Portfolio Management Using Value at Risk:

A Comparison Between Genetic Algorithms and Particle Swarm Optimization

Valdemar Antonio Dallagnol Filho

Supervised by Dr. Ir. Jan van den Berg

Master Thesis Informatics & Economics

July 2006

Dedicated to my parents, Lourdes and Valdemar.

Acknowledgements

First I would like to thank my supervisor Dr. Ir. Jan van den Berg for his help and invaluable insights. It was really a pleasure to work with him.

Furthermore, I would like to thank my family for their love and unconditional support: Lourdes, Valdemar, Luciana, Paulo, Rodrigo and Lucas.

Finally, there were many people that I met during my brief stay in Rotterdam who made my experience abroad unforgettable. Thank you for all the fun, my friends!

Abstract

In this Thesis, it is shown a comparison of the application of Particle Swarm Optimization and Genetic Algorithms to risk management, in a constrained portfolio optimization problem where no short sales are allowed. The objective function to be minimized is the value at risk calculated using historical simulation.

Four strategies for handling the constraints were implemented for PSO: bumping, amnesia, random positioning and penalty function. For GA, two selection operators (roulette wheel and tournament); two crossover operators (basic crossover and arithmetic crossover); and a mutation operator were implemented.

The results showed that the methods are capable of finding good solutions in a reasonable amount of time. PSO showed to be faster than GA, both in terms of number of iterations and in terms of total running time. However, PSO demonstrated to be much more sensible to the initial position of the particles than GA. Tests were also made regarding the number of particles needed to solve the problem, and 50 particles/chromosomes seemed to be enough for problems up to 20 assets.

Contents

1 Introduction **1**

2 Risk Measures **4**

2.1 Introduction 4

2.2 The Mean-Variance Approach 5

2.3 Value at Risk 10

2.3.1 Parametric Method 12

2.3.2 Historical Simulation Method 14

2.3.3 Monte Carlo Method 16

2.4 Coherent Risk Measures 17

3 Nature Inspired Strategies for Optimization **19**

3.1 Introduction 19

3.2 Particle Swarm Optimization 20

3.3 Genetic Algorithms 25

3.4 Conclusion 29

4 Experiment Set-Up **30**

4.1 Data Collecting 30

4.2 Model Building 31

4.3 Experiment design 33

5 Empirical Results **35**

5.1 The objective function 35

5.2 Consistency of the algorithms 36

5.3 Speed of algorithms 40

5.4 Sensitivity to initial position 40

5.5 Influence of the number of particles / chromosomes 41

6 Conclusions **45**

A Summary of the data used **47**

B Tables with results **51**

C Pictures with results **58**

References **67**

List of Figures

2.1 Efficient frontier of risky assets 8

2.2 Minimum variance and tangency portfolios 9

2.3 Two distributions with the same VaR but different CVaR 11

2.4 VaR and CVaR of portfolios of 2 assets 13

3.1 Feasible solution space for portfolio optimization with 3 assets 22

3.2 Particle bumping into boundary 23

3.3 Example of chromosomes with binary and real encoding 25

3.4 Basic crossover 27

3.5 Whole arithmetic crossover 28

3.6 Mutation operator 28

5.1 Contour plot showing the VaR of a portfolio with 3 assets 37

5.2 Example of random initial position of particles 44

A.1 Distribution of daily returns of Verizon, for two different horizons . . . 48

C.1 Optimal portfolio weights, using different objective functions and different horizons for the data 59

C.2 Typical run of PSO using bumping strategy 60

C.3 Typical run of PSO using amnesia strategy 61

C.4 Typical run of PSO using random positioning strategy 62

C.5 Typical run of GA using roulette wheel selection and whole arithmetic crossover 63

C.6 Typical run of GA using tournament selection and basic crossover . . . 64

List of Tables

A.1 Average returns of the companies 49

A.2 Standard deviations of the returns of the companies 50

B.1 Comparison of different risk measures 51

B.2 Consistency of PSO and GA for portfolios with different sizes 52

B.3 Speed of PSO and GA for portfolios with different sizes 53

B.4 Influence of the initial position on the consistency 54

B.5 Influence of the number of particles on the consistency 55

B.6 Consistency of PSO and GA considering a big number of particles/ chromosomes 56

B.7 Speed of PSO and GA considering a big number of particles/ chromosomes 57

Chapter 1

Introduction

The main idea of this Master Thesis is to check the applicability of Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) to risk management. A portfolio containing multiple assets reduces the overall risk by diversifying away the idiosyncratic risk. It is therefore good to consider as many assets as possible, with the limitations of the costs of maintaining such a varied portfolio. Calculating the optimal weights for the portfolio may be a computationally intensive task and thus it is interesting to find heuristic optimization methods that are fast and yet reliable. To test the performance of PSO and GA in this task, subsets of the stocks of the Dow Jones Industrial Average are used here and the percentage of the investment put in each of the assets (weights) is defined by minimizing the Value at Risk (VaR) of the portfolio. Moreover, the constraint of no short-sales is added, which means that none of the weights can be negative.

Value at Risk is a measure of risk that tries to determine the maximum loss of a portfolio for a given confidence level. The VaR may also be interpreted as the *quantile* of a distribution - the value below which lie $q\%$ of the values, for a given time horizon. Although some people argue that it is not a good measure of risk, because of its lack of coherence (see section 2.4), it is much used in practice, especially considering the BIS (Bank for International Settlements) requirement (Hawkins, 2000).

To solve the optimization problem of minimizing the variance (another common measure of risk), quadratic programming has often been used. But when the problem includes a large number of assets or constraints, finding the best solution becomes more time demanding. In these cases, different approaches have been employed, including PSO and GA. Xia *et al.* (2000) used a Genetic Algorithms for solving the mean-variance

optimization problem with transaction costs. Chang *et al.* (2000) focused on calculating the mean-variance frontier with the added constraint of a portfolio only holding a limited number of assets. They have used three heuristics to solve this problem, including a Genetic Algorithm. Finally Kendall Kendall & Su (2005) maximizes the Sharpe ratio using Particle Swarm Optimization, but for only a very limited number of assets. It was not found in the literature articles applying GA or PSO to portfolio optimization using VaR, which shows the relevance of this Thesis.

The *Particle Swarm Optimization* algorithm is based on the behavior of fishes and birds, which collaboratively search an area to find food. It is a systematic random search like Genetic Algorithms in the sense that the algorithm moves through the solution space towards the most promising area, but the exact path is not deterministic. PSO has a population consisting of various particles, with each particle representing a solution. The particles are all initialized in the search space with random values for the weights and the velocities. Then they 'fly' through the search space, with the velocity in each iteration determined by a momentum term plus two random terms associated with the best solution found by the particle and to the best solution found by all the particles in the population. If some constraints are imposed, it is possible that at some point particles will try to cross the boundaries of the feasible space, mainly because of their momenta. There are different strategies to assure that the solutions found remain feasible. In this work, four of them are implemented and discussed: bumping, amnesia, random positioning and penalty function (see section 3.2 for more details about the strategies).

Genetic Algorithms are techniques that mimic biological evolution in Nature. Given an optimization problem to solve, GA will have a population of potential solutions to that problem. To determine which are better fit in a given generation, a fitness function (the objective function of the optimization) is used to qualitatively evaluate the solutions. After using some selection strategy to choose the best chromosomes of the population, offsprings are generated from them, hoping that the offspring of two good solutions will be even better. One major decision in GA is the way the solutions are encoded. In this work the real encoding is used, with the genes consisting of real numbers and representing the weights for each of the assets (see section 3.3).

GA uses special operators (selection, crossover, mutation), and the design of these operators is a major issue in a GA implementation and usually is done ad hoc. In this work, two selection operators (tournament and roulette-wheel selection) and two crossover operators (basic crossover and whole arithmetic crossover) are implemented.

A mutation operator to assure genetic variability was also implemented. GA may also use elitism to make sure that the best solutions of each generation survive intact into the next generation.

To implement the algorithms, the software Matlab® version 7.0 was used. To estimate the VaR of a given portfolio, it was used the historical simulation method. This method has as advantage that it does not requires the simplifying assumption that the underlying distribution is normal, and yet does not add much computational burden.

The two characteristics that were used to evaluate the performance of the algorithms are the consistency (ability to always arrive at the global optimum of the problem) and the speed of convergence to the best solution. It was also investigated the influence of the number of particles/chromosomes on the quality of the solutions; and the sensitivity of the algorithms to the initial position of the particles/chromosomes.

This Thesis is structured as followed: chapter 2 makes a review about risk and possible measures of it: variance (section 2.2), value at risk (section 2.3) and conditional value at risk (section 2.4). Special attention is given to VaR and to three ways of calculating it: the parametric method, the historical simulation method and the Monte Carlo Method.

The next part (chapter 3) deals with Nature inspired strategies for optimization, in particular Particle Swarm Optimization (section 3.2) and Genetic Algorithms (section 3.3). It is shown the basics of these methods, together with strategies for handling constraints of the portfolio optimization problem.

Chapter 4 explains the experiment set-up. Section 4.1 describes the data used in the empirical part. Section 4.2 describes the parameters chosen for the optimization methods and the initialization procedure. And in section 4.3, the design of the experiments is discussed. Finally in chapter 5, the results of the experiments are presented and discussed.

Chapter 2

Risk Measures

2.1 Introduction

Finding a general definition of risk in the literature is hard, as points Holton (2004), who defines it as an "*exposure to a proposition of which one is uncertain*". He emphasizes that there are two components of risk: *exposure* and *uncertainty*. In general, there is exposure to a proposition when there are material consequences from it. As pointed by Holton, the test to be see if there are material consequences is: "*would we care?*" Or in other words "*if we immediately considered the proposition, would we have a preference for it to be true or false?*" It is possible that a person is exposed to a proposition without even knowing about it. Take the example of children playing with sharp knives - they are exposed to the possibility of getting cut, even though they are unaware of it. The second component of risk, uncertainty, means that the person does not know whether a proposition is true or false. Probability may be used as a metric of uncertainty, although it only quantifies perceived uncertainty. Very often the word *risk* is used meaning *probability* and measuring only negative risks. However, in the Risk Management area, risk is used to refer to a combination of the probability of an event happening together with the possible outcome of that event.

Knight (1921)[1] distinguishes *risk* from *uncertainty*:

> "But Uncertainty must be taken in a sense radically distinct from the familiar notion of Risk, from which it has never been properly separated. The term 'risk', as loosely used in everyday speech and in economic discussion,

[1] Available online at: http://www.econlib.org/library/Knight/knRUP.html

> really covers two things which, functionally at least, in their causal relations to the phenomena of economic organization, are categorically different. (...) The essential fact is that 'risk' means in some cases a quantity susceptible of measurement, while at other times it is something distinctly not of this character; and there are far-reaching and crucial differences in the bearings of the phenomenon depending on which of the two is really present and operating. (...) It will appear that a measurable uncertainty, or 'risk' proper, as we shall use the term, is so far different from an unmeasurable one that it is not in effect an uncertainty at all. We shall accordingly restrict the term 'uncertainty' to cases of the non-quantitative type."

Financial risks, as defined by Jorion (2001) are "*those which relate to possible losses in financial markets, such as losses due to interest rate movements or defaults on financial obligations*". It is part of the job of financial institutions to manage the financial risks, so that they are kept into tolerable levels. In order to manage the risks taken, some metrics are needed. In this chapter, three possible risk measures are explained: the mean-variance framework, the Value at Risk (VaR) and the Conditional Value at Risk (CVaR). The VaR has a special importance after the Basel Capital Accord of 1998 and the later amendments, which set capital requirements for commercial banks to guard against credit and market risks; and allow banks to use their own VaR models to calculate the capital required.

The remaining of this chapter is structured as follows. Section 2.2 gives a quick review of the mean-variance approach introduced by Markowitz. Although it may be shown that the scope of application of the Markowitz framework is limited to some special cases, it is still used in practice and it is a standard in introductory risk textbooks. Section 2.3 explains the Value at Risk and three ways of calculating it: the parametric method (section 2.3.1), the historical simulation method (section 2.3.2) and the Monte Carlo method (section 2.3.3). Finally, section 2.4 explains the concept of coherent risk measures and defines a coherent risk measure: the Conditional Value at Risk.

2.2 The Mean-Variance Approach

Harry Markowitz introduced in 1952 the concepts of Modern Portfolio Theory (MPT) and because of his innovative ideas was awarded in 1990 the Nobel Prize, together with Merton Miller and William Sharpe. Prior to his seminal article, investors

usually assessed the risk and rewards of each of the securities individually when constructing a portfolio. The risk involved was considered an adjustment needed to the expected returns and was calculated ad hoc (Szego, 2002). The securities that offered the best rewards with low risk where used to construct a portfolio, which means that eventually an investor could end up with a portfolio of securities all from the same sector, just because they seemed to have good risk-reward characteristics when considered individually. This intuitively is not a good decision, because if there is a crash in the sector, all of the eggs would be in the same basket.

Markowitz insight was that instead of evaluating the securities individually, what should be evaluated was the risk of the overall portfolio. To assess the risk of the individual securities, Markowitz proposed to use the variance of the returns, and in the case of a portfolio, the risk should be measured by using the covariance between all pairs of investments (Markowitz, 1952).

The variance of the returns is defined by:

$$\sigma_X^2 = E\left[(X - E[X])^2\right], \tag{2.1}$$

where X are random returns. The covariance is defined by:

$$cov(X, Y) = \sigma_{X,Y} = E\left[(X - E[X])(Y - E[Y])\right], \tag{2.2}$$

where X and Y are random returns. It should be noted that $cov(X, X) = \sigma_X^2$. With the pairwise covariances it is possible to build the covariance matrix:

$$\Omega = \begin{bmatrix} \sigma_{1,1} & \sigma_{1,2} & \dots & \sigma_{1,N} \\ \sigma_{2,1} & \sigma_{2,2} & \dots & \sigma_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{N,1} & \sigma_{N,2} & \dots & \sigma_{N,N} \end{bmatrix}. \tag{2.3}$$

Given a portfolio with N stocks; n_i shares invested in each stock ($i = 1, 2, \dots, N$) and being the prices per stock given by $v_1, v_2, \dots, v_N$, the value of the portfolio is:

$$V = \sum_{i=1}^{N} n_i v_i. \tag{2.4}$$

The weights of the investment in each asset are given by:

$$\omega_i = \frac{n_i v_i}{V}, \tag{2.5}$$

where it should be noted that:

$$\sum_{i=1}^{N} \omega_i = 1. \tag{2.6}$$

The vector of weights and the vector of expected returns are defined respectively as:

$$\overrightarrow{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_N \end{bmatrix} \quad \text{and} \quad \overrightarrow{\mu} = \begin{bmatrix} E[R_1] \\ E[R_2] \\ \vdots \\ E[R_N] \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_N \end{bmatrix}. \tag{2.7}$$

The expected return of a portfolio can be calculated by:

$$E[R_p] = \mu_p = \sum_{i=1}^{N} \omega_i \mu_i = \overrightarrow{\omega}^T \overrightarrow{\mu}. \tag{2.8}$$

To reduce the risk of a portfolio, it is interesting to include securities with returns that move in opposite directions, so that if one performs badly, there is another one that performs well to compensate for that. Thus the risk of the overall portfolio is reduced. The use of the covariance between the returns of the securities in a portfolio allows this assessment.

The risk of the portfolio, measured by its variance, is calculated by:

$$\sigma_p^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \omega_i \omega_j \sigma_{i,j} = \overrightarrow{\omega}^T \Omega \overrightarrow{\omega} \tag{2.9}$$

It is possible to find a portfolio that minimizes the risk for a given expected return, or stated differently, a portfolio that maximizes the expected return for a given level of risk. A portfolio with this characteristic is called an *efficient portfolio*. The set of all efficient portfolios form the *efficient frontier*. Figure 2.1 illustrates a typical efficient frontier. All efficient portfolios lie in the upper part of the solid line (in black) and all the others lie in the area delimited by it. The lower part of the solid line (in gray) form the boundary of the area with the feasible portfolios, but the portfolios in that region are obviously not efficient, because it is always possible to find another one with lower risk for the same expected return.

