 - 0.0006565 \times x_1^2 - 300 &= 0 \\
0.007629 \cos(x_3 + 1.4847699) \times x_1 x_2 + x_5 - 0.0006565 x_2^2 &= 0
\end{aligned}$$

$$\begin{aligned}
340 \leq x_1 &\leq 420 \\
340 \leq x_2 &\leq 420 \\
0 \leq x_3 &\leq 0.52359999999999995 \\
0 \leq x_4 &\leq 400 \\
0 \leq x_5 &\leq 1000 \\
-1000 \leq x_6 &\leq 1000 \\
-300 \leq x_7 &\leq 300 \\
0 \leq x_8 &\leq 1000 \\
-100 \leq x_9 &\leq 100 \\
0 \leq x_1 0 &\leq 100 \\
-100 \leq x_1 1 &\leq 1000
\end{aligned}$$

Global Solution:

- Objective Function: 8827.5977
- Continuous variables: **x=**

$$(373.830727085, 420, 0.1532919640, 107.8119257491, 196.3186193947,$$
$$21.3071347941, 107.8119257491, 0, 100, 96.3186193947, 0)^T$$

Problem 45 (ex 8.1.1.)

$$\min_{\mathbf{x}} f(\mathbf{x}) = -x_4, \quad \text{subject to}$$

$$0.09755988x_1x_5 + x_1 - 1 \leq 0$$
$$0.0965842812x_2x_6 + x_2 - x_1 \leq 0$$
$$0.0391908x_3x_5 + x_3 + x_1 - 1 \leq 0$$
$$0.03527172x_4x_6 + x_4 - x_1 + x_2 - x_3 \leq 0$$
$$\sqrt[2]{x_5} + \sqrt[2]{x_6} - 4 \leq 0$$
$$0 \leq x_1 \leq 1$$
$$0 \leq x_2 \leq 1$$
$$0 \leq x_3 \leq 1$$
$$0 \leq x_4 \leq 1$$
$$0.00001 \leq x_5 \leq 16$$
$$0.00001 \leq x_6 \leq 16$$

Global Solution:

- Objective Function: -0.388811
- Continuous variables: $\mathbf{x} = (0.771516, 0.516992, 0.204192, 0.388811, 3.03557, 5.09726)^T$