

ABSTRACT

RADHAKRISHNAN, ALAMELU. Evolutionary Algorithms for Multiobjective Optimization with Applications in Portfolio Optimization. (Under the supervision of Dr. Negash Medhin.)

Multiobjective optimization (MO) is the problem of maximizing/minimizing a set of nonlinear objective functions (modeling several performance criteria) subject to a set of nonlinear constraints (modeling availability of resources). The MO problem has several applications in science, engineering, finance, etc. It is normally not possible to find an optimal solution in MO, since the various objective functions in the problem are usually in conflict with each other. Therefore, the objective in MO is to find the *Pareto front* of efficient solutions that provide a tradeoff between the various objectives.

Classical techniques assign weights to the various objectives in the MO problem, and solve the resulting single objective problem using standard algorithms for nonlinear optimization. Moreover, these techniques only compute a single solution to the problem forcing the decision maker to miss out on other desirable solutions in the MO problem.

We investigate the use of evolutionary algorithms to solve MO problems in this thesis. Unlike classical methods, evolutionary strategies directly solve the MO problem to find the Pareto front. These algorithms use probabilistic rules to search for solutions and are very efficient in solving medium sized MO problems. We use evolutionary algorithms to compute the *efficient frontier* in the classical Markowitz mean-variance optimization problem in finance, and illustrate our results on an example.

EVOLUTIONARY ALGORITHMS FOR MULTIOBJECTIVE OPTIMIZATION WITH APPLICATIONS IN PORTFOLIO OPTIMIZATION

by

ALAMELU RADHAKRISHNAN

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

OPERATIONS RESEARCH

Raleigh, North Carolina

March 27, 2007

APPROVED BY

Dr. Jeffrey Scroggs

Dr. Salah Elmaghraby

Dr. Negash Medhin
(Chair of Advisory Committee)

BIOGRAPHY

Alamelu Radhakrishnan is from India. She is currently pursuing a Master's in Operations Research at North Carolina State University and will graduate in Spring 2007. She received her undergraduate degree in Electrical Engineering from University of Madras, Chennai, India and a M.S. in Electrical Engineering from Polytechnic University, Brooklyn, New York.

ACKNOWLEDGEMENTS

I would like to thank my committee for their advice, guidance and feedback that they provided me with throughout this thesis. I would like to thank my parents and my sister for their love and support that they have given me all my life. Finally, I would like to thank my husband for his love, support, help and encouragement without which this thesis as well as the Master's degree would not have been possible.

Contents

List of Tables	v
List of Figures	vi
1 Multiobjective Optimization	1
1.1 Introduction	1
1.2 Mathematical formulation of a multiobjective optimization problem	2
1.3 Example: The portfolio selection problem in finance	3
1.4 Pareto Optimality	5
1.5 Methods to solve multiobjective optimization problems	10
1.5.1 Classical Methods	10
1.5.2 Evolutionary Algorithms	11
1.6 Contribution of the thesis	13
2 Differential Evolution: An Evolutionary Algorithm for Multiobjective Opti-	
mization	14
2.1 Introduction	14
2.2 Algorithm for Differential Evolution	15
2.3 Features of differential evolution	17
2.4 Computational Results	19
2.5 Conclusions and Future work	25
Bibliography	27

List of Tables

2.1	Total Returns for four assets between 1960 and 2003 (Cornuejols and Tütüncü [4]).	21
2.2	Rates of Return for the four assets.	22
2.3	Geometric mean for the four assets.	23
2.4	Covariance between the four assets.	23
2.5	Efficient Portfolios of the four assets for different rates of return.	25

List of Figures

1.1	A plot of 200 Pareto optimal solutions in the decision variable space (top) and objective function space (bottom) for (1.7).	8
1.2	A plot of 200 Pareto optimal solutions in the decision variable space (top) and objective function space (bottom) for (1.8).	9
2.1	Efficient Frontier: standard deviations versus rate of returns for (2.10).	24

Chapter 1

Multiobjective Optimization

1.1 Introduction

Optimization is the process of finding solutions that minimize or maximize a set of objective functions subject to constraints. When multiple objectives are present, the optimization problem is called a *multiobjective optimization* (MO) problem. Multiobjective optimization (Ehrgott [6]), Miettinen [17], Deb [5], Coello Coello et al. [2]) is also referred to as multi-criteria optimization, multi-performance or vector optimization (Jahn [10]).

An MO problem can be formally defined as finding (Osyczka [20]) *a vector of decision variables that satisfies constraints and optimizes a vector function whose elements represent the objective functions.*

The objective functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term optimize means finding a solution which would give the values of all the objective functions acceptable to the decision maker.

It is usually not possible to find an optimal solution in a multiobjective optimization problem. This is because the multiple objective functions that are present often conflict each other and it is impossible to optimize all the objective functions at the same time. Instead, a set of solutions called *best solutions* providing a trade off between the objective functions can be found. The solution that is most suited to a particular application is chosen from the set of

best solutions by the decision maker.

One of the first applications of multiobjective optimization was in economics to solve public investment problems in the 1960s (Cohon and Marks [3]). Other applications arise in control (Zadeh [30]), and water resource planning (Marglin [15]). Multiobjective optimization is also frequently used in engineering, science, industry and finance (Coello Coello et al. [2] and Deb [5]).

This chapter is organized as follows: Section 1.2 provides a mathematical definition of a multiobjective optimization problem. Section 1.3 provides an application of multiobjective optimization in solving the portfolio selection problem in finance. Section 1.4 discusses an important concept in multiobjective optimization called *Pareto optimality*. Section 1.5 briefly describes different methods including classical methods and evolutionary algorithms for solving multiobjective optimization problems. Finally, Section 1.6 describes our contribution in the thesis.

1.2 Mathematical formulation of a multiobjective optimization problem

Consider the problem

$$\begin{aligned} \min \quad & f(x) \\ \text{such that} \quad & g_j(x) \leq 0, \quad j = 1, \dots, m \\ & h_l(x) = 0, \quad l = 1, \dots, p. \end{aligned} \tag{1.1}$$

The vector $x \in \mathbb{R}^n$ contains the decision variables in (1.1). The set $S = \{x \in \mathbb{R}^n \mid g_j(x) \leq 0, \quad j = 1, \dots, m, \quad h_l(x) = 0, \quad l = 1, \dots, p\}$ is the feasible region of (1.1) and depicts constraints such as the limited availability of resources in the problem. The mapping $f : \mathbb{R}^n \rightarrow \mathbb{R}^k$ defined by $f(x) = (f_1(x), \dots, f_k(x))^T$ contains the k objective functions (possibly nonlinear) of (1.1). We define the *feasible objective region* Z as the image of the feasible region S under the mapping f , i.e., $Z = \{y \in \mathbb{R}^k \mid y_i = f_i(x), \quad i = 1, \dots, k, \forall x \in S\}$. We assume that all the objective functions in $f(x)$ are being minimized. If an objective function $f_i(x)$ is to be maximized, it is

equivalent to minimizing the function $-f_i(x)$.