Calculating the portfolios on the frontier can be done through an optimization problem stated as:

$$min(\sigma_p^2) = \sum_i \sum_j w_i w_j \sigma_{i,j}, \tag{2.10}$$

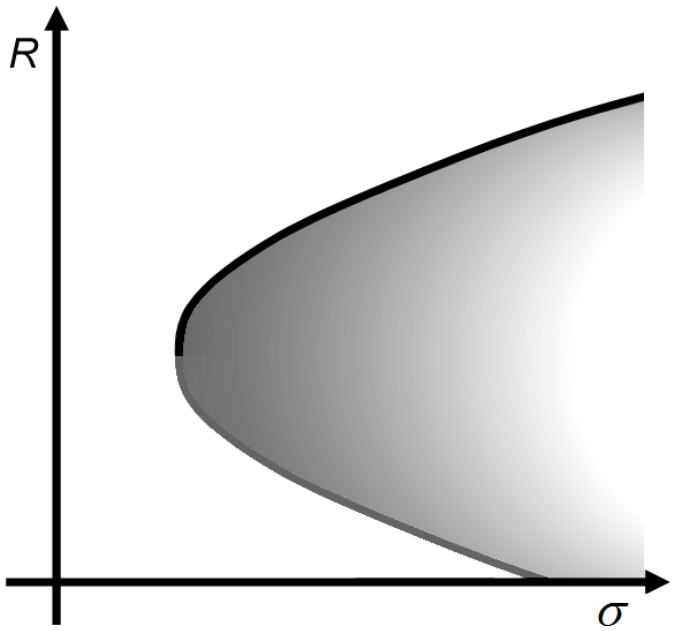


Figure 2.1: All possible portfolios are in the area shaded in gray. The upper part of the parabolic region (black line) is called the efficient frontier of risky assets and is formed by the portfolios which minimize the variance for a given expected return

subject to:

$$R_p = \sum_{i=1}^{N} w_i r_i \qquad \text{and} \qquad \sum_{i=1}^{N} w_i = 1. \tag{2.11}$$

In the case where no short sales are allowed, there is the extra constraint:

$$0 \leq w_i \leq 1. \tag{2.12}$$

The portfolio optimization problem can also be written as the dual problem where the return is maximized given an investor's desired level of risk.

Two interesting portfolios in the efficient frontier are the *minimum variance portfolio* and the *tangency portfolio*. The minimum variance portfolio is simply the portfolio that has the minimum variance among all other portfolios, not considering the returns, and is the portfolio A in figure 2.2.

To define the tangency portfolio it is necessary to first define the Sharpe ratio. The Sharpe ratio combines the information from the mean and the variance of an asset and is defined as:

$$S_p = \frac{R_p - R_f}{\sigma_p}, \tag{2.13}$$

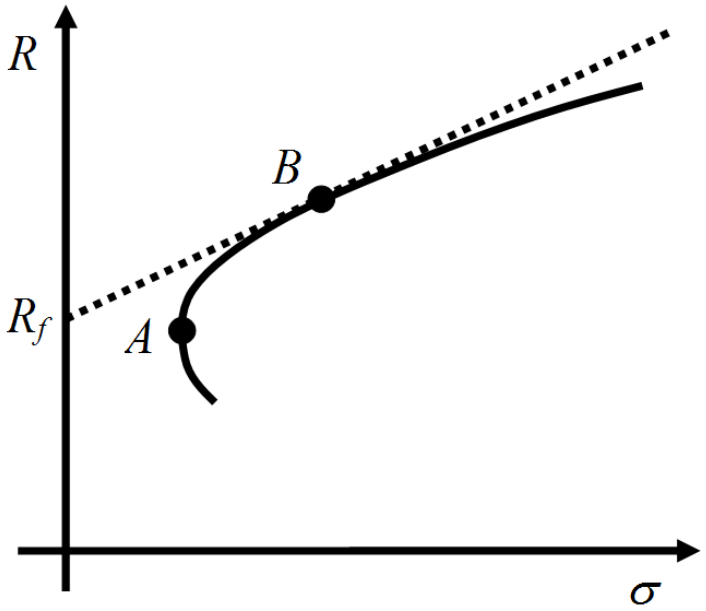


Figure 2.2: Efficient frontier of risky assets (solid line) and capital allocation line with riskless asset (dashed). Portfolio A is the portfolio with minimum variance and portfolio B is the tangency portfolio with maximum Sharpe ratio

where S_p is the Sharpe ratio of the portfolio, R_p is the return on the risky portfolio and R_f is the risk-free rate. $R_p - R_f$ is the excess return of the portfolio and σ_p is the standard deviation of the returns. The Sharpe ratio uses excess returns instead of nominal returns because instead of showing how much return it is possible to get from a risky portfolio, it shows how much extra return you may get for going from a riskless to a risky position.

In figure 2.2, point B denotes the portfolio with maximum Sharpe ratio. By making a portfolio that combines the risky portfolio B with the risk-free rate, we have the return over the investment defined by:

$$R_i = (1 - p)R_f + pR_p, \tag{2.14}$$

where p represents the proportion of the money that is invested in the risky portfolio with respect to the amount that is invested in the riskless position. This equation generates the *capital allocation line*, shown as a dashed line in figure 2.2, which allows the investor to choose the level of risk he is willing to assume by simply changing the proportion of the investment between the riskless asset and the tangency risky portfolio (which remains constant).

The main problem of using the Markowitz mean-variance framework is that it is only suited to the case of elliptic distributions (distributions where the equity-density

surfaces are ellipsoids). This is the case of the normal or the t-distribution with finite variances. A symmetric distribution is not necessarily elliptic. The use of this framework with assets that present returns defined by non-elliptic distributions can seriously underestimate extreme events that may cause great losses (Szego, 2002).

2.3 Value at Risk

Value at Risk (VaR) is a measure of risk that tries to determine the maximum loss of a portfolio for a given confidence level and holding period. It allows managers to say: "*we are X percent certain that we will not lose more than V dollars in the next N days*" (Hull, 2002). Being X a real-valued random variable; $F_X(x) = P[X \leq x]$ the cumulative distribution function associated to it; and $F_X^{-1}(\alpha)$ the inverse function of $F_X(x)$, the VaR for a confidence level of $(1-\alpha)$ is defined as (Inui & Kijima, 2005):

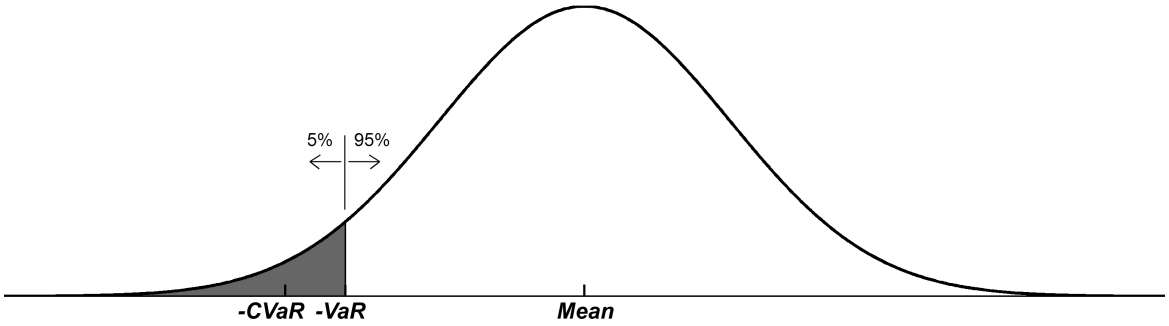
$$VaR_{(1-\alpha)} = -F_X^{-1}(\alpha) = -inf[x \mid F_X(x) \geq \alpha]. \tag{2.15}$$

Figure 2.3a shows a normal probability density function of returns of a portfolio for a given time period (for example, 1 day), with the VaR for a confidence level of 95%. VaR does not tell how much will a portfolio lose in that time period, but it shows that statistically only in 1 day out of 20 (5% of the days) the losses of a portfolio will be larger than the VaR indicated in the figure. The Conditional Value at Risk (CVaR) will be detailed in section 2.4.

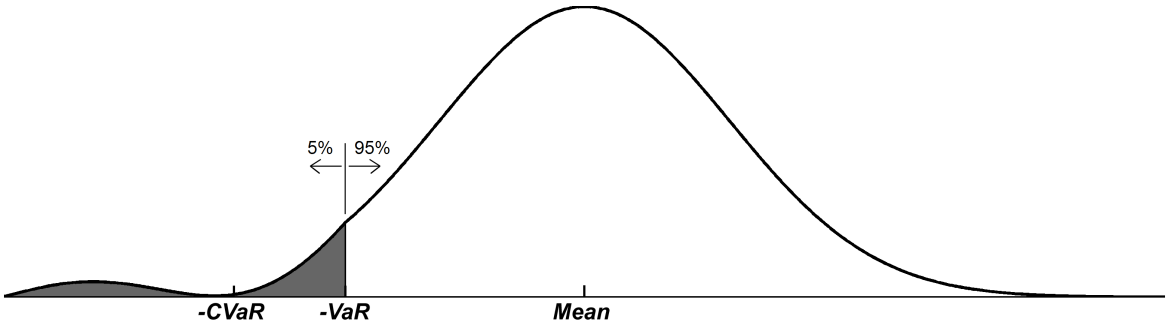
The VaR may be interpreted as the *quantile* of a a distribution - the value below which lie q% of the values (Bodie *et al.*, 2005). For the normal distribution, it is easy to calculate the VaR. For example, considering $q = 5\%$ (a confidence level of 95%), the VaR will lie 1.65 standard deviations below the mean, and may be interpreted as if with 5% of probability there will be a loss equal than or larger than VaR.

Although in Figure 2.3a the distribution was considered to be normal, the VaR framework is not limited to this case. The normal distribution is a commonly used simplification to make the calculations easier. However, considering the distribution to be normal may significantly underestimate the risks involved when the actual distribution has "fat tails".

VaR is a standard measure of risk for banks, which usually use confidence levels of 99%, according to the BIS (Bank for International Settlements) requirement (Hawkins,



(a) VaR of a normal distribution with 95% confidence level



(b) VaR of a skewed distribution with 95% confidence level

Figure 2.3: Two distributions with the same VaR but different CVaR. Conditional Value at Risk (CVaR) is the expected loss given that we are in the q% left tail of the distribution. It is treated in more detail in section 2.4

2000). However, the highest is the confidence level (the closer to 100%), the rarer are the events situated on the left of the VaR in the probability distribution. This way, this events will be more unlikely to have happened in the past and to be present in the historical data used in the model. Thus, it will be harder to do accurate forecast about these extreme events in the future.

VaR calculations assume that the portfolio composition will not be changed over the holding period. The most common choice for holding periods in banks is 1 day. BIS prescribes a 10-day holding period for VaR calculations, but it allows banks to calculate this value by multiplying the bank's 1-day VaR by the square root of 10, which is valid if market moves are independently distributed over time (Hawkins, 2000).

The popularization of VaR in the Financial world is mainly due to JPMorgan, which in the late 1980s developed a VaR system that encompassed all the company, modeling hundreds of risk factors. Quarterly, a covariance matrix was updated with historical data; and daily the risk represented by the positions taken by the company

were assessed and presented in a Treasury meeting. The clients of the company became interested in the system and instead of selling the software, JPMorgan opted to publish a methodology and distribute the calculated covariance matrix. The service was named RiskMetrics® (Holton, 2003, p. 18).

However, there are some critiques to the use of VaR as a risk measure (Szego, 2002):

- it lacks subadditivity, in such a way that portfolio diversification may lead to an increase of risk (see section 2.4 for an explanation of coherent risk measures; and Tasche (2002) for an example of the lack of subadditivity in VaR);
- it is non-convex and has many local minima, making it hard to be used in optimization problems. Figure 2.4 shows the VaR of a portfolio of two assets, as a function of the weight, allowing to visualize the existence of local minima. It is also possible to see that the portfolios obtained are different depending on the risk measure used - minimizing VaR and estimating it with the parametric method, the best w is found to be 0.46, while using historical simulation, the best w is found to be 0.32; and minimizing CVaR using historical simulation the best w is 0.50;
- it may provide results that are conflicting at different confidence levels.

Despite the critiques that the use of VaR receives from the academic literature, the use of VaR is prescribed by regulatory agencies because it is a compact representation of risk level and allows the measurement of downside risk (Szego, 2002).

To calculate the Value at Risk, three possible methodologies are (Holton, 2003, p. 3):

- parametric method;
- historical simulation method;
- and Monte Carlo method.

2.3.1 Parametric Method

The parametric method relies on the assumption that the risk factors follow a known probability distribution. Suppose that the returns of the assets of a portfolio in

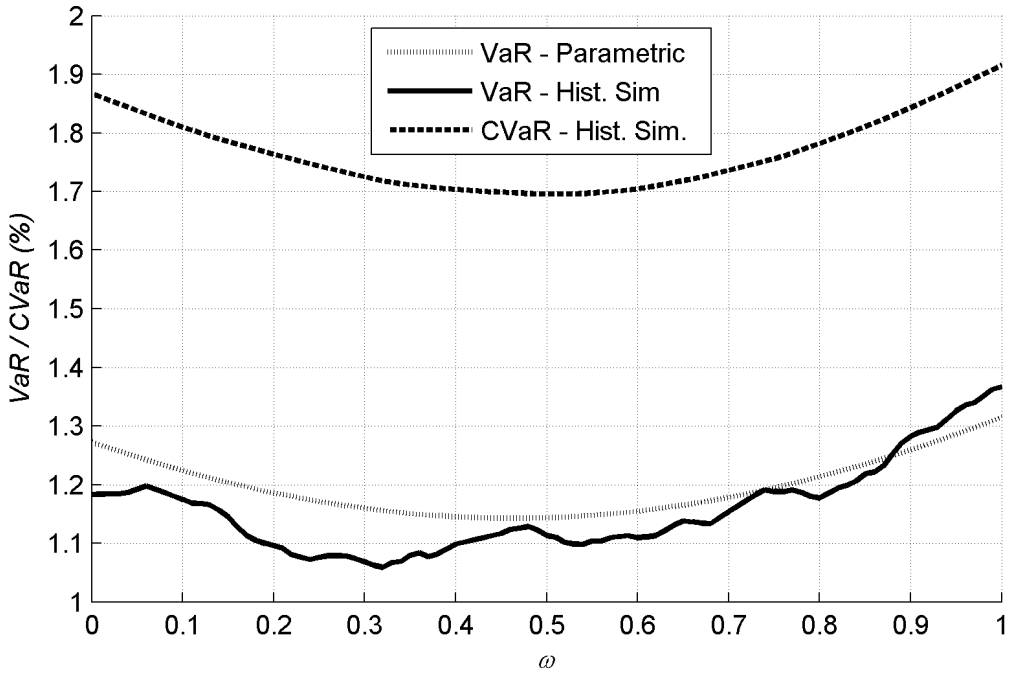


Figure 2.4: VaR and CVaR of portfolios of 2 assets, showing the existence of local minima for the VaR calculated using historical simulation. The two stocks used in the portfolio are American Express Co. and American International Group Inc., including 1250 data points with daily frequency (approximately 5 years), starting on 6/jun/2001. The proportion of the portfolio invested in American Express Co. is ω and the remaining $(1-\omega)$ is invested in American International Group Inc.

an 1-day period follow a normal distribution:

$$R_p \sim N(\mu_p, \sigma_p)$$

with R_p being the stochastic returns of the portfolio. The 1- and 10-day VaR of the portfolio with a 99% confidence level are:

$$VaR_p(1, 99\%) = 2.33\, \sigma_p\, V \tag{2.16}$$

$$\begin{aligned} VaR_p(10, 99\%) &= \sqrt{10} \cdot VaR_p(1, 99\%) \\ &= \sqrt{10} \cdot 2.33\, \sigma_p\, V. \end{aligned} \tag{2.17}$$

Obviously given a same holding period and assuming that the distribution of returns is normal, the problem of finding the portfolio that minimizes the Value at Risk is the same as the problem of finding a portfolio with minimum variance, no matter which confidence level is being used.