It is important to distinguish between the constraint space S and the objective function space Z in multiobjective optimization. The set Z plays an important role in the concept of *Pareto optimality* in multiobjective optimization, and we discuss this in Section 1.4.

1.3 Example: The portfolio selection problem in finance

Consider the classic portfolio selection problem in finance where our aim is to find an optimal portfolio of securities (stocks, bonds, etc.) that provides a tradeoff between the expected return and the risk involved. An investor wishes to invest certain amount of money in n securities, $S_1, \dots, S_n$. Each security S_i has a random return whose expected value and standard deviation are given by μ_i and σ_i respectively. Moreover, the correlation coefficient of the returns of two securities S_i and S_j is denoted by ρ_{ij} . Given this information, the $n \times n$ symmetric positive semidefinite *covariance* matrix is given by $\Sigma = (\sigma_{ij})$, where $\sigma_{ii} = \sigma_i^2$, $i = 1, \dots, n$ and $\sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$, $\forall i \neq j$. If x_i denotes the proportion of the total money invested in security S_i , the expected return $E[x]$, and the variance $\text{Var}[x]$ of the resulting portfolio $x = (x_1, \dots, x_n)$ are given by

$$E[x] = \mu_1 x_1 + \dots + \mu_n x_n = \mu^T x, \quad (1.2)$$

and

$$\text{Var}[x] = \sum_{i,j=1}^n \rho_{ij} \sigma_i \sigma_j x_i x_j = x^T \Sigma x. \quad (1.3)$$

The set of feasible portfolios is given by $X = \{x \in \mathbb{R}^n : Ax = b, Cx \geq d\}$. The set X includes the following constraints: $\sum_{i=1}^n x_i = 1$ indicating that the available capital is invested in the n securities, and $x_i \geq 0$ depicting short sales restrictions (where the investor cannot sell securities that he/she does not own). Other constraints in X include diversification (that imposes a limit on the amount allotted to a particular security or the securities of a sector) and transaction costs (Cornuejols and Tütüncü [4]).

Our aim is to find a feasible portfolio x that maximizes the return while minimizing the

variance (risk). Such a portfolio is called an *efficient portfolio*. An efficient portfolio is also defined as a portfolio that has minimum variance among all portfolios that guarantee a certain value of expected return or the one that has the maximum return over all portfolios whose variances are less than a prespecified value. The set of all the efficient portfolios form a curve in 2D space (standard deviation vs expected return), also known as the *efficient frontier*.

The portfolio selection problem is also known in the literature as *Markowitz's portfolio optimization* problem (Markowitz [16]) or the *mean-variance optimization* (MVO) problem. The efficient frontier can be calculated in three different ways (Cornuejols and Tütüncü [4]):

1. **Minimize the variance:** In this scheme, one produces the set of efficient portfolios by solving a quadratic programming (QP) model given by

$$\begin{aligned}
\min \quad & \frac{1}{2}x^T \Sigma x \\
\text{s.t.} \quad & \mu^T x \geq R \\
& Ax = b \\
& Cx \geq d
\end{aligned} \tag{1.4}$$

for values of R ranging between R_{min} and R_{max} . The first constraint in (1.4) indicates that the expected return should at least meet the target value R . The QPs are convex and can be solved efficiently using interior point methods (Cornuejols and Tütüncü [4]).

2. **Maximize the return:** In this scheme, one produces the set of efficient portfolios by solving a quadratically constrained model given by

$$\begin{aligned}
\max \quad & \mu^T x \\
\text{s.t.} \quad & \frac{1}{2}x^T \Sigma x \leq \sigma^2 \\
& Ax = b \\
& Cx \geq d
\end{aligned} \tag{1.5}$$

for values of σ^2 ranging between σ_{min}^2 and σ_{max}^2 . The first constraint in (1.5) indicates that the variance (risk) should be less than the target value σ^2 . The above model can

once again be solved using conic programming and interior point methods (Cornuejols and Tütüncü [4]).

3. **Minimize the variance and maximize the return simultaneously:** In the thesis, we will solve the portfolio optimization as the following multiobjective optimization problem

$$\begin{aligned} \min \quad & \begin{bmatrix} \frac{1}{2}x^T \Sigma x \\ -\mu^T x \end{bmatrix} \\ \text{s.t.} \quad & Ax = b \\ & Cx \geq d. \end{aligned} \tag{1.6}$$

The two objectives in (1.6) minimize the risk and to maximize the return over the set of feasible portfolios. Maximizing $\mu^T x$ is equivalent to minimizing $-\mu^T x$. Intuitively, it is clear that the expected return and the risk are conflicting objectives, since one has to take a large risk to produce a large return. The set of efficient (Pareto) solutions to the multiobjective optimization problem (1.6) form the efficient portfolio. We will discuss our numerical results in solving a portfolio selection problem in Section 2.4.

1.4 Pareto Optimality

In this section, we introduce an important concept in multiobjective optimization called *Pareto optimality* (Miettinen [17], Deb [5]). In multiobjective optimization, we usually deal with conflicting objectives, and it is not possible to find a solution that minimizes all the objectives simultaneously. However, a set of solutions providing a *tradeoff* between the objectives can be found. These are solutions where it is impossible to improve the value of an objective function without at least worsening one of the other objective functions. This concept is called Pareto optimality and was first introduced by Edgeworth in 1881. However, this is named after the French-Italian economist and sociologist Vilfredo Pareto who further developed this concept (see Pareto [21], [22]). Therefore, Pareto optimality is also sometimes referred to as *Edgeworth-Pareto optimality*.

Definition 1 A decision vector $x^* \in S$ is Pareto optimal if there is no other decision vector $x \in S$ such that $f_i(x) \leq f_i(x^*)$ for all $i = 1, \dots, k$ and $f_j(x) < f_j(x^*)$ for at least one index j .

Pareto optimality can also be defined in terms of the objective function space Z as below:

Definition 2 An objective vector $z^* \in Z$ is Pareto optimal if there is no other objective vector $z \in Z$ such that $z_i \leq z_i^*$ for all $i = 1, \dots, k$ and $z_j < z_j^*$ for at least one index j , i.e., z^* is Pareto optimal if the decision vector corresponding to it is Pareto optimal.

The above concept of Pareto optimality is sometimes referred to as global Pareto optimality or strongly efficient solutions. The other related concepts are locally Pareto optimal, and weakly Pareto optimal solutions. They can be defined as follows:

Definition 3 A decision vector $x^* \in S$ is weakly Pareto optimal if there is no other decision vector $x \in S$ such that $f_i(x) < f_i(x^*)$ for all $i = 1, \dots, k$.

Definition 4 A decision vector $x^* \in S$ is locally Pareto optimal if there exists $\delta > 0$ such that x^* is Pareto optimal in $S \cap B(x^*, \delta)$ where $B(x^*, \delta)$ is a suitable neighborhood of x^* of radius δ .

Local Pareto optimality and weak Pareto optimality can also be defined in terms of the objective function space (Miettinen [17]).

The Pareto optimal solutions or the strongly efficient solutions form a hypersurface known as the *Pareto front* in the objective function space Z . These solutions represent tradeoffs from which the decision maker picks the desired solution. The Pareto optimal solutions are also referred to as *non-dominated solutions* since it is not possible to improve an objective without worsening at least another objective. Consider the following multiobjective problem

$$\begin{aligned} \min \quad & \begin{bmatrix} x_0^2 + x_1 \\ x_0 + x_1^2 \end{bmatrix} \\ \text{s.t.} \quad & -10 \leq x_0 \leq 10 \\ & -10 \leq x_1 \leq 10. \end{aligned} \tag{1.7}$$

Figure 1.1 shows the plot of 200 Pareto optimal solutions in the decision variable space (top) and objective function space (bottom) for the multiobjective optimization problem (1.7). As seen in Figure 1.1 (bottom), the Pareto optimal solutions form a Pareto front in the objective function space. The Pareto front divides the feasible and non optimal solutions from the infeasible solutions. The Pareto optimal solutions in Figure 1.1 form a front in the decision variable space. The formation of a front in the decision variable space depends on the mapping f between the Pareto optimal solutions in the objective function space and the decision variable space. Consider another multiobjective problem

$$\begin{aligned} \min \quad & \begin{bmatrix} |6 - \sqrt{x_0^2 + x_1^2}| \\ |8 - \sqrt{x_0^2 + x_1^2}| \end{bmatrix} \\ \text{s.t.} \quad & -10 \leq x_0 \leq 10 \\ & -10 \leq x_1 \leq 10. \end{aligned} \tag{1.8}$$

Figure 1.2 shows the plot of 200 Pareto optimal solutions in the decision variable space (top) and objective function space (bottom) for the problem (1.8). Figure 1.2 (top) illustrates that the Pareto optimal solutions do not always form a front in the decision variable space. Here, the non dominated solutions are either on or between the two circles. Any point outside the ring formed by the two concentric circles can be dominated by at least one point on or between the circles. It is worth mentioning that the Pareto front is not always continuous, it can be convex or concave or consist of sections that are concave and convex. If the objective functions are not conflicting, then the Pareto front consists of just a single point corresponding to the optimal solution. The neighboring points in the Pareto front are not necessarily the neighboring points in the variable space. In a multiobjective optimization problem, it is not realistically possible to find all the points on the Pareto front. It is more reasonable to find a small set of Pareto optimal points that approximates the true Pareto front.

Figure 1.1: A plot of 200 Pareto optimal solutions in the decision variable space (top) and objective function space (bottom) for (1.7).

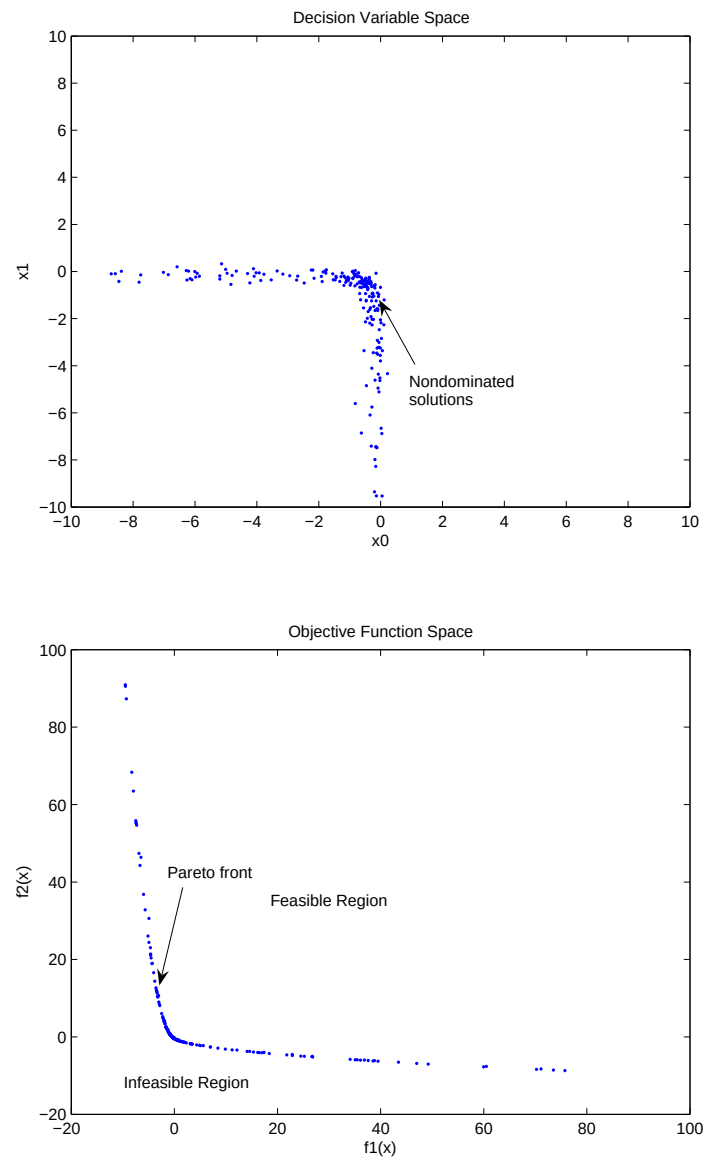
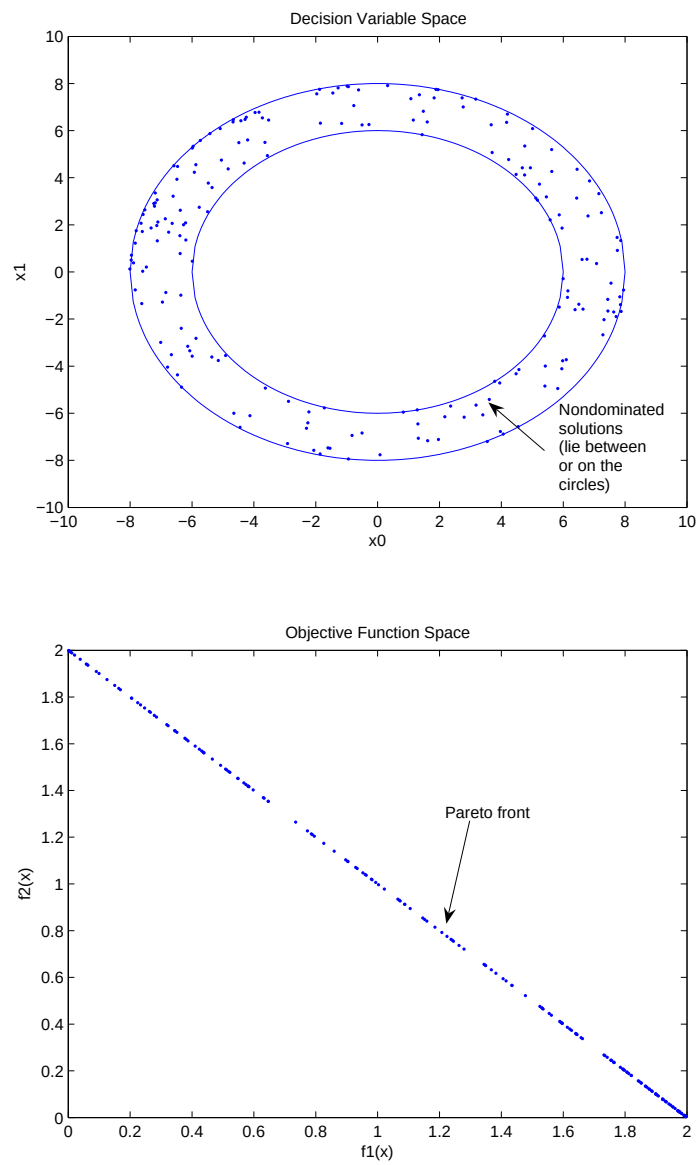