By what was said, it would be expected that portfolios calculated by minimizing the VaR using the parametric method and considering the distribution of returns to be normal will be less conservative than portfolios calculated using non-parametric methods with the actual (or approximated) distribution of returns, because the normal distribution considers that extreme events are very improbable to happen. Returns lower than 4 standard deviations from the expected return, for example, have a probability of happening of only 0.003167%. However, there is evidence that individual returns distributions show fat tails, implying that extraordinary losses happen more frequently that predicted by the normal distribution.

On the other hand, if it is considered a well-diversified portfolio with many different risk factors, even if the risk factors distributions are not normal, the central limit theorem says that the resulting distribution of returns of the portfolio will converge to a normal distribution. This way, it may be considered that the portfolio returns follow a normal distribution, as long as the portfolio is well diversified and the risk factor returns are independent from each other (Crouhy *et al.*, 2001, p. 193).

2.3.2 Historical Simulation Method

Historical simulation is another approach to calculating the Value at Risk, which is more flexible than the parametric method. Because of the simple intuition behind, it is widely accepted by management and trading communities. Besides that, it has the advantage of easily handling options in the portfolio (Best, 1998).

The flexibility of historical simulation comes from the fact that it does not assume that the returns follow any specific distribution. Therefore, historical simulation will produce higher VaR numbers for distributions with fat tails than the ones obtained when using the parametric method and assuming an underlying normal distribution.

It is possible however that a particular scenario of the risk factors that would produce a large loss is not present in the historical data used for simulation. This is more likely to happen when using short periods in the time series. For this reason, many banks prefer to use Monte Carlo simulation instead of historical simulation (Best, 1998).

The procedure used to calculate the VaR of a portfolio using historical simulation is given below:

1. Consider a portfolio of N assets defined by a vector of weights:

$$\overrightarrow{\omega_0} = \begin{bmatrix} \omega_{0,1} & \omega_{0,2} & \dots & \omega_{0,N} \end{bmatrix}^T. \tag{2.18}$$

2. For every asset price or risk factor involved in the problem, obtain a series of returns for a given time period (for example, 500 days). In the case that log-returns are used, they are calculated as follows:

$$r_{k,t} = log\frac{p_{k,t}}{p_{k,t-1}}, \tag{2.19}$$

where $r_{k,t}$ and $p_{k,t}$ are respectively the return and price of the asset k at time t.

3. Consider each of the days in the time series of returns as a scenario for possible changes in the next day. As there are N assets, each day t of historical data will form a scenario defined by:

$$\overrightarrow{r_t} = \begin{bmatrix} r_{1,t} & r_{2,t} & \dots & r_{N,t} \end{bmatrix}^T. \tag{2.20}$$

It is important to notice that from this point on the scenarios $\overrightarrow{r_t}$ are no longer seen as time series, but just as a set of different possible realizations of the random vectors $\overrightarrow{r_t}$, obtained from historical data.

4. Apply each of the scenarios to the composition of the portfolio today, i.e., do not apply the price changes in cascade[2] to the portfolio. This means that the result of the application of scenario t to the portfolio is:

$$\overrightarrow{\omega_t} = \begin{bmatrix} \omega_{t,1} \\ \omega_{t,2} \\ \vdots \\ \omega_{t,N} \end{bmatrix} = \begin{bmatrix} \omega_{0,1} \cdot e^{r_{1,t}} \\ \omega_{0,2} \cdot e^{r_{2,t}} \\ \vdots \\ \omega_{0,N} \cdot e^{r_{N,t}} \end{bmatrix}. \tag{2.21}$$

Note that although the notation $\omega_{t,k}$ is used to represent weights in the portfolio, they will not be normalized in this procedure, in such a way that $\sum_{k=1}^{N} \omega_{t,k}$ for $k \neq 0$ may be different than one.

5. The log-returns of the portfolio for each of the scenarios are calculated as:

$$R_t = log\left(\sum_{k=1}^{N} \omega_{t,k}\right), \tag{2.22}$$

remembering that $\sum_{k=1}^{N} \omega_{0,k} = 1$

[2] Cascade (from the Merriam-Webster Online Dictionary - http://www.m-w.com): "*something arranged or occurring in a series or in a succession of stages so that each stage derives from or acts upon the product of the preceding*"

6. Sort the portfolio returns (R_t) for the various scenarios into percentiles.

7. The VaR will be the return that corresponds to the desired level of confidence. For example, if there are 500 days and a level of 99% confidence is desired, the VaR will be the fifth worst return of the portfolio.

It is important to note that each scenario of changes is applied to the composition of the portfolio today and not in cascade. This is done because if the changes were made in cascade, the percentage value changes would no longer refer to the original portfolio value. And also the proportion between the assets in the portfolio would change in relation to one another.

The main drawback of historical simulation is that it relies completely on the particular realization of a stochastic process represented by the set of historical data available. Thus, it should be believed that the events that happened in the past are a good representation of what will happen in the future. Events like a market crash or periods of extremely volatility present in the historical data may jeopardize the results obtained by the method. Another problem of the method is that it depends on data availability. If one year of data is used, 250 data points will be present. In the Monte Carlo simulations, for example, at least 10 000 scenarios are generated. The use of small samples may not represent the distribution well enough and interpolation may be necessary(Crouhy *et al.*, 2001, p. 206-212).

2.3.3 Monte Carlo Method

The Monte Carlo method was invented by Stanislaw Ulam in 1946 (Eckhardt, 1987) and covers any technique of statistical sampling used to approximate solutions to quantitative problems. In the method, the random process under analysis is simulated repeatedly, where in each simulation will be generated a scenario of possible values of the portfolio at the target horizon. By generating a large number of scenarios, eventually the distribution obtained through simulation will converge towards the true distribution. A good description of the method can be found, for example, in Holton (2003, cap. 5).

As advantages of the method, Crouhy *et al.* (2001) mentions the fact that any distribution of the risk factors may be used; the method can be used to model any complex portfolio; and it allows the performance of sensitivity analyses and stress

testing. As disadvantages it may be mentioned the fact that outliers are not incorporate into the distribution; and that it is very computer intensive.

2.4 Coherent Risk Measures

Considering that a risk measure may be defined as a relation ρ between a space X of random variables and a non-negative real number R, a coherent risk measure $\rho : X \to R$ must satisfy the following four properties (Artzner *et al.*, 1999):

Translation invariance: $\rho(X + \alpha \cdot r) = \rho(X) - \alpha$, for all random variables X, real numbers α, and rates of return r on a reference instrument. It implies that with the addition of a riskless return $\alpha \cdot r$ to the portfolio, the risk $\rho(X)$ decreases by α.

Subadditivity: $\rho(X_1+X_2) \leq \rho(X_1)+\rho(X_2)$, for all random variables X_1 and X_2. The interpretation given to this property is that "merger does not create extra risk", or that by splitting up a company, the risk could be reduced. Any positively homogenous functional ρ is convex if and only if it is subadditive (Szego, 2002).

Positive homogeneity: $\rho(\lambda X) = \lambda\rho(X)$, for all random variables X and real numbers $\lambda \geq 0$. It should be noted that the consequences of lack of liquidity due to a position size, which may influence risk, should be taken into account when computing the future value of a position.

Monotonicity: $X_1 \leq X_2$ implies $\rho(X_1) \leq \rho(X_2)$, for all random variables X_1 and X_2.

Given the set of properties above, VaR can not be considered a coherent measure of risk. For example, only in the special case when the joint distribution of returns is elliptic VaR is subadditive; and in this case the portfolio that minimizes the VaR is the same that would be obtained by simply minimizing the variance.

As an example of a risk measure that may be proved to satisfy the aforementioned four properties, and thus to be coherent, is the Conditional Value at Risk (CVaR), also called Mean Excess Loss, Mean Shortfall, or Tail VaR. It tries to answer the question: "*if things do get bad, how much can we expect to lose?*" (Hull, 2002). CVaR is the expected loss given that we are in the $q\%$ left tail of the distribution. Being X a real-valued random variable; $f_X(x)$ the probability density function associated to it;

and $x_\alpha = VaR_{(1-\alpha)}$, the CVaR for a confidence level of $(1-\alpha)$ is defined as (Inui & Kijima, 2005):

$$\begin{aligned} CVaR_{(1-\alpha)} &= -E[X \mid X \leq -VaR_{(1-\alpha)}] \\ &= -\frac{1}{\alpha}\int_{-\infty}^{x_\alpha} x f_X(x)dx \end{aligned} \tag{2.23}$$

Some characteristics of CVaR are:

- it is more conservative than VaR ($CVaR(p) \geq VaR(p)$, for any portfolio p);
- it is convex, which makes portfolio optimization easier. Figure 2.4 is an example where CVaR is clearly convex;
- it is coherent risk measure in the sense of Artzner *et al.* (1999);
- it is a better representation of the risks involved in extreme events. For example, in in Fig. 2.3, two portfolios exhibit the same VaR, but clearly the portfolio shown in Fig. 2.3b is riskier than the one shown in Fig. 2.3a;
- linear programming can be used for optimization.

Chapter 3

Nature Inspired Strategies for Optimization

3.1 Introduction

Computer Science and information technology always used biological and natural processes as a source of inspiration. Some examples of Nature Inspired strategies for problem solving are Artificial Ants Colonies, Swarm Intelligence and Evolutionary Computing.

In this research it is given special attention to two systematic random searches inspired in Nature: *Particle Swarm Optimization* (PSO) and *Genetic Algorithm* (GA). Both algorithms are intrinsically parallel, which means that they explore several locations of the solution space at the same time. Many other algorithms for solving the same kind of problems are serial and can only explore the solution space of a problem in one direction at a time. Another notable strength of PSO and GA is that they perform well in problems for which the search space is complex - those where the objective function is discontinuous, noisy, changes over time, or has many local optima. Most practical problems have a vast solution space, which are impossible to search exhaustively; the challenge then becomes how to avoid the local optima. PSO and GA, by their characteristic of exploring simultaneously different parts of the solution space, are less prone to converge to these local optima. Finally, these algorithms are flexible to handle constraints, which may be implemented more easily, when comparing to the 'standard' optimization techniques.

3.2 Particle Swarm Optimization

The *Particle Swarm Optimization* (PSO) algorithm was introduced by Kennedy & Eberhart (1995) and is based on the behavior of fishes and birds, which collaboratively search an area to find food. It has links to Genetic Algorithms and Evolutionary computing, but does not suffer from some of the problems found in Genetic Algorithms, such as: the iteration with the group improves the solution, instead of detracting the progress; and PSO has a memory, which is not the case of Genetic Algorithms, where even if elitism is used only a small number of individuals will preserve their identities (Eberhart & Kennedy, 1995).

PSO has a population consisting of various particles, with each particle representing a solution. The particles are initially positioned randomly in the search space (i.e. assigned random values for the weights). It is desirable that the particles are initially well spread to assure a good exploration and minimize the risk of getting trapped into local optima.

Apart from *position*, a particle in the PSO also have a *velocity*, which is also initialized randomly. The velocity determines in which direction a particle will move and how far it will go. The position of particle i in the next iteration is calculated as:

$$pos_{i,d,t+1} = pos_{i,d,t} + vel_{i,d,t+1}, \tag{3.1}$$

with $vel_{i,d,t+1}$ being the velocity of particle i in dimension d at iteration t, calculated by:

$$\begin{aligned} vel_{i,d,t+1} = w_t \cdot vel_{i,d,t} + c_1 r_1 (pbest_{i,d} - pos_{i,d,t}) \\ + c_2 r_2 (gbest_d - pos_{i,d,t}), \end{aligned} \tag{3.2}$$

where w_t is a weight which gives the particle momentum; r_1 and r_2 are random numbers; c_1 and c_2 are scaling constants; $pbest_{i,d}$ is the best position of particle i in dimension d in all previous iterations; $gbest_d$ is the best position of the entire population in dimension d in all previous iterations; and $pos_{i,d,t}$ is the position of particle i in dimension d at time t. This formulation is usually referred to as *Particle Swarm Optimization with Inertia*. w_t is here defined as:

$$w_t = w_{max} - \frac{w_{max} - w_{min}}{N} \cdot t, \tag{3.3}$$

with w_{max} and w_{min} being the value of w_t for the first ($t = 0$) and last ($t = N$) iterations respectively, considering $w_{max} \geq w_{min}$. N is the total number of iterations and t the number of the present iteration.

In equation (3.3) it is possible to see that the momentum influence is higher in the beginning of the search, decreasing linearly until the end of the search, when it becomes less important to the direction of flight, comparing to the influence of $pbest_{i,d}$ and $gbest_d$. Higher values of w_t increase the exploration of the search space, because the direction of flight of the particles will be guided mainly by the previous values of the velocity (remembering that the initial velocities are random). On the other hand, lower values of w_t mean more exploitation of "good" regions in the solution space, because the particles will tend to move to the direction of the best solutions found both locally and globally.

To increase the exploration of the solution space, in this implementation random numbers r_1 and r_2 are drawn from independent uniform distributions ranging from $(p_i - 1)$ to p_i, with $p_i \in [0, 1]$ and $i \in \{1, 2\}$. For this particular implementation of the algorithm, p_1 and p_2 were chosen to be both equal to 0.9 in such a way that $r_1, r_2 \in [-0.1, 0.9]$. Of course, the better exploration of the solution space comes at the cost of reducing the exploitation of promising areas, as some particles may be diverged to the opposite directions of $pbest_{i,d}$ and/or $gbest_d$.

There are other formulations to calculate the acceleration of the particles, for example, the constriction factor method (CFM), proposed by Clerc (1999), or the use of neighborhoods of particles (Eberhart & Kennedy, 1995), but these strategies are not used in this work.

In the case of the portfolio selection problem, a solution consists of a set of weights for the various assets in the portfolio. If there are N possible assets to choose from, then a solution in the search space will contain $N - 1$ weights. Being ω_i the weight for the asset i, the weight of the last asset is simply determined by manipulating equation 2.6 into:

$$\omega_N = 1 - \sum_{i=1}^{N-1} \omega_i, \tag{3.4}$$

As it is a very common constraint in real life, especially in emerging markets (de Roon, 2001), the optimization problem treated in this work does not allow short sales, which means that the weights of the assets in the portfolio cannot be negative. In the case of portfolios with N assets, the feasible solutions space is delimited by

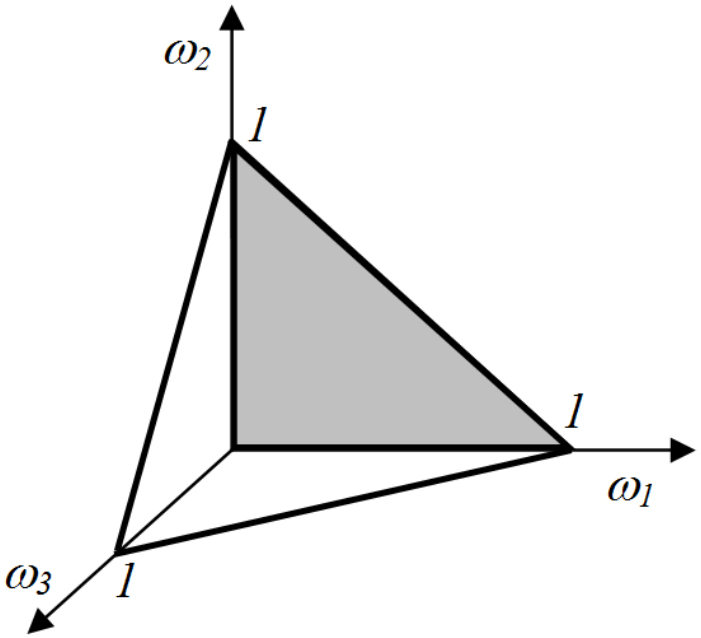


Figure 3.1: Feasible solution space for portfolio optimization with 3 assets

hyperplanes with equations:

$$\begin{aligned} \omega_1 &= 0, \\ &\vdots \\ \omega_{N-1} &= 0, \\ \sum_{i=1}^{N-1} \omega_i &= 1. \end{aligned} \tag{3.5}$$

In the specific case with 3 assets, the feasible solution space is depicted in Figure 3.1, corresponding to triangle with vertices in the points $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. As one of the weights may be calculated by knowing the other two (see equation (3.4)), the algorithm may concentrate in finding a solution lying in the shaded area of the figure, and calculate afterwards ω_3.