Figure 1.2: A plot of 200 Pareto optimal solutions in the decision variable space (top) and objective function space (bottom) for (1.8).



1.5 Methods to solve multiobjective optimization problems

The methods that are used to solve multiobjective optimization problems can be broadly classified as

1. Classical Methods
2. Evolutionary Algorithms.

1.5.1 Classical Methods

In classical methods, the various objective functions of the multiobjective optimization problem are combined together in a single objective function

$$\hat{f}(x) = \sum_{i=1}^k w_i f_i(x), \quad (1.9)$$

where $w_i > 0$ are appropriate weights. The single objective function $\hat{f}(x)$ is minimized over the feasible set S using traditional nonlinear optimization techniques for a single objective function. Classical methods are also classified as *a priori* or *progressive* techniques based on when the weights are assigned to the objective functions as follows:

1. **A priori:** In this method, the weights are assigned to the objective functions before optimization is performed. One of the requirements of this method is that the order of importance of the objectives has to be known ahead of time. Once the weights are chosen, they are fixed throughout the optimization procedure. Setting the weights before optimization omits desirable solutions from the model. For the assigned weights to be effective, the objective functions need to be normalized to factor in their different dynamic ranges. This is not an easy task because it requires the knowledge of the extreme values of the objective functions. These drawbacks make a priori preference methods (Miettinen [17]) very difficult to use.
2. **Progressive:** In this method, the weights are updated periodically by the decision maker based on the current solution (refer Miettinen [17]) in the optimization process. This

method is better than the a priori technique because corrections are made using the information obtained during optimization. However, prior knowledge of the problem is often required to define a scheme of preference to bias the search so that the decision maker's biases do not lead to undesirable solutions.

To summarize, one of the main shortcomings of classical methods is that some prior knowledge of the problem is required to assign reasonable weights. In classical methods, it is not possible to find multiple solutions in a single run and also not possible to find all the Pareto optimal solutions. This causes the decision maker to miss out other desirable solutions to a problem. However, these algorithms are known to converge to a Pareto optimal solution of the multiobjective problem (Stewart [29]). For more details on classical methods, we refer the reader to Deb [5], Coello Coello et al. [2], and Miettinen [17].

1.5.2 Evolutionary Algorithms

The classical methods are extensively being replaced by evolutionary algorithms, that are based on Darwin's theory of evolution. Evolutionary algorithms include evolutionary strategies (Rechenberg [27] and Schwefel [28]), genetic algorithms (Holland [9] and Goldberg [8]), and differential evolution (Price and Storn [24], [25]).

All evolutionary algorithms aim to improve the existing solution using the techniques of *recombination*, *mutation*, and *selection*. The general paradigm is as follows:

1. **Initialization:** The initial population consisting of μ members (parents) is chosen randomly.
2. **Recombination:** The μ parent vectors randomly recombine with each other to produce $\lambda \geq \mu$ child vectors.
3. **Mutation:** After recombination, the λ child vectors undergo mutation where a random deviation is added to each child vector.
4. **Selection:** The two most commonly used selection strategies are (μ, λ) and $(\mu, \mu + \lambda)$ selection strategies. In (μ, λ) strategy, the best μ child vectors replace the existing μ

parent vectors to become parents in the next generation, whereas in $(\mu, \mu + \lambda)$ strategy, the best μ vectors from the child and parent populations become parents in the next generation.

5. **Termination:** The number of iterations (generations) performed depends on the convergence criterion chosen.

We briefly mention a few advantages of evolutionary algorithms over classical methods.

1. Evolutionary algorithms are multiobjective optimization techniques that generate a set of equally desirable solutions using the concept of Pareto optimality. The decision maker chooses a solution from the set of available Pareto solutions, and thus implicitly assigns a set of weights. Unlike classical methods, no weights are assigned to the various objectives during the course of the algorithm. Therefore, the solutions are found without introducing bias.
2. In evolutionary algorithms, we deal with a population of desirable solutions in each iteration whereas in classical methods we deal with only one solution. Unlike classical methods where a pre-defined rule is used to search through the solutions, evolutionary algorithms use probabilistic rules to search through solutions. Moreover, evolutionary algorithms are easier to implement and are typically faster than classical techniques. Moreover, several parallel implementations are currently available (Lampinen [13]).

Although, evolutionary strategies and genetic algorithms are categorized as evolutionary algorithms, they have an important difference: Evolutionary strategies encode parameters as floating point numbers and then manipulate them using arithmetic operators whereas genetic algorithms encode parameters as bit strings and then manipulate them using logical operators. So, evolutionary strategies are suitable for continuous optimization while genetic algorithms are more suitable in combinatorial optimization. For a detailed overview of evolutionary algorithms, refer Deb [5] and Coello Coello et al. [2].

1.6 Contribution of the thesis

We investigate the use of differential evolution (an evolutionary algorithm) to solve multiobjective optimization problems in this thesis. Unlike classical methods, differential evolution directly solves the multiobjective problem to find the Pareto front of efficient solutions. Moreover, this algorithm uses probabilistic rules to search for solutions and is very efficient in solving medium sized multiobjective optimization problems. We use differential evolution to compute the *efficient frontier* in the classical Markowitz mean-variance optimization problem in finance and illustrate our results on an example.

The thesis is organized as follows: Section 2.1 introduces differential evolution algorithm. Section 2.2 and Section 2.3 describe the detailed differential evolution algorithm and salient features of the algorithm for solving multiobjective optimization problems respectively. Section 2.4 presents our computational results obtained with the algorithm in solving the portfolio selection problem from finance. We conclude with our conclusions and future work in Section 2.5.

Chapter 2

Differential Evolution: An Evolutionary Algorithm for Multiobjective Optimization

2.1 Introduction

In this thesis, multiobjective optimization problems are solved using an evolutionary algorithm called differential evolution. The idea of *differential evolution* was first conceived by Kenneth Price and Rainer Storn (Price and Storn [24], [25]) in 1995. Differential evolution is an evolutionary algorithm that uses the techniques of recombination, mutation and crossover to improve solutions. It is further categorized as an evolutionary strategy because it encodes real valued parameters as floating point numbers.

Differential evolution differs from other evolutionary strategies in the technique used to perturb (mutate) population vectors. Direct search methods such as Nelder-Mead (Nelder and Mead [18]) and Controlled Random Search (CRS) (Price [23]) use reflections of the existing vectors to introduce perturbation. Other evolutionary strategies make use of standard probability density functions such as gaussian or cauchy to introduce perturbation. On the other hand, differential evolution perturbs the population vectors by adding a scaled difference of two

existing randomly selected population vectors. Differential evolution is *self-adjusting* because the perturbations that the difference between the vectors are large in the beginning of the optimization and become smaller as the algorithm approaches the optimal solution. This property makes differential evolution computationally less expensive and less complicated than the other evolutionary strategies.

2.2 Algorithm for Differential Evolution

Consider the multiobjective problem

$$\begin{aligned}
\min \quad & f(x) \\
\text{s.t.} \quad & g_j(x) \leq 0, \quad j = 1, \dots, m \\
& x_i \leq u_i, \quad i = 1, \dots, n \\
& x_i \geq l_i, \quad i = 1, \dots, n
\end{aligned} \tag{2.1}$$

where $x \in \mathbb{R}^n$ and $f(x) = (f_1(x), \dots, f_k(x))^T$. We assume that any equality constraint $h(x) = 0$ in the original MO problem is written as two inequality constraints $h(x) \leq 0$ and $-h(x) \leq 0$ and thus represented in the constraint set of 2.1.

A quick review of notation: Let r , Np denote the generation and the size of the population in each generation respectively. The population in the r th generation is denoted by $P^r \in \mathbb{R}^{n \times Np}$. Let $x_{j,i}^r$ denote the j th component of the i th population vector in the r th generation. We present the differential evolution algorithm that solves the multiobjective optimization problem below:

Algorithm 1 (Differential Evolution for Multiobjective Optimization)

1. **Initialize:** Set generation count $r = 1$. Let P^r be the initial population matrix, whose entries $x_{j,i}^r$ are computed as

$$x_{j,i}^r = K(u_j - l_j) + l_j, \quad j = 1, \dots, n, \quad i = 1, \dots, Np, \tag{2.2}$$

where K is a random number chosen uniformly from $[0, 1]$.

2. **Mutation:** Generate mutated vectors

$$v_i^r = x_{r_0}^r + F(x_{r_1}^r - x_{r_2}^r), \quad i = 1, \dots, Np, \quad (2.3)$$

where for each i , r_0 , r_1 , and r_2 are three distinct indices (different from i) randomly chosen from $\{1, \dots, Np\}$ and $F > 0$ is a suitable parameter whose value is generally chosen to be less than 1.

3. **Crossover:** Generate Np trial vectors, where the i th vector is given by

$$u_{j,i}^r = \begin{cases} v_{j,i}^r & : \text{ if } K \leq Cr \text{ or } j = j' \\ x_{j,i}^r & : \text{ otherwise.} \end{cases} \quad (2.4)$$

where K is a random number chosen uniformly from $[0, 1]$, Cr (crossover probability) is a parameter in $[0, 1]$, and j' is an index chosen randomly from $\{1, \dots, n\}$.

4. **Selection:**

$$x_i^{r+1} = \begin{cases} u_i^r & \text{if } \left\{ \begin{array}{l} \forall j \in \{1, \dots, m\} \quad g_j(u_i^r), g_j(x_i^r) \leq 0 \\ \text{and} \\ \forall l \in \{1, \dots, k\} \quad f_l(u_i^r) \leq f_l(x_i^r) \\ \text{or} \\ \forall j \in \{1, \dots, m\} \quad g_j(u_i^r) \leq 0 \\ \text{and} \\ \exists j \in \{1, \dots, m\} \quad g_j(x_i^r) > 0 \\ \text{or} \\ \exists j \in \{1, \dots, m\} \quad g_j(u_i^r) > 0 \\ \text{and} \\ \forall j \in \{1, \dots, m\} \quad g'_j(u_i^r) \leq g'_j(x_i^r) \end{array} \right. \\ x_i^r & \text{otherwise} \end{cases} \quad (2.5)$$

where $g'_j(u_i^r) = \max(g_j(u_i^r), 0)$ and $g'_j(x_i^r) = \max(g_j(x_i^r), 0)$ are the constraint violations.

5. **Update generation count:** Set $r = r + 1$. If $r > r_{max}$, *STOP*. Else, return to Step 2.

2.3 Features of differential evolution

We briefly mention the salient features of this algorithm.

1. The mutation performed is called *differential mutation* due to the use of the scaled difference of two randomly chosen population vectors. The parameter F is a scaling parameter that controls the rate at which the population evolves.
2. To introduce additional perturbation, differential evolution uses *uniform crossover*. The crossover is said to be uniform because irrespective of the position of a parameter in the trial vector, it has an equal probability Cr of getting its value from the mutated vector.
3. The crossover probability indicates the probability of choosing a parameter from the

mutated vector over the current population vector. During crossover, to ensure that the trial vector is not identical to the current vector, the value of parameter at the $j = j'$ th position is taken from the mutated vector.

4. The restriction on $r0$, $r1$, and $r2$ guarantees that differential evolution's unique strategy of mutation and crossover does not reduce to simply mutation, crossover or arithmetic recombination.
5. The parameters F and Cr control the convergence speed and the robustness of the algorithm. By trial and error, suitable values for the parameters have been found to be $Np = 5 \times n \dots 30 \times n$, $F = 0.90$, and $Cr = 0.9$. Depending on the problem, the values can be modified to increase the rate of convergence. For details on how to choose values, we refer the reader to Price, Storn and Lampinen [26] and Price and Storn [24], [25].
6. Since trial vectors are generated through a series of steps, it is necessary to check whether they satisfy the boundary conditions. The computationally less expensive as well as the commonly used rule is

$$u_{j,i}^r = \begin{cases} K.(u_j - l_j) + l_j & : \text{ if } u_{j,i}^r < l_j \text{ or } u_{j,i}^r > u_j \\ u_{j,i}^r & : \text{ otherwise} \end{cases} \quad (2.6)$$

where, $j = 1, \dots, n$, $i = 1, \dots, Np$ and K is a uniform random number between 0 and 1.

7. The feature that makes differential evolution unique is the shape of the probability density function (PDF) used during mutation. Instead of using standard PDFs, differential evolution uses vector difference distribution. There are $Np(Np - 1)/2$ unique non zero vector differences for a population of size Np , Taking into account the two opposite directions, there are $Np(Np - 1)$ non zero vector differences in the distribution. This implies that the mean of the distribution is always zero. The shape of the distribution changes automatically depending on the objective function surface being searched. This can be attributed to the self adapting nature of differential evolution.