There are different strategies to make sure that the particles abide to the imposed constraints and each has its advantages and disadvantages. Two conventional strategies to make sure that all the particles stay within the feasible space are here called *bumping* and *random positioning* (Zhang *et al.*, 2004).

The *Bumping* strategy resembles the effect of a bird hitting a window. As the particle reaches the boundary of the feasible space, it 'bumps'. The particle stops on

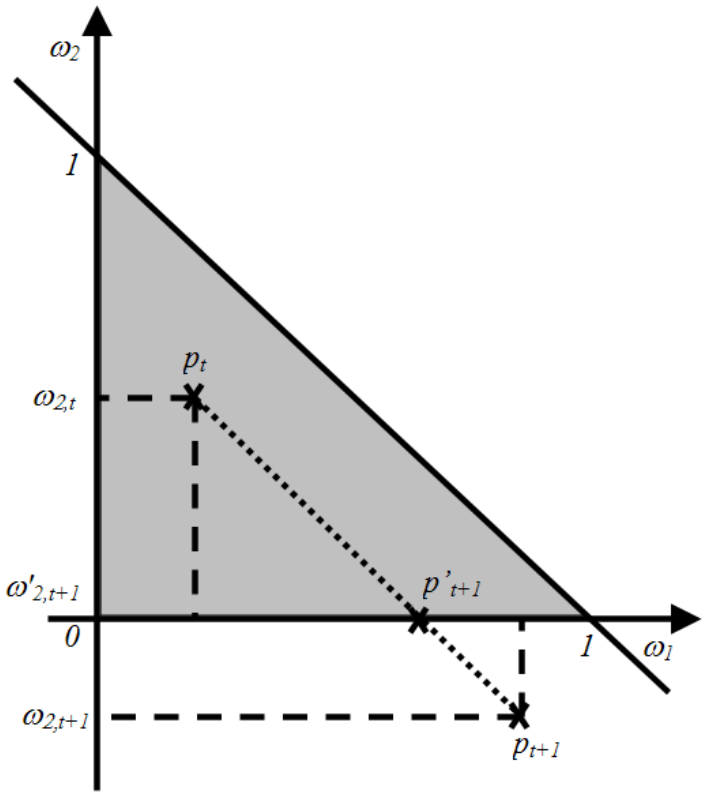


Figure 3.2: Particle bumping into boundary

the edge of the feasible space and loses all of its velocity. Figure 3.2 shows a particle bumping into a boundary. The initial position of the particle was p_t and the position for the next iteration was calculated to be p_{t+1}. However, p_{t+1} is outside the feasible solutions space, having $\omega_2 < 0$. To avoid that the particle enters the infeasible space, it is stopped at p'_{t+1}, defined by the weights recalculated as:

$$\omega'_{d,t+1} = w_{d,t} + v_d \cdot \frac{\omega_{d,t}}{\omega_{d,t} - \omega_{d,t+1}} \qquad \text{for } d = 1, ..., N-1 \tag{3.6}$$

where $\omega_{d,t}$ is the weight d of the particle in the last iteration; $\omega_{d,t+1}$ is the weight d of the particle without bumping; $\omega'_{d,t+1}$ is the weight after bumping; v_d is the velocity of particle d. The weight $\omega'_{N,t+1}$ is calculated by:

$$\omega'_{N,t+1} = 1 - \sum_{i=1}^{N-1} \omega'_{i,t+1}, \tag{3.7}$$

as usual.

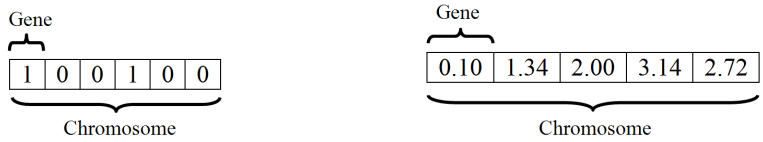
It should be clear by looking at Figure 3.2 that (3.6) comes from triangle similitude, where the distance between p_t and p_{t+1} is the hypotenuse and $\omega_{i,t} - \omega_{i,t+1}$ is one of the cathetus.

After bumping, in the next iteration the particle will gain velocity again, starting from velocity zero and applying (3.2). If the *gbest* is near the edge of feasible space, then bumping makes sure the particles remain near the edge and thus near the *gbest*. However, because of the loss of velocity caused by bumping into the boundaries, the particles may get 'trapped' at the current *gbest* and not reach the real global optimum, resulting in a premature convergence to a sub-optimal solution.

The *random positioning* strategy simply changes the negative weight into a random, feasible weight, and normalizes the weights so that they add up to one. This strategy increases the exploration of the search space, when comparing to bumping. The premature convergence, which may occur with bumping, does not occur here. However the opposite may be true for random positioning: there will no convergence to the real global optimum at all. Especially if the optimal solution is near the boundaries of the feasible region, it may happen that particles approaching it will be thrown back into different and worse areas. And when they fly back towards the promising area, if they try to violate a constraint they will be thrown again to a random position. Thus the particles may get stuck in a loop and never be able to come to a stop at the optimal solution.

Hu & Eberhart (2002) come with a different strategy, henceforth called *amnesia*. With the initialization all the particles are feasible solutions. But during the iterations the particles are allowed to cross the boundaries and fly into the infeasible space. However the particles will not remember the value of the objective function of the solutions found in the infeasible space and if they find a better solution, it will not be recorded neither as *pbest* nor as *gbest*. Because of this *pbest* and *gbest* will always lie inside the feasible space and thus the particles will be attracted back there. A downside of amnesia may be the fact that the particles 'waste time' flying in infeasible space, which in some cases may result in an inefficient search.

Finally, it may be used the *penalty function* strategy, which is well known in optimization problems. It consists in adding up a penalty to the evaluated objective function of a particle if it is located outside the feasible space. If the penalty is big enough, the points explored outside the feasible area will not be remembered as optimal solutions. In this implementation, the penalty added to the objective function is 1, which adds a loss equal to 100% of the capital of a portfolio located outside the feasible area.



(a) Binary encoding (b) Real encoding

Figure 3.3: Example of chromosomes with binary and real encoding

3.3 Genetic Algorithms

A *Genetic Algorithm* is a search technique to find solutions to optimization and search problems. One of the first references to it was made by Holland (1975). It uses concepts inspired from biological evolution such as inheritance, selection, crossover and mutation. In a GA a population of candidate solutions is simulated and, like in Nature, only the best individuals survive and are able to transmit their genes to the next generations. The individuals are usually referred to as *chromosomes*. Commonly the chromosomes are represented as binary strings, but there are other possible encodings, such as the real encoding (Arumugam & Rao, 2004), where each gene in the chromosome is a real number. The real encoding allows the algorithm to search for a solution in a continuous space, rather than in a discrete search space. Each element of a chromosome is called a gene, and depending on the choice of encoding, may be a binary digit or a real number, for example. Figure 3.3 shows an example of two chromosomes, using binary and real encoding.

The algorithm starts with a population of random individuals and the evolution happens in generations. In each generation the fitness of the individuals is evaluated and accordingly selected for being recombined or mutated into the new population. The steps of a Genetic Algorithm may be summarized to:

```
* Initialize population with random individuals
* Repeat
   * Evaluate the fitness of the individuals in the population
   * Select pairs of individuals to reproduce, according
     to their fitness
   * Generate new population, through crossover and
     mutation of the selected individuals
* Until terminating condition (e.g. number of iterations,
  solution found that satisfies a criterium, etc)
```

In a generation, the individuals have their fitness evaluated. The function that quantitatively assigns a fitness value to an individual is the objective function of the optimization, which is tried to be maximized. The fitter individuals are more likely to be selected for reproduction. However, the selection processes is usually stochastic, allowing that some less fit chromosomes are selected for reproduction, trying to keep the diversity of the population and avoiding premature convergence to sub-optimal solutions. Two well known methods for selection are the *roulette wheel* and the *tournament* selection.

In *tournament* selection pairs of individuals are randomly chosen from the larger population and compete against each other. For each pair, the individual with the larger fitness value wins and is selected for reproduction. *Roulette wheel* selection is a form of fitness-proportionate selection in which the chance of an individual being selected is proportional to the amount by which its fitness is greater or less than its competitors' fitness.

Reproduction in Genetic Algorithms is done through two operators: *crossover* (sometimes called recombination) and/or *mutation*. In crossover, two 'parent' chromosomes are recombined hoping that mixing good solutions will result in even better ones. On average, the fitness of the population will increase, as mainly the best individuals are chosen for breeding from the previous generation. Mutation is an operator used to assure genetic diversity in the population, resembling biological mutation. Its purpose is to avoid that the individuals in a population become too similar to one another and converge to local minima. It also avoids the fact that if a population is too homogeneous, the evolution may become too slow.

Designing the operators for selection, crossover and mutation is a major issue in a GA implementation and many times is done ad-hoc. Two different selection methods were implemented in this work: the tournament selection and the roulette wheel selection. A more in-depth look of these methods can be found in Goldberg & Debb (1991).

Two different crossover operators were implemented, and named as: *basic crossover* and *whole arithmetic crossover*. Both are crossover operators commonly used for real-coded GA, mentioned for example in Blanco *et al.* (2001). Each crossover involves two parents, that were selected via the selection operator, and the crossover combines their genes to produce two offsprings. The *basic crossover* picks a random point the parent's chromosomes and then swaps the genes of the parents after that point. Figure 3.4 gives an example of the use of this operator, supposing that the random point for crossover

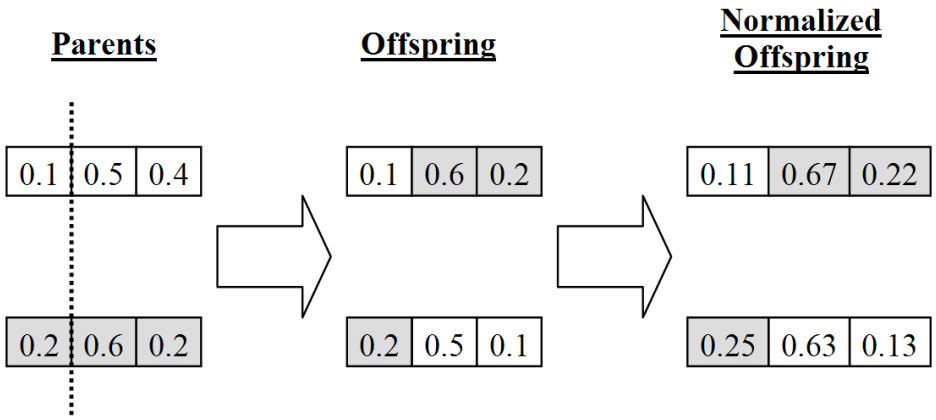


Figure 3.4: Basic crossover

was chosen to be after the first weight. As the solutions may violate the constraint that the weights add up to one after the swap, the weights must be normalized to get the final offspring. Normalization, however, may bring an extra impact on the weights, because it moves the particle in the search space (and thus increases the randomness), resulting in an increase of the exploration of the space and in a reduction of the exploitation of good solutions. However, as a new offspring's weights can sum up to no more than 2, normalizing the weights will in the worse case result in dividing the weights by 2, which seems acceptable. The nonnegative constraint however will never be violated with this crossover, because nothing is subtracted from the weights.

The *whole arithmetic crossover* is designed to be used with real encoding, instead of binary encoding. The first offspring will be calculated as a fraction p of the values of the genes of parent 1 and $1-p$ from parent 2, with $p \in [0,1]$. The second offspring will be the opposite, being composed as a fraction $1-p$ of parent 1 and p of parent 2. Figure 3.5 gives an example of the use of the whole arithmetic crossover, using the same parents as the ones used in Figure 3.4 and supposing that $p = 0.6$. The advantage of this operator is that it automatically holds to both constraints: the weights stay positive and add up to one.

With both crossover methods presented, there is a transfer of information between successful candidate solutions: individuals can benefit from what others have learned, and solutions can be mixed and combined, with the potential to produce an offspring that has the strengths of both its parents. Of course in the course of evolution it may happen that individuals with a bad fit are generated, but they may be easily discarded from one generation to another, so eventually the population of solutions will become fitter.

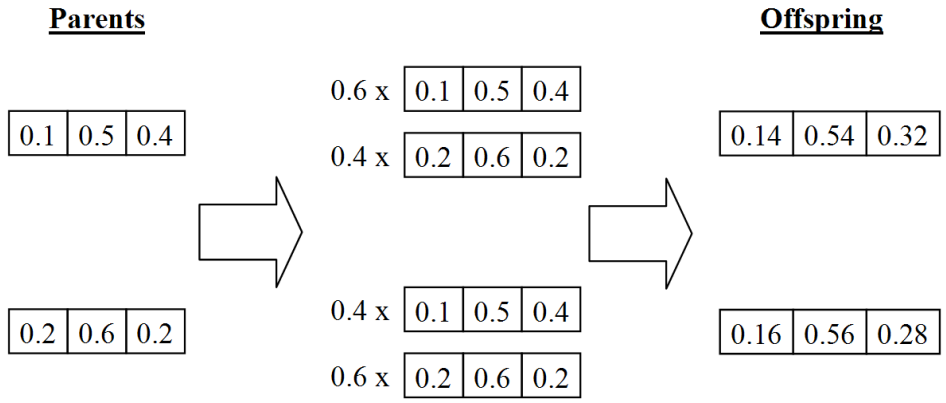


Figure 3.5: Whole arithmetic crossover

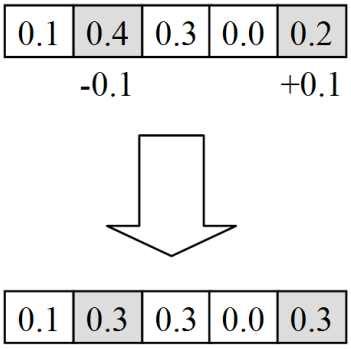


Figure 3.6: Mutation operator

Finally, there is the *mutation* operator, which is analogous to the mutation of living beings that happens in Nature. The mutation operator in GA causes an alteration in an individual's chromosome. In the way it was implemented here, two genes (weights) are chosen randomly to be altered by adding a small value to one and subtracting the same value from the other. Figure 3.6 gives an example of the use of the mutation operator, supposing that the second and the fifth genes were chosen to be mutated, with the random value 0.1 chosen to be subtracted from the second gene and added to the fifth gene.

When applying the mutation operator, the two constraints imposed must be respected (no negative weights and the sum of the weights being equal to one). To make sure that the weights of a solution add up to one, if a subtraction cause a weight to become negative or the addition causes a weight to become larger than one, the value of that weight is set to zero (or one respectively) and all the weights of the solution are normalized.

To avoid that the best solutions are lost throughout the evolution process, by the application of crossover and mutation to each generation, the *elitism* strategy may be used. Elitism assures that the best n solutions of a generation (in this implementation, $n = 1$) are transmitted untouched to the next generation.

3.4 Conclusion

From what was shown in this quick review of the methods, it is expected that when applying them to the portfolio optimization problem, PSO will present a more focused search than GA, with more emphasis in the exploitation of promising areas than the exploration of the whole search space, because of the way that the position of the particles are updated based on the global and local optima. However, some extra exploration may be introduced by tuning the values of p_1 and p_2 mentioned in section 3.2. The use of the random positioning strategy is also a way to increase the exploration of the solution space. It is also expected that the PSO will be more dependent of the initial position of the particles. If they are not well spread in the solution space, it is more likely that it will converge to local optima. Again, the random positioning strategy may contribute to avoid this pitfall. However, given that the initial position of the particles is 'good', the other boundary handling strategies (bumping, amnesia and penalty function) will probably show a better performance than random positioning, in particular if the global optimum is near the boundaries of the feasible solution space. Amnesia and penalty function should perform similarly, as both allow the particles to fly out of the feasible solutions region, not recording (or penalizing) the solutions found there.