8. The control parameters F and Cr also influence the working of the algorithm. The parameter F reduces the occurrence of stagnation (especially in smaller populations) by preventing the trial vectors from being very close to each other. The parameter Cr also helps prevent stagnation by introducing vectors in addition to the existing $Np(Np - 1)$ vectors to choose from. These additional vectors are created during crossover by skillfully combining parameters from the trial as well as target vectors. Due to the dependence of the distribution on the above mentioned factors, it is practically impossible to predict the shape of the PDF (Lampinen and Storn [19]).
9. In the differential evolution algorithm, the inequality constraints and the objective functions are handled during selection. The selection technique is based on Lampinen's direct constraint handling method (Lampinen [11], [12]) that is used to solve constrained single objective problems. It employs Pareto dominance to check the feasibility of a trial vector and also to decide whether the trial vector is a non dominated solution. The trial vector replaces the current population vector in the next generation when one of the following three conditions is met:
 - (a) The trial and current vectors are both feasible and the objective function values of the trial vector are smaller than that for the current vector.
 - (b) The trial vector is feasible but the current vector is infeasible.
 - (c) The trial vector is infeasible but the constraint violation of the trial vector is less than the current vector.

The objective function values are compared only if the trial vector and the current population vector are both feasible implying that the selection scheme directs the search first towards the feasible regions before considering the objective values.

2.4 Computational Results

We illustrate the use of multiobjective optimization to solve a portfolio selection problem from finance in this section. Consider the problem of finding an optimal portfolio of four assets

whose future expected returns are estimated using data collected from previous years. Table 2.1 shows the total returns for the four assets for 44 years between 1960 and 2003. The S & P 500 Index, the 10-year Treasury Bond Index, the 1-day federal fund rate and the NASDAQ Composite Index are used to compute the returns on Assets 1, 2, 3 and 4 respectively.

Let TR_{it} , $i = 1, \dots, 4$ and $t = 0, \dots, T$ denote the total return of asset i in the year t where $t = 0$ is 1960 and $t = T$ is 2003. We compute the rate of returns

$$r_{it} = \frac{TR_{i,t} - TR_{i,t-1}}{TR_{i,t-1}} \quad i = 1, \dots, 4, \quad t = 1, \dots, T \quad (2.7)$$

using the entries in Table 2.1 and the entries are displayed in Table 2.2. We then compute the expected value and the variance for the portfolio selection problem 1.6 in Section 1.3 as follows:

The geometric mean (and not the arithmetic mean) is used in the computation of the expected value to account for the multiplicative nature of the rate of return over time. The geometric mean for asset i is calculated as

$$\mu_i = \left(\prod_{t=1}^T (1 + r_{it}) \right)^{\frac{1}{T}} - 1 \quad (2.8)$$

and the results are displayed in Table 2.3. The covariance between the assets i and j is calculated using the formula

$$\Sigma_{i,j} = \frac{1}{T} \sum_{t=1}^T (r_{it} - \hat{r}_i)(r_{jt} - \hat{r}_j) \quad (2.9)$$

where $\hat{r}_i$ and $\hat{r}_j$ are the arithmetic mean of assets i and j . The covariance matrix is given in Table 2.4. The multiobjective formulation (1.6) for the portfolio selection problem is given by

Table 2.1: Total Returns for four assets between 1960 and 2003 (Cornuejols and Tütüncü [4]).

Year	Asset 1	Asset 2	Asset 3	Asset 4
1960	20.255	262.935	100.0	34.461
1961	25.686	268.73	102.33	45.373
1962	23.430	284.09	105.33	38.556
1963	28.746	289.162	108.89	46.439
1964	33.448	299.894	113.08	57.175
1965	37.581	302.695	117.97	66.982
1966	33.784	318.197	124.34	63.934
1967	41.873	309.103	129.94	80.935
1968	46.480	316.051	137.77	101.790
1969	42.545	298.249	150.12	99.389
1970	44.221	354.671	157.48	89.607
1971	50.545	394.532	164.0	114.12
1972	60.146	403.942	172.74	133.73
1973	51.311	417.252	189.93	92.19
1974	37.731	433.927	206.13	59.82
1975	51.777	457.885	216.85	77.62
1976	64.166	529.141	226.93	97.88
1977	59.574	531.144	241.82	105.05
1978	63.488	524.435	266.07	117.98
1979	75.303	531.04	302.74	151.14
1980	99.779	517.86	359.96	202.34
1981	94.867	538.769	404.48	195.84
1982	115.308	777.332	440.68	232.41
1983	150.181	907.712	522.84	247.35
1985	197.829	1200.63	566.08	324.39
1986	234.755	1469.45	605.2	348.81
1987	247.08	1424.91	646.17	330.47
1988	288.116	1522.4	702.77	381.38
1989	379.409	1804.63	762.16	454.82
1990	367.636	1944.25	817.87	373.840
1991	479.633	2320.64	854.1	586.34
1992	516.178	2490.97	879.04	676.95
1993	568.202	2816.4	905.06	776.8
1994	575.705	2610.12	954.39	751.96
1995	792.042	3287.27	1007.84	1052.1
1996	973.897	3291.58	1061.15	1291.0
1997	1298.82	3687.33	1119.51	1570.3
1998	1670.01	4220.24	1171.91	2192.7
1999	2021.4	3903.32	1234.02	4069.3
2000	1837.36	4575.33	1313.0	2470.5
2001	1618.98	4827.26	1336.89	1950.4
2002	1261.18	5558.4	1353.47	1335.5
2003	1622.94	5588.19	1366.73	2003.4

Table 2.2: Rates of Return for the four assets.