From GA it is expected that the whole arithmetic crossover operator result in a better performance than the basic crossover, because it seems to preserve better the knowledge existent in the parent solutions. The basic crossover probably will introduce more randomness to the search and reduce the exploitation of good regions in the solution space. Based on the literature, it is not expected a superior performance from neither the tournament nor the roulette wheel selection. It also expected that GA will show more robustness to initial conditions, due to its higher inherent randomness.

Chapter 4

Experiment Set-Up

4.1 Data Collecting

For the empirical part of this research, data was collect from the Yahoo! Finance web site[1]. It was used the adjusted close price with daily frequency from all stocks in the the Dow Jones Industrial Average[2] (DJIA), which includes 30 major stocks in the U.S. market, with the split and dividend multipliers prescribed by the CRSP (Center for Research in Security Prices) standards. Data was gathered from 05/Jan/1987 to 30/May/2006 (approximately 19 years), totalizing 4895 data points, but for the experiments subsets of it were used. The adjusted price series was used to calculate the log-returns of the stocks, according to equation (2.19). It was noticed that the historical series downloaded from Yahoo! Finance had some missing data points, and to correct it data extracted from the Datastream[3] database was used. Datastream data was not used directly because a series adjusted for split and dividends was not available.

A summary of the data is shown in Tables A.1 and A.2. It is visible that depending on the horizon chosen, the return and risk associated to the stocks may vary significatively. For Verizon, for example, daily returns can vary from 0.006% when using 10 years of historical data, to -0.015% when using only 1 year; and daily risk (measured by the standard deviation of returns) can vary from 0.399% to 0.831%, for 1 year and

[1] http://finance.yahoo.com

[2] http://www.djindexes.com

[3] Datastream is a database provided by Thomson Financial, containing stock market data, company accounts and economics time series. http://www.datastream.com/

10 years horizon respectively. Figure A.1 shows clearly how the distribution of returns change with the change according to the horizon used.

4.2 Model Building

Both PSO and GA are algorithms that depend on good initial positioning of the particles in the solution space. If it was known beforehand in which region the global optimum is located, the algorithms would converge much faster with the particles/chromosomes located initially in that area. However, as there is no a priori knowledge of the best region, it is interesting to have the particles/chromosomes well spread in the feasible solutions space. To allow for a fair comparison between the methods, the same procedure was used to initialize the position of the particles/chromosomes of both methods, with steps listed below:

1. Generate vector:
$$\vec{s} = \begin{bmatrix} 1 & 2 & \cdots & N \end{bmatrix}^T,$$
where N is the number of assets in the problem.

2. Generate vector $\vec{s}'$, by making a random permutation of the vector $\vec{s}$.

3. Generate weights in the sequence determined by $\vec{s}'$. Each weight is a random number from an uniform distribution, ranging from zero to one minus the sum of all the weights already attributed for that particle.

4. Normalize the resulting vector, making the sum of the weights add up to one.

This way, the first weight generated ($\omega_{s'(1)}$) belongs to the interval $[0, 1]$, the second weight belongs to the interval $[0, 1 - \omega_{s'(1)}]$, the third to $[0, 1 - (\omega_{s'(1)} + \omega_{s'(2)})]$, and so on. It should be noted here that $\omega_{s(i)}$ refers to the weight indexed by the i-th element of the vector $\vec{s}'$.

As an example, imagine the case with 3 assets. For a given particle, a possible vector $\vec{s}'$ generated could be:

$$\vec{s}' = \begin{bmatrix} 3 & 1 & 2 \end{bmatrix}^T,$$

which means that the first weight to be initialized will be the one corresponding to the third asset, the second weight to be initialized corresponds to the first asset and finally

the weight corresponding to the second asset will be initialized. The first weight will be a random number in the interval $[0, 1]$, for example, 0.7, resulting in the (incomplete) vector of weights:

$$\overrightarrow{\omega} = \begin{bmatrix} 0 & 0 & 0.7 \end{bmatrix}^T.$$

In the next step, the weight corresponding to the first asset is initialized with a random value in the interval $[0, 0.3]$, for example 0.2. The weight of the second asset is initialized last, with a random number belonging to the interval $[0, 0.1]$, for example 0.05, resulting in the (non-normalized) vector of weights:

$$\overrightarrow{\omega} = \begin{bmatrix} 0.2 & 0.05 & 0.7 \end{bmatrix}^T.$$

To normalize the weights, the vector $\overrightarrow{\omega}$ is divided by the sum of all the weights (in this case, 0.95), resulting in the normalized weight vector:

$$\overrightarrow{\omega} = \begin{bmatrix} 0.21 & 0.05 & 0.74 \end{bmatrix}^T.$$

Li-Ping *et al.* (2005) recommend that for the PSO, the population size for a higher dimensional problem should not be larger than 50 particles, and for a less complicated problem, a population of 20 to 30 particles should be enough, with 30 particles being a good choice. It was noticed that 30 particles seemed to be not enough for the problem under analysis, so for most of the experiments, the size of the population was chosen to be 50 particles. GA was compared to PSO always using the same number of particles/chromosomes for both algorithms.

The parameters of PSO were tuned empirically, and the following values were chosen:

- $w_{max} = 0.9$,
- $w_{min} = 0.4$,
- $c_1 = 0.5$,
- $c_2 = 0.5$,
- $p_1 = 0.9$,
- $p_2 = 0.9$.

For the penalty function strategy, a value of 1 was chosen to be added to the objective function when the solution was unfeasible, which means adding a loss of 100% of the capital invested.

The parameters to be tuned in GA are two probabilities. The first is the probability that a crossover will be performed on two selected chromosomes and the second is the probability that a mutation will happen on the offspring of two solutions. The probability of a crossover occurring was chosen to be 0.8 (80%) and of a mutation 0.01 (1%).

It was noticed that the population in the GA algorithm tended to become very similar after a certain number of iterations. It is interesting that the population converge, but at the same time having equal solutions in the population reduces the exploration of the solution space without adding to exploitation. It was then implemented that when a new generation was calculated the algorithm checked if there were redundant individuals, with 4 digits precision. If any redundant individuals were found, they were mutated with a probability of 100%.

For calculating the Value at Risk and the Conditional Value at Risk, it was chosen a confidence level of 95%. This means that taking 2 years of data (500 data points) and applying the historical simulation method, 25 data points will be located to the left of $-VaR$ (see Figure 2.3). Higher confidence levels would require a larger number of data points and could significantly slow down the experiments.

4.3 Experiment design

The experiments run were basically of two types: one designed to test the *consistency* and the other designed to test the *speed* of the algorithms. *Consistency* is here defined as the ability to always find the optimal solution. This measure is needed because both PSO and GA have random components, and it may happen that one performed better simply because it was lucky. For a given number of assets in the portfolio (sets of 5, 10 and 20 assets were used), 5 subsets of assets were randomly picked from the stocks in the Dow Jones Industrial Average. The reason why several random subsets of the original 30 assets set were generated is to test the performance of the algorithms under different objective function landscapes. Each strategy for boundary handling of PSO and GA was run 5 times over these sets, to try to eliminate the influence of a lucky initial solution. The algorithms were run for a number of iterations

that was big enough for them to converge. The average number of iterations needed by a certain strategy to find a solution that is within 0.1% of the best solution found on that particular run was calculated and presented as $\overline{N_{it}}$. This measures if the convergence of the algorithm to the neighborhood of the best solution is quick or not, in the sense of number of iterations of the algorithm, not considering the time to calculate each iteration. The standard deviation of the number of iterations needed was also calculated, and presented as σ_N. Next, it was calculated the average error between the best solution found by the algorithms in each run, compared to the best solution found by the different strategies on all runs (considered to be the global optimum of the problem), for a particular set of assets. The results are presented as $\overline{\varepsilon_{VaR}}$, and the standard deviation of this measure is presented as σ_ε. The last two measures show how good the algorithm is to find the global optimum of the problem.

The second test was designed to examine the *speed* of the different strategies. The algorithms were run again 5 times for each of the 5 subsets of assets randomly picked and the average time per iteration (t/it) was calculated. As it is interesting to know how long an algorithm takes to converge, it was assumed that if $N'_{it} = \overline{N_{it}} + 2\sigma_N$ iterations were executed, the algorithm would be able to arrive within 0.1% of the best solution that would be found in a particular run approximately 97.7% of the times (considering a normal distribution of N_{it}). $\overline{N_{it}}$ was taken from the consistency test run with the same parameters. The total time needed for the algorithm to converge was presented as $\bar{t}$.

All algorithms were implemented in Matlab® version 7.0. The experiments were performed on two workstations Dell® Optiplex® GX260 with a Intel® Pentium® 4 2.4Ghz processor, 1Gb RAM and 80GB hard drive; and on a HP® Pavilion® notebook, zv6000 series, with a AMD® Athlon® 64 3200+ Processor, 512Mb RAM and 80Gb hard drive.

Chapter 5

Empirical Results

5.1 The objective function

A crucial question when trying to measure risk is how much data should be used, or in other words, how long in the past should one go. If it is desired to test the risk to extreme events, a large horizon should be chosen, as it is more likely that they happened in the past. However, data from the distant past may be not representative of the present, as companies' strategies and the market conditions change. For this work 2 years of data (500 data points) were used. This choice seemed to be realistic to the problems found in the real world and did not demand excessive computer time to run the experiments, when comparing to larger horizons.

To illustrate the effect of choosing different horizons for the input data, portfolios with 5 assets (stocks of 3M, Citigroup, Coca-Cola, Gm and Microsoft) were optimized for different risk measures (variance, VaR calculated with historical simulation, and CVaR calculated with historical simulation), and the optimal weights found are shown in Figure C.1. It is visible that if a horizon includes (or not) a especially bad year for a company, this will be reflected in the weights of the portfolio, and eventually a company will not even be included in a portfolio. This is the case of GM, that is not included in a portfolio with the variance minimized, using 2 years of past data. Depending on the risk measure used, the composition of the portfolio may change drastically, as is the case of portfolios generated with 5 years of data. With that horizon, the weight of Coca-Cola stocks in the portfolio that minimizes the variance is 50.1%; minimizing the VaR it is 69.7%; and minimizing the CVaR it is 35.6%.

Figure C.1 shows that portfolios generated by minimizing the variance or by minimizing the CVaR are more similar than those generated by minimizing the VaR. This happens probably because CVaR is smoother, as it is calculated through an integral (see equation 2.23), and the variance is related to the more 'well behaved' part of the distribution of returns. VaR, on the other hand, is calculated in an area of the distribution that is more sensible to extreme values and to the fact that a discrete distribution is being used. This fact is confirmed by checking Table B.1, where 5 assets were chosen from the stocks in the DJIA and portfolios were optimized with this 5 assets for the three different risk measures. The resulting portfolios had the risk measured according to the other risk measures, and the difference is shown in the table. The best portfolio obtained by minimizing the variance had the CVaR 0.12% worse than the one obtained directly by minimizing the CVaR; and the portfolio obtained by minimizing the CVaR had a variance only 0.09% worse than the variance of the best portfolio obtained by minimizing the variance. However, portfolios obtained by minimizing the variance and by minimizing the CVaR had a VaR respectively 5.56% and 5.09% worse than that of the portfolio obtained by minimizing the VaR, confirming what was said before.

The use of VaR with historical simulation presents a challenge to any optimization method, because of the complex objective function landscape. Figure 5.1 shows the contour plot of the value at risk of portfolios with 3 stocks: 3M Co., American International Group Inc. and E. I. DuPont de Nemours & Co., calculated using historical simulation with 2 years of data. It is visible the existence of several local minima, with the global minimum located in $\omega_1 = 0.26$ and $\omega_2 = 0.08$, where VaR equals 0.0059. When more dimensions (more assets) are added, the complexity increases even more.

5.2 Consistency of the algorithms

The first experiment run was designed to test the consistency of the algorithms (see section 4.3) for solving the portfolio optimization problem. PSO and GA were executed with 50 particles/chromosomes, for portfolios with different sizes (5, 10 and 20 assets). Table B.2 presents the results of the experiment. PSO using the bumping strategy showed the best performance when comparing with the the other strategies of PSO and with GA, when analyzing the number of iterations needed to converge to a solution ($\overline{N_{it}}$) and also to the standard deviation of this measure (σ_N). The average error between the VaR found by this strategy and the best solution ($\overline{\varepsilon_{VaR}}$), which shows if the algorithm got trapped into local optima or did not converge at all, has the same

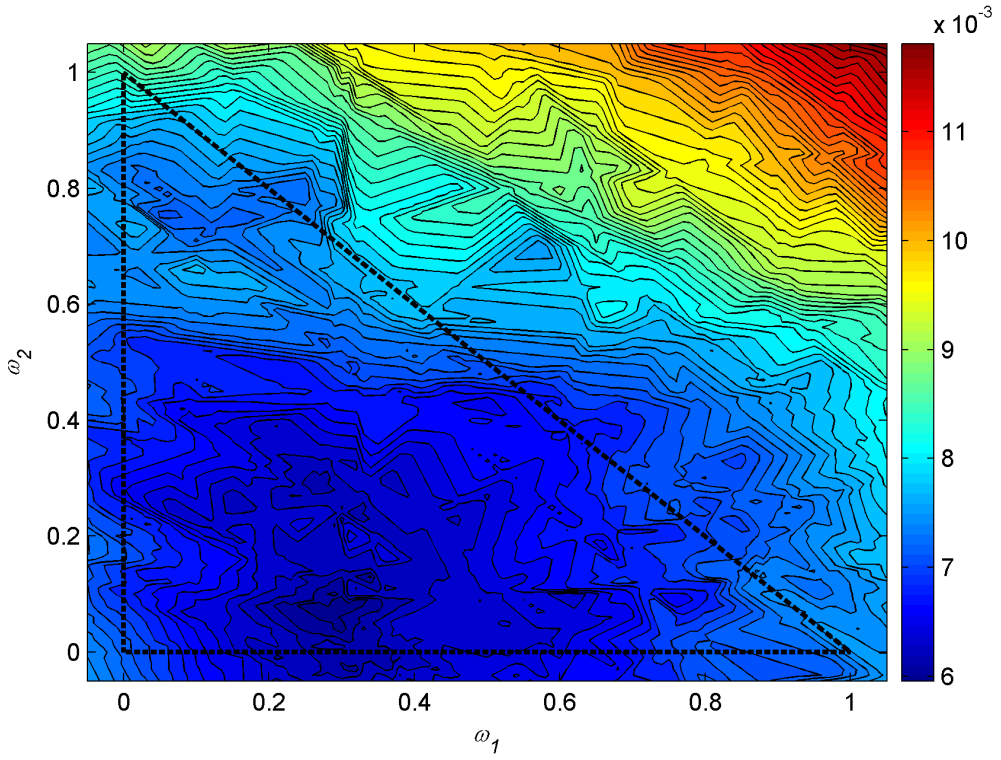


Figure 5.1: Contour plot showing the VaR of portfolios composed by stocks of 3M Co., American International Group Inc. and E. I. DuPont de Nemours & Co., calculated using historical simulation with 2 years of data. ω_1 is the weight of investment in 3M, ω_2 is the weight of investment in Am. International. The amount invested in DuPont is equal to $1-\omega_1-\omega_2$. The triangular shaped area with dashed line delimits the feasible solutions area.

order of magnitude of the other strategies, except when using 20 assets, when it was slightly higher than GA.