Year	Asset 1	Asset 2	Asset 3	Asset 4
1961	0.2681	0.0220	0.0233	0.3166
1962	-0.0878	0.0572	0.0293	-0.1502
1963	0.2269	0.0179	0.0338	0.2045
1964	0.1636	0.0371	0.0385	0.2312
1965	0.1236	0.0093	0.0432	0.1715
1966	-0.1010	0.0512	0.0540	-0.0455
1967	0.2394	-0.0286	0.0450	0.2659
1968	0.1100	0.0225	0.0603	0.2577
1969	-0.0847	-0.0563	0.0896	-0.0236
1970	0.0394	0.1892	0.0490	-0.0984
1971	0.1430	0.1124	0.0414	0.2736
1972	0.1899	0.0239	0.0533	0.1718
1973	-0.1469	0.0330	0.0995	-0.3106
1974	-0.2647	0.0400	0.0853	-0.3511
1975	0.3723	0.0552	0.0520	0.2976
1976	0.2393	0.1556	0.0465	0.2610
1977	-0.0716	0.0038	0.0656	0.0733
1978	0.0657	-0.0126	0.1003	0.1231
1979	0.1861	0.0126	0.1378	0.2811
1980	0.3250	-0.0248	0.1890	0.3388
1981	-0.0492	0.0404	0.1237	-0.0321
1982	0.2155	0.4428	0.0895	0.1867
1983	0.2256	0.0129	0.0947	0.1987
1984	0.0627	0.1529	0.0838	-0.1122
1985	0.3173	0.3227	0.0827	0.3115
1986	0.1867	0.2239	0.0691	0.0753
1987	0.0525	-0.0303	0.0677	-0.0526
1988	0.1661	0.0684	0.0876	0.1541
1989	0.3169	0.1854	0.0845	0.1926
1990	-0.0310	0.0774	0.0731	-0.1780
1991	0.3046	0.1936	0.0443	0.5684
1992	0.0762	0.0734	0.0292	0.1545
1993	0.1008	0.1306	0.0296	0.1475
1994	0.0132	-0.0732	0.0545	-0.0320
1995	0.3758	0.2594	0.0560	0.3991
1996	0.2296	0.0013	0.0529	0.2271
1997	0.3336	0.1202	0.0550	0.2163
1998	0.2858	0.1445	0.0468	0.3964
1999	0.2104	-0.0751	0.0530	0.8558
2000	-0.0910	0.1722	0.0640	-0.3929
2001	-0.1189	0.0551	0.0182	-0.2105
2002	-0.2210	0.1515	0.0124	-0.3153
2003	0.2868	0.0054	0.0098	0.5001

Table 2.3: Geometric mean for the four assets.

Asset 1	Asset 2	Asset 3	Asset 4
0.1073	0.0737	0.0627	0.0991

Table 2.4: Covariance between the four assets.

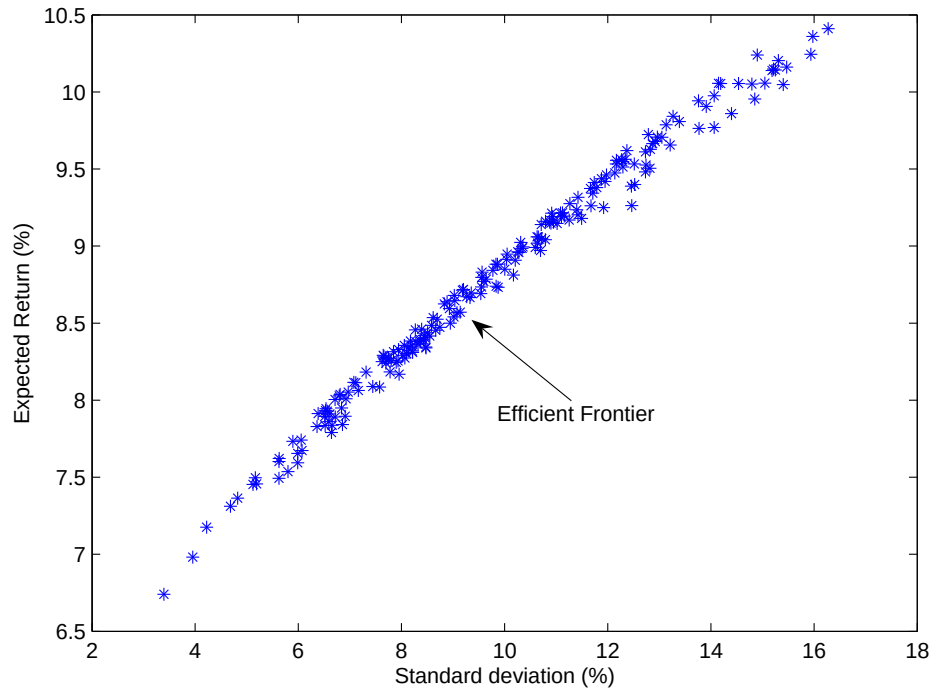
Covariance	Asset 1	Asset 2	Asset 3	Asset 4
Asset 1	0.02778	0.00387	0.00021	0.03491
Asset 2	0.00387	0.01112	-0.00020	0.00054
Asset 3	0.00021	-0.00020	0.00115	0.00043
Asset 4	0.03491	0.00054	0.00043	0.06343

$$\begin{aligned}
 \min \quad & \left[\begin{aligned} & 0.02778x_1^2 + 2 * 0.00387x_1x_2 + 2 * 0.00021x_1x_3 + 2 * 0.03491x_1x_4 \\ & + 0.01112x_2^2 - 2 * 0.00020x_2x_3 \\ & + 0.00115x_3^2 - 2 * 0.00054x_2x_4 - 2 * 0.00043x_3x_4 + 0.06343x_4^2 \\ & - (0.1073x_1 + 0.0737x_2 + 0.0627x_3 + 0.0991x_4) \end{aligned} \right] \quad (2.10) \\
 \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 = 1 \\
 & x_1, x_2, x_3, x_4 \geq 0.
 \end{aligned}$$

The multiobjective problem (2.10) is solved using the differential evolution algorithm. We implemented the algorithm in MATLAB. In order to check that the MATLAB code is correct, we used a three-asset portfolio problem (Cornuejols and Tütüncü [4]) with a known efficient frontier. We compared the efficient frontier obtained using the multiobjective approach with the known efficient frontier and found them to be the same. The values of the control parameters for (2.10) were chosen as $F = 0.95$, $Cr = 0.9$ and $Np = 200$. The algorithm is run for $r_{max} = 500$ iterations. Initially, we ran the algorithm for $r_{max} = 1000$ iterations. Since, there was no difference between the results for 1000 and 500 iterations, we set $r_{max} = 500$. The equality constraint $x_1 + x_2 + x_3 + x_4 = 1$ is handled as two inequality constraints. The code took about 10 seconds to solve the problem. The 200 Pareto optimal solutions computed represent the

efficient portfolios that provide a tradeoff between the expected return and the risk involved. Table 2.5 lists the efficient portfolios for a set of expected returns and variances. We plot the Pareto front objective function space in Figure 2.1. The front represents the efficient frontier for the portfolio selection problem.

Figure 2.1: Efficient Frontier: standard deviations versus rate of returns for (2.10).



One advantage of using multiobjective optimization model for portfolio selection is that the efficient portfolios for different rates of return are found by solving (2.10) only once whereas in the models (1.4) and (1.6), the problem has to be solved once for each value of R and σ^2 respectively. In the MVO problem, we have to decide the range of rates of return for which the efficient portfolios have to be computed whereas in the multiobjective approach, since the objectives are simultaneously solved, we do not have to decide the values for the returns in advance. Choosing the rates of return ourselves could lead to missing out possible rate of returns for which efficient portfolios can be found. The efficient frontier can be plotted automatically

Table 2.5: Efficient Portfolios of the four assets for different rates of return.