PSO using the amnesia or the penalty function strategy showed comparable results as expected, both in terms of number of iterations needed to converge and in terms of the errors comparing to the global optimum. This is not surprising, as the way they are implemented lead to similar search behavior. The two strategies took longer than bumping strategy to converge to a solution, because the particles tend to 'waste time' outside the feasible solutions space. The quality of the solutions found were comparable to the other algorithms for 5 and 10 assets and lower than the other strategies for 20 assets, as visible in the $\overline{\varepsilon_{VaR}}$ (higher values of $\overline{\varepsilon_{VaR}}$ mean lower quality/consistency). This may be due to the fact that in the first iterations, which in general are more exploratory for PSO, many solutions found are not feasible.

PSO using the random positioning strategy had a bad performance when comparing to the other strategies with respect to the number of iterations to converge to a solution. The explanation to this is that if the solutions are near the boundaries of the feasible solution space, it is very likely that the particles approaching the area will eventually try to cross the boundaries and be thrown into a random position away of it. This makes it harder for the algorithm to converge. The average errors to the best solution ($\overline{\varepsilon_{VaR}}$) are slightly lower than the other algorithms, due to the increased exploration performed by the strategy, showing better exploratory power.

Analyzing the performance of the Genetic Algorithms in Table B.2 it is visible that its performance was worse than Particle Swarm Optimization, when speaking of the number of iterations needed to converge. The explanation is the fact that PSO is a much more focused search, while GA presents more randomness. The additional randomness in GA results in a larger exploration of the search space, which especially for a higher dimensional problem (as it is the case of the portfolio optimization with 20 assets) results in solutions closer to the global optimum (lower $\overline{\varepsilon_{VaR}}$). It shows that GA seems less likely to converge to local minima than PSO, exception made to the random positioning strategy of PSO. However, the performance of the random positioning strategy for PSO seemed to be worse than the performance of GA, especially for a higher number of assets. Comparing the different operators used in GA, the basic crossover performed better than the arithmetic crossover, which is surprising, because it was expected that the arithmetic crossover would be better to preserve the knowledge existent in the population. However, the whole arithmetic crossover shows an 'averaging effect' on the solutions that make GA lose part of its exploratory power. Comparing the different selection strategies, the tournament selection performed slightly better than the roulette wheel selection, but no strong conclusion can be made. The combination of roulette wheel selection with whole arithmetic crossover showed the worst performance of all the strategies, regarding the number of iterations to converge in all portfolio sizes, and also regarding $\overline{\varepsilon_{VaR}}$ for 20 assets. The possible explanation for this is that this combination reduces the exploratory capacity of the GA, without adding much to the exploitation.

It is also noticeable that calculating $N'_{it} = \overline{N_{it}} + 2\sigma_N$, which gives the number of iterations needed to converge in 97.7% of the times (considering a normal distribution of N_{it}), GA will not have converged until the limit of 2 000 iterations was reached. The values of N'_{it} can be seen in Table B.3. Therefore, the $\overline{\varepsilon_{VaR}}$ observed for GA in the portfolio optimization with 20 assets may be the result of a brute force search. It may be said then that the algorithms may show a better consistency (lower $\overline{\varepsilon_{VaR}}$), but this

comes at a high price (N'_{it}).

To investigate how the solutions found by the algorithms improve through the iterations, the evolution of the best, average and worst solutions were recorded and shown in Figures C.2 (for PSO with bumping strategy), C.3 (for PSO with amnesia strategy), C.4 (for PSO with random positioning strategy), C.5 (for GA with roulette wheel selection and whole arithmetic crossover), and C.6 (for GA with tournament selection and basic crossover). It is also presented the largest, average and smallest Euclidian distance between the particles/chromosomes in each iteration. The Euclidian distance between two points $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$ in the n-dimensional space is defined as:

$$d = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}. \tag{5.1}$$

Through the use of the Euclidian distance it is possible to visualize if the algorithm is converging to a point, or if it is still searching for solutions in different areas of the solution space. It shows this way how is the trade-off between exploration and exploitation in a strategy. The same set of 5 assets (3M, Citigroup, Coca-Cola, Gm and Microsoft) was used for generating the pictures, running with a population of 50 chromosomes/particles. The graphs show the results of a specific run, which is considered to be a typical run. It should also be noticed that the evolution of the best solution refers to the best solution in the population. For GA this graph will always be decreasing, as elitism is used. For PSO, this is not the case, however if a better solution was found in a past iteration, it is recorded as the global best and will not be lost, even if none of the particles is located at that point in ulterior iterations.

Figure C.2a shows the evolution of the fitness of the best individual, the average individual, and the worst individual of the population using PSO with bumping strategy. Visibly the search is very focused, with all the population quickly converging to the best solution. Figure C.2b shows this convergence where all particles eventually converge to the optimal solution, except from an occasional wandering of some particles in early iterations.

Figure C.3 shows the evolution of the solutions of PSO using amnesia strategy. It is clear in Figure C.3b that in the beginning of the optimization some particles are out of the feasible solution space. This may be seen by noticing that the maximum distance between two particles inside the feasible solution space is $\sqrt{2} \approx 1.41$ (see equation 5.1, remembering that the sum of the weights is equal to one) and right in the first iterations the biggest distance is larger than that. The penalty function strategy gives

similar results and because of this was not presented.

Figure C.4 shows the performance of the PSO using random positioning strategy. It is visible in the pictures how it presents problems to converge to a solution. Both the largest and the average distance between the particles, seen in Figure C.4b are much bigger than those seen in the other strategies. The evolution of the solutions seen in Figure C.4a also shows that although the average VaR of the population seems to be decreasing, the convergence is very slow.

Figure C.5a shows the evolution of the fitness of the population using roulette wheel selection with arithmetic crossover. It is visible that the VaR of the average solution approaches quickly the VaR of the best solution, provoking a lack of diversity. This is better seen in Figure C.5b, where the distance between the chromosomes (i.e. the diversity of the population) is much lower than what is observed for example in Figure C.6, which shows a typical run of GA for the same assets, but with tournament selection and basic crossover.

5.3 Speed of algorithms

The second test performed was a test designed to measure the speed of the algorithms, as described in section 4.3. PSO and GA were executed again with 50 particles/chromosomes, for portfolios with 5, 10 and 20 assets. The results are presented in Table B.3. For all sizes of portfolios, the Particle Swarm Optimization needed less time per iteration than Genetic Algorithms, and the difference in the time needed per iteration increases with the number of assets in the portfolio. For portfolios with 5 assets, GA needed circa 30% more time per iteration; for portfolios with 10 assets, it needed 45% more; and for portfolios with 20 assets, it needed 70% more. The average time to solve the optimization problem is much lower for PSO than for GA, which is not surprising given the lower time per iteration needed in PSO, and the higher number of iterations needed to solve the problem (N'_{it}) for GA.

5.4 Sensitivity to initial position

One problem with PSO is the sensitivity to the initial position of the particles. It is a more focused search, with less randomness in the process, so if some of the

particles are not initialized near the region where the global optimum is located, it is unlikely that the algorithm will ever converge to it. To evaluate this sensitivity, another consistency test was executed, but now instead of using the procedure for initializing the position of the particles described in section 4.2, a worse one was used. In the start of each run, one of the 'corners' of the feasible solution space was chosen and the particles were initialized concentrated in that area. Figure 5.2a shows a typical initial position generated for a portfolio with 3 assets using the procedure described in section 4.2 and Figure 5.2b shows an initial position of particles generated by concentrating the particles in a small area of the feasible solution space (bad initialization). The results of the experiment are shown in Table B.4. It is visible that as the number of dimensions in the problem increase, the performance of the PSO is degraded. For portfolios with 20 assets, the $\overline{\varepsilon_{VaR}}$ of PSO is over 70%, which shows that the algorithm is not capable of finding the global optimum when there is a bad initialization of the particles. An exception is made for the random positioning strategy, that can overcome this problem, by the added randomness introduced through the boundaries handling strategy, but for large portfolios this strategy was barely able to converge at all ($\overline{N_{it}}$ was very close to the maximum number of iterations allowed), showing that the good results for $\overline{\varepsilon_{VaR}}$ were mainly due to brute force random search. GA presented a performance similar to the one obtained with a good initialization of the chromosomes, regarding $\overline{\varepsilon_{VaR}}$, showing that the increased exploration of the search space provided by the algorithm makes it more robust to bad initial positions. However, the number of iterations needed to converge was higher than the one needed for a good initialization of the chromosomes.

5.5 Influence of the number of particles / chromosomes

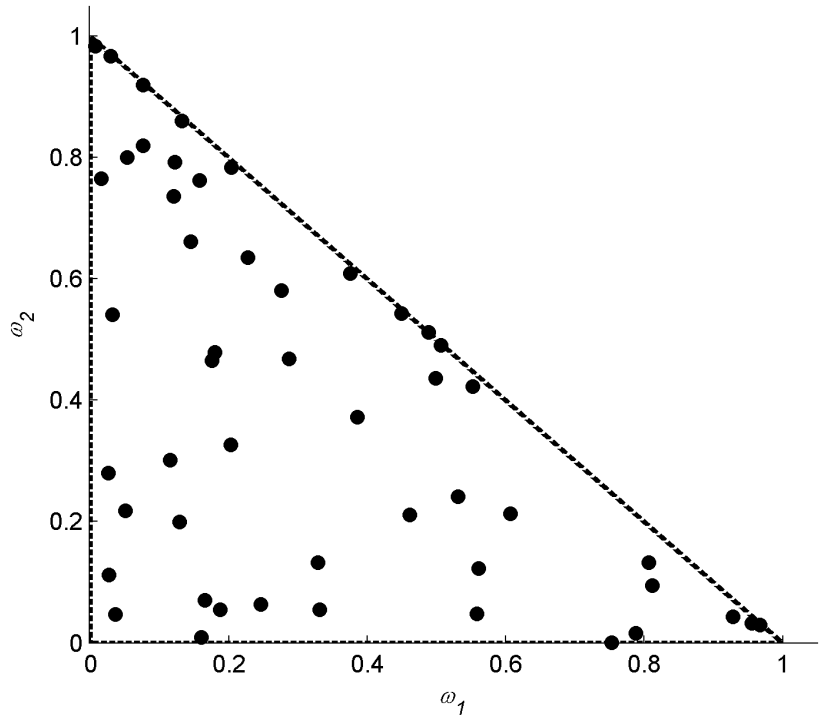
Even though the aforementioned work of Li-Ping *et al.* (2005) recommend a population not larger than 50 particles for PSO (with 30 particles being a good choice), it was noticed during the experiments that the number of particles used could play an important role. To further investigate this question, the consistency of the algorithms was tested using portfolios with 10 assets and with different number of particles/chromosomes: 10, 50 and 250. The results are presented in Table B.5. $\overline{\varepsilon_{VaR}}$ and the associated standard deviation (σ_ε) decrease significatively with the increase in the the number of particles. This effect was expected as a larger region of the solution space

is explored in the initial iterations, when a larger number of particles is present. The number of iterations needed to converge decreases slightly for PSO using the bumping, amnesia and penalty function strategies when increasing the number of particles. This happens because as there are more particles, if they are well spread in the solution space it is more likely that one of them will be closer to the global optimum, and thus less iterations are needed to reach it. PSO using random positioning strategy showed no improvement in the number of iterations to converge, reinforcing the idea that it finds the best solution mainly due to random search. For GA, increasing the number of particles did not seem to alter much the number of iterations needed to converge, except for the combination tournament selection/basic crossover, which had a large reduction in both $\overline{\varepsilon_{VaR}}$ and σ_ε. This combination showed to be the best for GA in the previous experiments, but further research should be made to investigate the reason of the significant increase of performance when running with 250 particles.

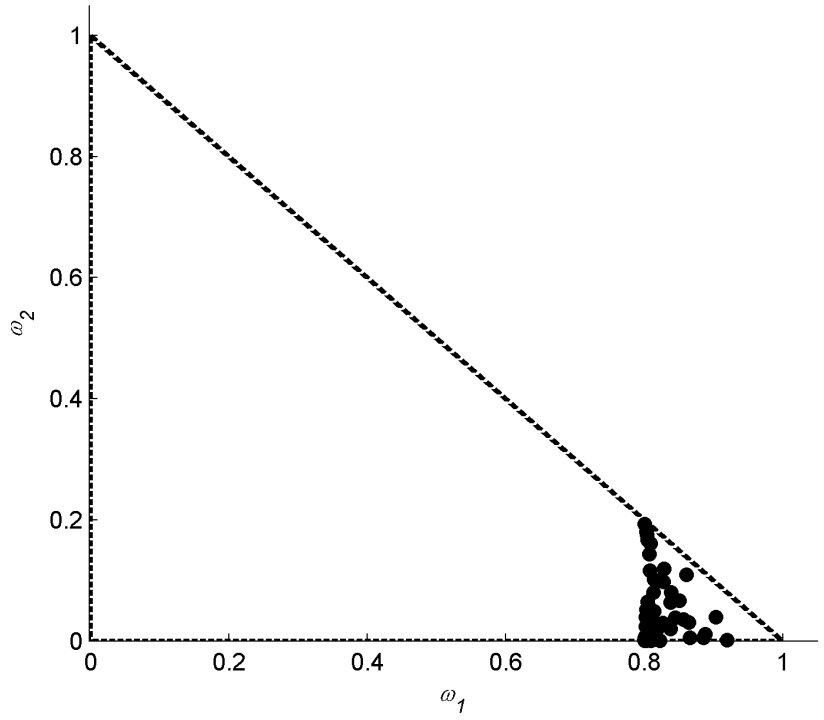
Another question that was investigated is whether it is better to use a large number of particles and few iterations or a few number of particles and run the algorithm for a large number of iterations. To try to answer this question, a consistency test was run, for portfolios of 5, 10 and 20 assets, running with 2 000 particles for a maximum of 50 iterations, with the results shown in Table B.6. A speed test based on the results of the consistency test was also run (see Table B.7). The idea of running only for a maximum of 50 iteration was to compare equivalent methods, in the sense that running with 50 particles and 2 000 iterations a maximum of 10 000 portfolios are checked and running with 2 000 particles for 50 iterations the same number of portfolios are visited.

Analyzing the results it is seen through $\overline{\varepsilon_{VaR}}$ and σ_ε that the ability to get near the global optimum increased considerably, confirming what was already seen in Table B.5. PSO with random positioning strategy and GA with roulette wheel selection and basic crossover showed a performance that is worse than the others for large portfolios. The decay in comparative performance for PSO with random positioning happens because with a large number of particles, there is already a big exploration of the solution space in early stages of the search, so the main strength of random positioning (exploration) is not so strongly needed. The results on Table B.7 show that, except for PSO using random positioning strategy, the time needed to converge is much larger using 2 000 particles than using 50 particles. This can explained by the fact that in the first iterations, PSO still has a big influence of the random velocity with which the particles were initialized, due to the momentum term. Just after a certain number of iterations, the velocities will better reflect the attraction exerted by the local and global bests. However the time per iteration is much larger when using a large number of particles,

and in this way the total execution time of the algorithms will be penalized by the initial 'non-focused' search. For GA a similar explanation exists, as in the beginning the diversity in the population is large and it is very likely that offsprings will be generated from not very fit individuals, wasting precious time in the beginning of the search. So, the idea of visiting the same number of portfolios by reducing the number of iterations and increasing the number of particles/chromosomes do not add to the algorithm performance.



(a) Good initialization: particles are well spread in the feasible solution space



(b) Bad initialization: particles are concentrated in a region

Figure 5.2: Example of random initial position of particles

Chapter 6

Conclusions

In this Thesis, it was shown the application of Particle Swarm Optimization and Genetic Algorithms to risk management, in a constrained portfolio optimization problem where no short sales are allowed. The objective function to be minimized was the value at risk calculated using historical simulation.

Several strategies to handle the constraints were implemented and the results were compared. For PSO, it was implemented the strategies bumping, amnesia, random positioning and penalty function. For GA, two selection operators (roulette wheel and tournament); two crossover operators (basic crossover and arithmetic crossover); and a mutation operator were implemented.

The results showed that the methods are capable of finding good solutions in a reasonable amount of time. PSO showed to be faster than GA, both in terms of number of iterations and in terms of total running time, which is explained by the more focused search performed by PSO. However, PSO demonstrated to be much more sensible to the initial position of the particles than GA, and if a bad initialization is made, it is very likely that PSO will not converge to the global optimum.

Regarding the strategies used for PSO, bumping seems to be the best in terms of speed, followed closely by amnesia and penalty function. The random positioning strategy did not perform well in this sense, presenting problems to converge to the best solution, although it was more robust to the initial position of the particles and the superior exploratory power allowed a good consistency comparing to the others.

GA showed to be able to find good solutions too, but was worse than PSO in terms of speed. However, its less focused search (larger randomness) makes it less prone to

be trapped into local minima, especially if the population is not initialized with the chromosomes well spread in the feasible solution space. The basic crossover showed to be better than the whole arithmetic crossover, preserving the diversity of the population and thus the exploration of the solution space. The combination tournament selection with basic crossover was somewhat better than the other strategies and, in the other hand, the combination roulette wheel selection with whole arithmetic crossover was the worst.

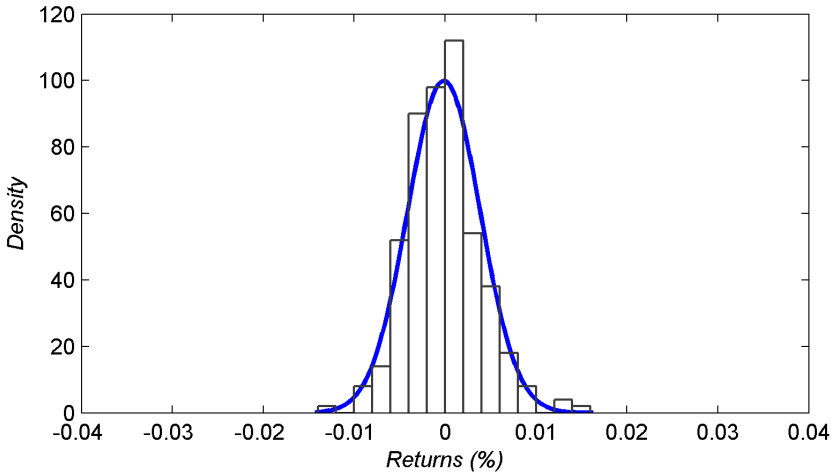
Tests were also made regarding the number of particles needed to solve the problem, and apparently using a big number of particles increments the consistency of the algorithms to find the global optimum, but at the cost of a big increase in computational time. Using 50 particles/chromosomes seemed to be enough for problems up to 20 assets. The bottom line is that consistency of the algorithms to find the global optimum can be increased, but this comes at a high price: longer execution times.

As suggestions for further research, it is proposed:

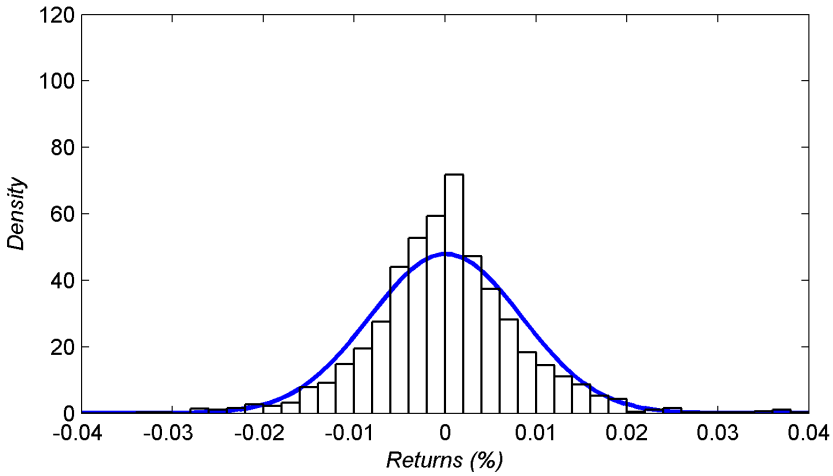
- Evaluate the performance of the algorithms with the inclusion of criteria to detect on-the-fly the convergence.
- Find a reformulation of the optimization problem, such that the solution found is less sensible to the use of different horizons of data, i.e. use a multi-criteria optimization that minimizes the objective function calculated over different horizons of data.
- Investigate how to use PSO/GA to add adaptability to portfolio management when dealing with different horizons.
- Check the effect of including the Constriction Coefficient proposed by Clerc (1999) in the performance of PSO for risk management.
- Test different encodings of the solutions for GA, for example the more traditional binary encoding.
- Compare PSO and GA with traditional methods.

Appendix A

Summary of the data used



(a) Including 1 year of historical data



(b) Including 10 years of historical data

Figure A.1: Distribution of daily returns of Verizon, for two different horizons

Table A.1: Average returns of the companies used in the experiments, for different horizons of the data, with daily frequency

Company	*1 year*	*2 year*	*5 year*	*10 year*
3M	0.018%	0.002%	0.015%	0.020%
Alcoa	0.026%	0.007%	-0.008%	0.016%
Altria	0.018%	0.042%	0.021%	0.021%
American Express	0.024%	0.018%	0.015%	0.027%
Am. International	0.014%	-0.016%	-0.009%	0.018%
AT&T	0.027%	0.016%	-0.009%	0.007%
Boeing	0.047%	0.053%	0.011%	0.013%
Caterpillar	0.075%	0.060%	0.038%	0.029%
Citigroup	0.010%	0.011%	0.005%	0.031%
Coca-Cola	0.001%	-0.010%	0.001%	0.001%
DuPont	-0.013%	0.005%	0.002%	0.006%
Exxon	0.013%	0.032%	0.014%	0.022%
GE	-0.009%	0.013%	-0.008%	0.018%
GM	-0.020%	-0.038%	-0.019%	-0.001%
Home Depot	-0.008%	0.007%	-0.008%	0.021%
Honeywell	0.024%	0.022%	-0.002%	0.010%
HP	0.061%	0.038%	0.006%	0.011%
IBM	0.009%	-0.006%	-0.012%	0.022%
Intel	-0.071%	-0.035%	-0.017%	0.013%
Johnson & Johnson	-0.017%	0.008%	0.008%	0.019%
JP Morgan	0.036%	0.018%	0.001%	0.013%
McDonalds	0.013%	0.022%	0.006%	0.008%
Merck	0.012%	-0.025%	-0.020%	0.006%
Microsoft	-0.017%	0.002%	-0.011%	0.021%
Pfizer	-0.025%	-0.030%	-0.017%	0.014%
Procter & Gamble	-0.002%	0.004%	0.022%	0.018%
United	0.029%	0.036%	0.017%	0.028%
Verizon	-0.015%	-0.003%	-0.013%	0.006%
Wal-Mart	0.003%	-0.012%	0.000%	0.024%
Walt Disney	0.018%	0.022%	0.000%	0.008%

Table A.2: Standard deviations of the returns of the companies used in the experiments, for different horizons of the data, with daily frequency

Company	*1 year*	*2 year*	*5 year*	*10 year*
3M	0.429%	0.477%	0.588%	0.708%
Alcoa	0.674%	0.665%	0.908%	0.975%
Altria	0.459%	0.489%	0.735%	0.919%
American Express	0.464%	0.429%	0.797%	0.945%
Am. International	0.425%	0.597%	0.771%	0.823%
AT&T	0.388%	0.400%	0.789%	0.870%
Boeing	0.566%	0.560%	0.845%	0.930%
Caterpillar	0.691%	0.655%	0.787%	0.916%
Citigroup	0.352%	0.377%	0.781%	0.968%
Coca-Cola	0.315%	0.389%	0.542%	0.733%
DuPont	0.470%	0.466%	0.666%	0.829%
Exxon	0.560%	0.550%	0.627%	0.678%
GE	0.369%	0.384%	0.755%	0.809%
GM	1.253%	1.087%	1.021%	0.969%
Home Depot	0.531%	0.535%	0.882%	1.024%
Honeywell	0.537%	0.554%	0.953%	1.024%
HP	0.743%	0.753%	1.095%	1.238%
IBM	0.394%	0.436%	0.709%	0.916%
Intel	0.654%	0.713%	1.143%	1.288%
Johnson & Johnson	0.380%	0.380%	0.581%	0.681%
JP Morgan	0.394%	0.406%	0.903%	1.009%
McDonalds	0.619%	0.551%	0.752%	0.801%
Merck	0.608%	0.918%	0.831%	0.848%
Microsoft	0.541%	0.482%	0.802%	0.990%
Pfizer	0.608%	0.649%	0.737%	0.870%
Procter & Gamble	0.374%	0.402%	0.486%	0.774%
United	0.453%	0.449%	0.817%	0.847%
Verizon	0.399%	0.415%	0.740%	0.831%
Wal-Mart	0.448%	0.422%	0.625%	0.865%
Walt Disney	0.509%	0.505%	0.904%	0.961%

Appendix B

Tables with results

Table B.1: Comparison of different risk measures. Portfolios with the same 5 assets were optimized to minimize VaR, CVaR and the Variance. The portfolios obtained were then measured using different metrics. *Minimized* indicates which objective function was minimized; *Variance*, *VaR* and *CVaR* indicate the average deviation from the portfolio which minimized these criteria

Minimized	*Variance*	*VaR*	*CVaR*
Variance	-	5.56%	0.12%
VaR	0.77%	-	0.49%
CVaR	0.09%	5.09%	-

Table B.2: Comparison of the *consistency* of PSO and GA algorithms for solving the portfolio optimization problem, running with 50 particles/chromosomes and a maximum of 2000 iterations. N_a is the number of assets included in the portfolio; *Algorithm* identifies to which algorithm the listed results refer to; $\overline{N_{it}}$ is the average number of iterations that the algorithm took to find a solution within 0.1% of the minimum VaR found in a specific run; σ_N is the standard deviation of this measure; $\overline{\varepsilon_{VaR}}$ is the average error between the VaR found by the algorithm in the different runs and the minimum VaR found over all runs; and σ_ε is the standard deviation of these errors

N_a	*Algorithm*	$\overline{N_{it}}$	σ_N	$\overline{\varepsilon_{VaR}}$	σ_ε
5	PSO Bumping	44.0	21.0	0.58%	0.79%
	PSO Amnesia	79.7	31.0	0.55%	0.73%
	PSO Random	562.7	522.6	0.37%	0.60%
	PSO Penalty	93.8	54.1	0.37%	0.69%
	GA Roul./Basic	142.0	168.7	0.52%	0.72%
	GA Tourn./Basic	172.6	242.1	0.60%	0.78%
	GA Roul./Arith.	472.6	574.2	0.91%	1.06%
	GA Tourn./Arith.	269.1	415.0	1.02%	1.02%
10	PSO Bumping	102.1	53.3	3.43%	1.38%
	PSO Amnesia	163.4	103.8	4.02%	2.30%
	PSO Random	1473.3	427.5	2.29%	1.41%
	PSO Penalty	190.4	88.1	3.34%	2.49%
	GA Roul./Basic	793.0	518.1	2.65%	1.76%
	GA Tourn./Basic	680.5	507.2	3.37%	1.79%
	GA Roul./Arith.	1257.0	454.5	3.12%	1.96%
	GA Tourn./Arith.	808.8	550.8	3.66%	1.76%
20	PSO Bumping	119.1	52.1	5.27%	2.50%
	PSO Amnesia	320.6	96.8	6.77%	2.31%
	PSO Random	1798.8	272.7	4.99%	2.40%
	PSO Penalty	299.6	65.4	6.29%	3.26%
	GA Roul./Basic	1239.9	506.4	3.62%	2.24%
	GA Tourn./Basic	1078.4	510.8	3.46%	2.28%
	GA Roul./Arith.	1615.2	272.6	5.72%	2.05%
	GA Tourn./Arith.	1298.2	365.8	4.60%	2.78%

Table B.3: Comparison of the *speed* of the PSO and GA algorithms for solving the portfolio optimization problem, running with 50 particles/chromosomes. N_a is the number of assets included in the portfolio; *Algorithm* identifies to which algorithm the listed results refer to; $\overline{t/it}$ is the average time per iteration, for the given number of assets and particles; N'_{it} is the fixed number of iterations used to solve the problem; and $\bar{t}$ is the average time in seconds to solve the optimization problem

N_a	*Algorithm*	$\overline{t/it}$ (ms)	N'_{it}	$\bar{t}$ (s)
5	PSO Bumping	34.2	86	2.9
	PSO Amnesia	33.9	142	4.8
	PSO Random	33.9	1 608	54.5
	PSO Penalty	34.5	202	7.0
	GA Roul./Basic	44.2	479	21.2
	GA Tourn./Basic	43.7	657	28.7
	GA Roul./Arith.	43.0	1 621	69.7
	GA Tourn./Arith	42.6	1 099	46.8
10	PSO Bumping	46.5	209	9.7
	PSO Amnesia	46.0	371	17.1
	PSO Random	46.0	2 328	107.1
	PSO Penalty	46.6	367	17.1
	GA Roul./Basic	69.6	1 829	127.3
	GA Tourn./Basic	69.2	1 695	117.2
	GA Roul./Arith.	68.3	2 166	148.0
	GA Tourn./Arith	68.0	1 910	129.9
20	PSO Bumping	70.8	223	15.8
	PSO Amnesia	70.1	514	36.0
	PSO Random	70.5	2 344	165.1
	PSO Penalty	71.2	430	30.6
	GA Roul./Basic	121.6	2 253	274.1
	GA Tourn./Basic	121.3	2 100	254.6
	GA Roul./Arith.	120.0	2 160	259.3
	GA Tourn./Arith	120.1	2 030	243.7

Table B.4: Comparison of the influence of the *initial position* of the particles on the *consistency* of the PSO and GA algorithms for solving the portfolio optimization problem, running with 50 particles/chromosomes and a maximum of 2000 iterations. The particles were initially concentrated in an area of the search space. N_a is the number of assets included in the portfolio; *Algorithm* identifies to which algorithm the listed results refer to; $\overline{N_{it}}$ is the average number of iterations that the algorithm took to find a solution within 0.1% of the minimum VaR found in a specific run; σ_N is the standard deviation of this measure; $\overline{\varepsilon_{VaR}}$ is the average error between the VaR found by the algorithm in the different runs and the minimum VaR found over all runs; and σ_ε is the standard deviation of these errors

N_a	*Algorithm*	$\overline{N_{it}}$	σ_N	$\overline{\varepsilon_{VaR}}$	σ_ε
5	PSO Bumping	82.1	80.2	4.02%	4.74%
	PSO Amnesia	74.2	42.4	2.10%	3.65%
	PSO Random	262.5	292.1	0.26%	0.47%
	PSO Penalty	68.3	23.4	2.31%	3.73%
	GA Roul./Basic	410.3	441.9	0.51%	0.87%
	GA Tourn./Basic	390.3	519.2	0.96%	1.19%
	GA Roul./Arith.	621.3	461.9	1.90%	2.11%
	GA Tourn./Arith	398.3	427.1	1.26%	1.51%
10	PSO Bumping	202.9	136.0	36.59%	27.92%
	PSO Amnesia	194.7	76.0	12.72%	12.19%
	PSO Random	1455.9	441.7	2.63%	1.78%
	PSO Penalty	208.7	106.6	13.87%	15.46%
	GA Roul./Basic	787.3	502.4	2.78%	2.36%
	GA Tourn./Basic	636.0	465.1	4.17%	2.78%
	GA Roul./Arith.	1087.8	493.3	3.64%	1.99%
	GA Tourn./Arith	951.1	499.1	3.78%	2.34%
20	PSO Bumping	107.5	77.4	76.87%	54.68%
	PSO Amnesia	319.0	90.6	70.13%	38.39%
	PSO Random	1726.7	142.7	3.92%	2.06%
	PSO Penalty	320.2	86.0	74.45%	56.72%
	GA Roul./Basic	1371.2	486.1	5.47%	4.03%
	GA Tourn./Basic	933.3	389.0	4.56%	2.99%
	GA Roul./Arith.	1492.9	341.7	6.60%	3.51%
	GA Tourn./Arith	1318.0	480.3	6.11%	3.68%