Rate of Return	Variance	Asset 1	Asset 2	Asset 3	Asset 4
0.0711	0.0017	0.1246	0.2245	0.6403	0.0106
0.0755	0.0030	0.2353	0.0837	0.6435	0.0375
0.08	0.0049	0.2750	0.2903	0.3840	0.0508
0.0851	0.0071	0.4548	0.1732	0.3654	0.0066
0.0901	0.0108	0.5194	0.2380	0.1995	0.0431
0.0947	0.0159	0.6494	0.0143	0.2569	0.0794
0.1004	0.0200	0.8003	0.0999	0.0753	0.0245

because it resembles the Pareto front.

2.5 Conclusions and Future work

The thesis is concerned with multiobjective optimization problems where one minimizes a set of nonlinear objective functions over a feasible set described by nonlinear constraints. The aim in multiobjective optimization is to find the Pareto front of efficient solutions that provide a suitable tradeoff between the various objectives in the problem.

We use an evolutionary algorithm called differential evolution to solve multiobjective optimization problems in the thesis. This algorithm has the following advantages over classical methods for multiobjective optimization: Differential evolution solves the multiobjective optimization problem directly and computes the Pareto front of optimal solutions. On the other hand, classical methods assign a priori weights to the various objectives in the multiobjective problem and solve the resulting single objective problem using classical algorithms for nonlinear optimization. As a result, the differential evolution algorithm finds a population of Pareto optimal solutions (rather than a single biased solution that depends on the weights) for a multiobjective problem. Moreover, the differential evolution algorithm is very easy to implement and is usually faster than classical methods on small-medium sized multiobjective problems.

We test our differential evolution algorithm on the classical portfolio selection problem from finance where one attempts to find an feasible portfolio that maximizes the return while minimizing the risk involved. The Pareto front corresponds to the efficient frontier for the

portfolio selection problem.

This thesis can be extended to solve multiobjective problems that include integer and discrete variables by appropriately modifying the differential evolution algorithm. Also, the multiobjective problem for portfolio selection can be modified to include constraints due to short sales, diversification and transaction costs. The portfolio selection problems that arise in the real world have a large number of assets and usually also include additional information such as risk rating, earning estimates, etc. The multiobjective problem can be adapted to take the additional information into account.

Bibliography

- [1] C.W. CARROL, *The created response surface technique for optimizing non linear restrained systems*, Operations Research, 9, 1962, pp. 169-184.
- [2] C.A. COELLO COELLO, D.A. VAN VELDHUIZEN, G.B. LAMONT, *Evolutionary Algorithms for Solving Multiobjective Problems*, Kluwer Academic Publishers, 2002.
- [3] J.L. COHON AND D.H. MARKS, *A Review and Evaluation of Multiobjective Programming*, Water Resources Research, 11(2), 1975, pp. 208-220.
- [4] G. CORNUEJOLS AND R. TÜTÜNCÜ, *Optimization Methods in Finance*, Cambridge University Press, 2007.
- [5] K. DEB, *Multiobjective Optimization using Evolutionary Algorithms*, John Wiley & Sons Ltd., 2001.
- [6] M. EHRGOTT, *Multicriteria optimization*, Springer - Berlin, New York, 2000.
- [7] K.R. FRISCH, *The logarithmic potential method of convex programming*, Memorandum, University Institute of Economics, Oslo, 1955.
- [8] D.E. GOLDBERG, *Genetic Algorithms in search optimization and machine learning*, Addison-Wesley, 1989.
- [9] J.H. HOLLAND, *Outline for a logical theory of adaptive systems*, Journal of the Association for Computing Machinery, 3, 1962, pp. 297-314.

- [10] J. JAHN, *Vector Optimization: Theory, Applications, and Extensions*, Springer Verlag, 2004.
- [11] J. LAMPINEN, *A Constraint Handling Approach for the Differential Evolution Algorithm*, Proceedings of the 2002 Congress on Evolutionary Computation, Volume 2, pp. 1468-1473, 2002.
- [12] J. LAMPINEN, *Multi-Constrained Non-Linear Optimization by the Differential Evolution Algorithm*, edited by R. Roy , M. Koppen, S. Ovaska, T. Furuhashi and F. Hoffman, Soft Computing and Industry - Recent Advances, pp. 305-318, Springer Verlag, 2002.
- [13] J. LAMPINEN, *Differential Evolution - new naturally parallel approach for engineering design optimization*, edited by B.H.V. Topping, Development in computational mechanics with high performance computing, Civil-Comp Press, Edinburgh, 1999, pp. 187-197.
- [14] J. LAMPINEN AND I. ZELINKA, *Mixed Variable Non-Linear Optimization by Differential Evolution*, Proceedings of Nostradamus'99, 2nd International Prediction Conference, 1999.
- [15] S. MARGLIN, *Public Investment Criteria*, MIT Press, Cambridge, Massachusetts, 1967.
- [16] H. MARKOWITZ, *Portfolio Selection*, Journal of Finance, 7, 1952, pp. 71-91.
- [17] K. M. MIETTINEN, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, 1999.
- [18] J.A. NELDER AND R. MEAD, *A simplex method for function minimization*, Computer Journal, 7, 1965, pp. 308-313.
- [19] J. LAMPINEN AND R. STORN, *New Optimization Techniques in Engineering*, edited by G.C. Onwubolu and B.V. Babu, Springer Verlag, 2004.
- [20] A. OSYCZKA, *Multicriteria optimization for engineering design*, edited by J.S. Gero, Design Optimization, Academic Press, 1985, pp. 193-227.

- [21] V. PARETO, *Cours d'Economie Politique*, Librairie Droz, Geneve, 1964 (the first edition in 1896).
- [22] V. PARETO, *Manual of Political Economy*, The Macmillan Press Limited, 1971 (the original edition in French in 1927).
- [23] W.L. PRICE, *A controlled random search procedure for global optimization*, edited by L.C.W. Dixon and G.P. Szegö, Toward global optimization 2, North Holland, Amsterdam, 1978, pp. 71-84.
- [24] K.V. PRICE, R.M. STORN, *Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces*, Technical Report TR-95-012, ICSI, 1995.
- [25] K.V. PRICE, R.M. STORN, *Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*, Journal of Global Optimization, 11 (4), 1997, pp. 341-359.
- [26] K.V. PRICE, R.M. STORN, J.A. LAMPINEN, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, 2005.
- [27] I. RECHENBERG, *Evolutionstrategie*, Frommann-Holzboog, Stuttgart, 1973.
- [28] H.P. SCHWEFEL, *Evolution and optimum seeking*, Wiley, 1994.
- [29] T.J. STEWART, *Convergence and Validation of Interactive Methods in MCDM: Simulation Studies*, edited by M.H. Karwan, J. Spronk and J. Wallenius, Essays in Decision Making: A Volume in Honor of Stanley Zionts, Springer-Verlag, 1997, pp. 7-18.
- [30] L.A. ZADEH, *Optimality and Nonscalar Valued Performance Criteria*, IEEE Transactions in Automatic Control, AC-8(1), 1963, pp. 59-60.