Table B.5: Comparison of the influence of the *number of particles* on the *consistency* of the PSO and GA algorithms for solving the portfolio optimization problem, running with 10 assets and a maximum of 2000 iterations. N_p is the number of particles/chromosomes used; *Algorithm* identifies to which algorithm the listed results refer to; $\overline{N_{it}}$ is the average number of iterations that the algorithm took to find a solution within 0.1% of the minimum VaR found in a specific run; σ_N is the standard deviation of this measure; $\overline{\varepsilon_{VaR}}$ is the average error between the VaR found by the algorithm in the different runs and the minimum VaR found over all runs; and σ_ε is the standard deviation of these errors

N_p	*Algorithm*	$\overline{N_{it}}$	σ_N	$\overline{\varepsilon_{VaR}}$	σ_ε
10	PSO Bumping	105.7	44.4	6.54%	2.89%
	PSO Amnesia	153.6	44.2	6.04%	3.52%
	PSO Random	1519.5	410.6	3.62%	1.86%
	PSO Penalty	153.6	37.6	6.26%	3.31%
	GA Roul./Basic	797.0	539.2	4.62%	2.62%
	GA Tourn./Basic	863.6	507.4	4.22%	2.27%
	GA Roul./Arith.	1063.9	451.6	4.09%	2.12%
	GA Tourn./Arith	1036.2	493.3	3.78%	2.33%
50	PSO Bumping	102.1	53.3	3.43%	1.38%
	PSO Amnesia	163.4	103.8	4.02%	2.30%
	PSO Random	1473.3	427.5	2.29%	1.41%
	PSO Penalty	190.4	88.1	3.34%	2.49%
	GA Roul./Basic	793.0	518.1	2.65%	1.76%
	GA Tourn./Basic	680.5	507.2	3.37%	1.79%
	GA Roul./Arith.	1257.0	454.5	3.12%	1.96%
	GA Tourn./Arith.	808.8	550.8	3.66%	1.76%
250	PSO Bumping	58.8	38.9	1.27%	0.97%
	PSO Amnesia	137.8	68.3	1.55%	1.10%
	PSO Random	1494.4	341.6	1.81%	1.03%
	PSO Penalty	126.6	43.8	1.46%	1.04%
	GA Roul./Basic	650.0	414.0	0.62%	0.88%
	GA Tourn./Basic	231.9	203.6	1.09%	0.90%
	GA Roul./Arith.	1341.2	410.9	2.74%	1.12%
	GA Tourn./Arith	1047.8	601.7	2.66%	1.04%

Table B.6: Comparison of the *consistency* of PSO and GA algorithms for solving the portfolio optimization problem, running with 2000 particles/chromosomes and a maximum of 50 iterations. N_a is the number of assets included in the portfolio; *Algorithm* identifies to which algorithm the listed results refer to; $\overline{N_{it}}$ is the average number of iterations that the algorithm took to find a solution within 0.1% of the minimum VaR found in a specific run; σ_N is the standard deviation of this measure; $\overline{\varepsilon_{VaR}}$ is the average error between the VaR found by the algorithm in the different runs and the minimum VaR found over all runs; and σ_ε is the standard deviation of these errors

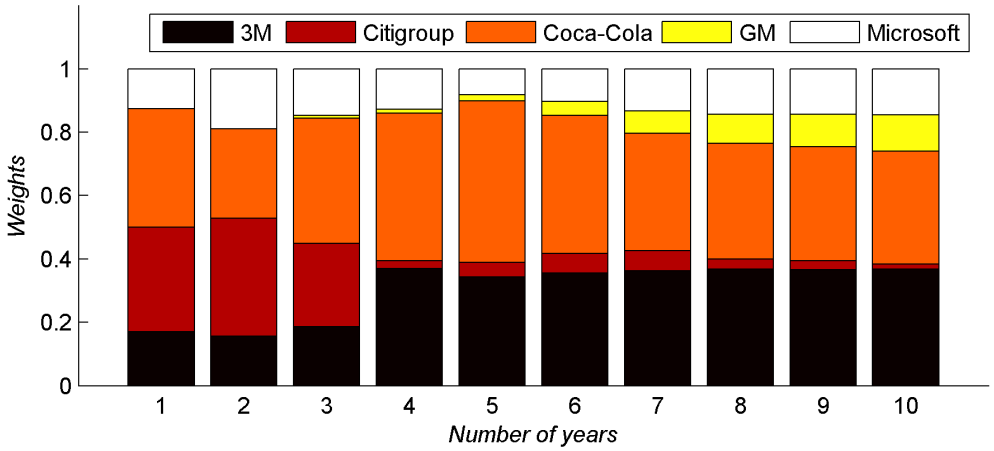
N_a	*Algorithm*	$\overline{N_{it}}$	σ_N	$\overline{\varepsilon_{VaR}}$	σ_ε
5	PSO Bumping	13.5	4.4	0.04%	0.08%
	PSO Amnesia	19.0	3.3	0.10%	0.14%
	PSO Random	24.1	3.9	0.17%	0.26%
	PSO Penalty	19.3	3.3	0.20%	0.38%
	GA Roul./Basic	34.1	9.4	0.53%	0.29%
	GA Tourn./Basic	19.0	8.6	0.41%	0.39%
	GA Roul./Arith.	19.2	11.9	1.22%	0.66%
	GA Tourn./Arith	16.5	11.4	0.88%	0.64%
10	PSO Bumping	24.2	4.7	1.26%	1.14%
	PSO Amnesia	34.6	4.0	1.55%	1.35%
	PSO Random	44.6	4.5	2.08%	1.25%
	PSO Penalty	35.3	3.2	1.57%	1.26%
	GA Roul./Basic	41.9	6.3	2.55%	0.97%
	GA Tourn./Basic	38.1	6.9	1.41%	1.10%
	GA Roul./Arith.	15.2	10.3	5.85%	2.24%
	GA Tourn./Arith	29.7	14.7	3.31%	1.20%
20	PSO Bumping	32.0	3.5	2.87%	2.17%
	PSO Amnesia	47.1	1.7	3.87%	1.87%
	PSO Random	48.5	1.8	6.25%	1.80%
	PSO Penalty	46.8	1.8	3.99%	2.39%
	GA Roul./Basic	45.6	4.6	5.32%	1.34%
	GA Tourn./Basic	46.8	2.7	1.97%	1.25%
	GA Roul./Arith.	22.1	12.4	8.54%	1.69%
	GA Tourn./Arith	36.9	12.4	5.99%	1.38%

Table B.7: Comparison of the *speed* of the PSO and GA algorithms for solving the portfolio optimization problem, running with 2000 particles/chromosomes. N_a is the number of assets included in the portfolio; *Algorithm* identifies to which algorithm the listed results refer to; $\overline{t/it}$ is the average time per iteration, for the given number of assets and particles; N'_{it} is the fixed number of iterations used to solve the problem; and $\overline{t}$ is the average time in seconds to solve the optimization problem

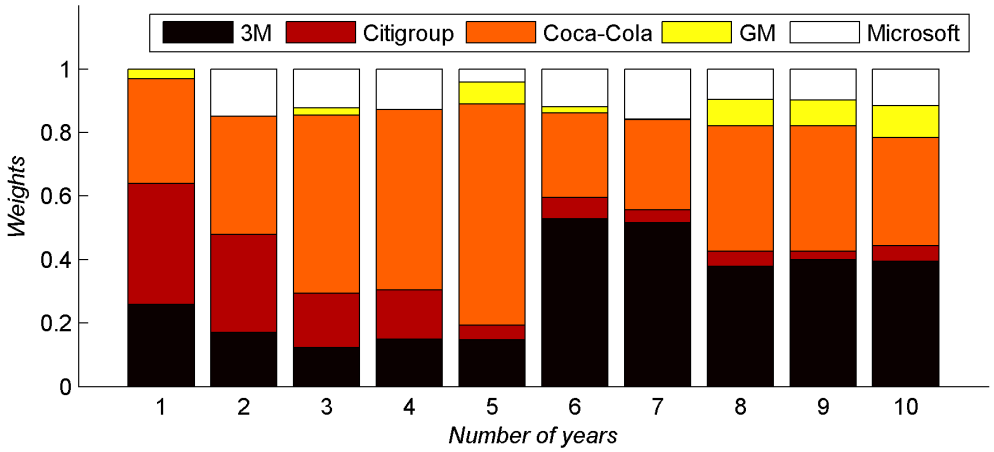
N_a	*Algorithm*	$\overline{t/it}$ (s)	N'_{it}	$\overline{t}$ (s)
5	PSO Bumping	1.36	23	31
	PSO Amnesia	1.36	26	35
	PSO Random	1.36	32	44
	PSO Penalty	1.39	26	36
	GA Roul./Basic	6.44	53	341
	GA Tourn./Basic	6.02	37	223
	GA Roul./Arith.	6.50	43	279
	GA Tourn./Arith	6.02	40	241
10	PSO Bumping	1.85	34	63
	PSO Amnesia	1.85	43	80
	PSO Random	1.85	54	100
	PSO Penalty	1.88	42	79
	GA Roul./Basic	8.49	55	467
	GA Tourn./Basic	8.09	52	421
	GA Roul./Arith.	8.46	36	305
	GA Tourn./Arith	8.15	60	489
20	PSO Bumping	2.86	39	112
	PSO Amnesia	2.85	51	145
	PSO Random	2.86	53	151
	PSO Penalty	2.89	51	148
	GA Roul./Basic	13.31	55	732
	GA Tourn./Basic	12.97	53	687
	GA Roul./Arith.	12.56	47	590
	GA Tourn./Arith	12.36	62	767

Appendix C

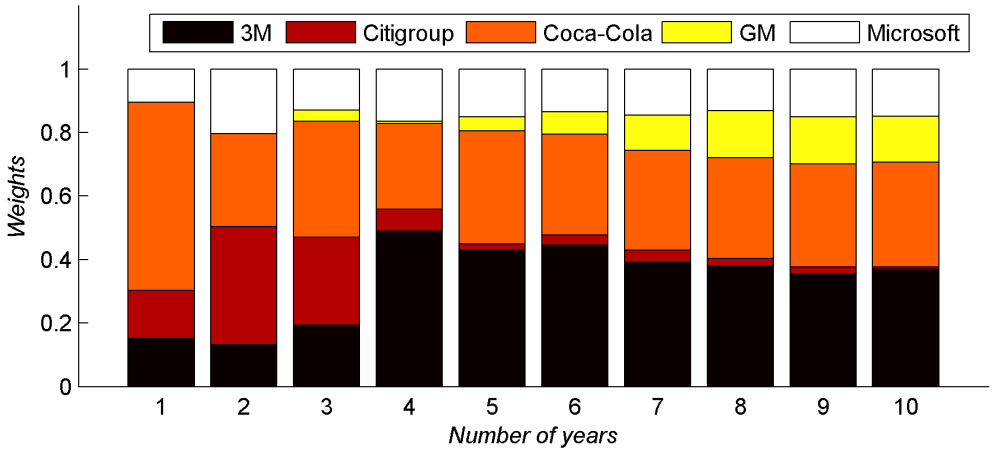
Pictures with results



(a) Minimizing the variance of the returns of the portfolio

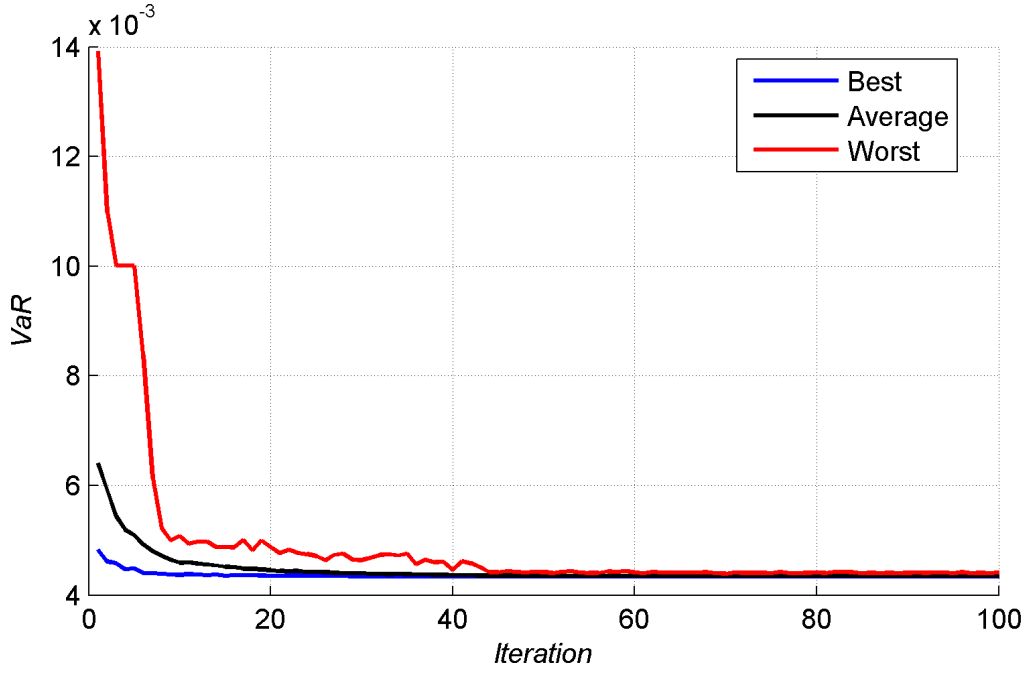


(b) Minimizing the VaR of the returns of the portfolio

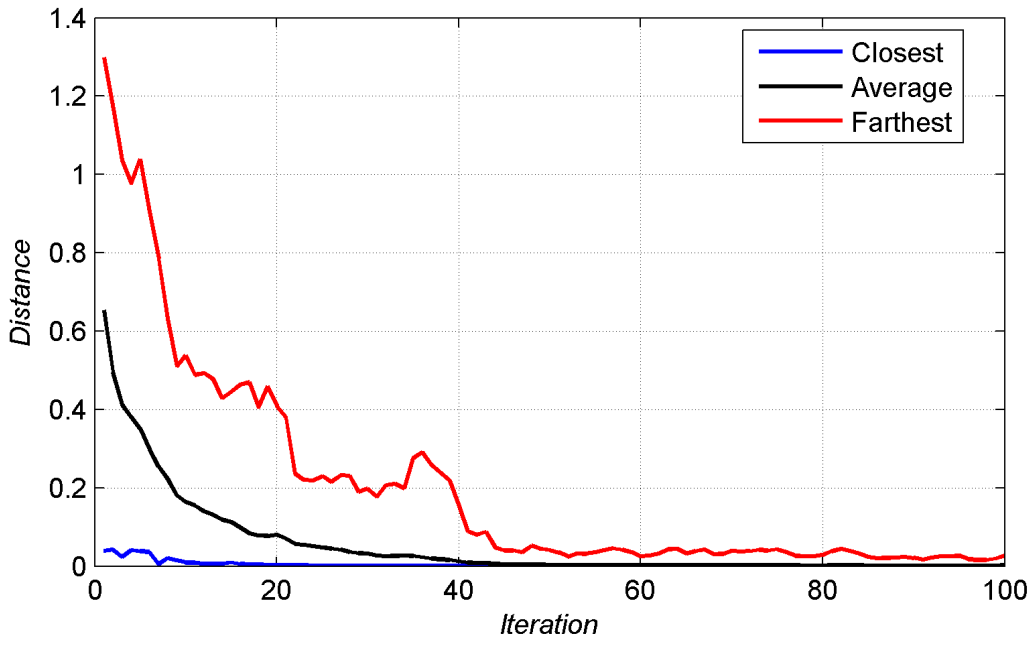


(c) Minimizing the CVaR of the returns of the portfolio

Figure C.1: Optimal portfolio weights, using different objective functions and different horizons for the data

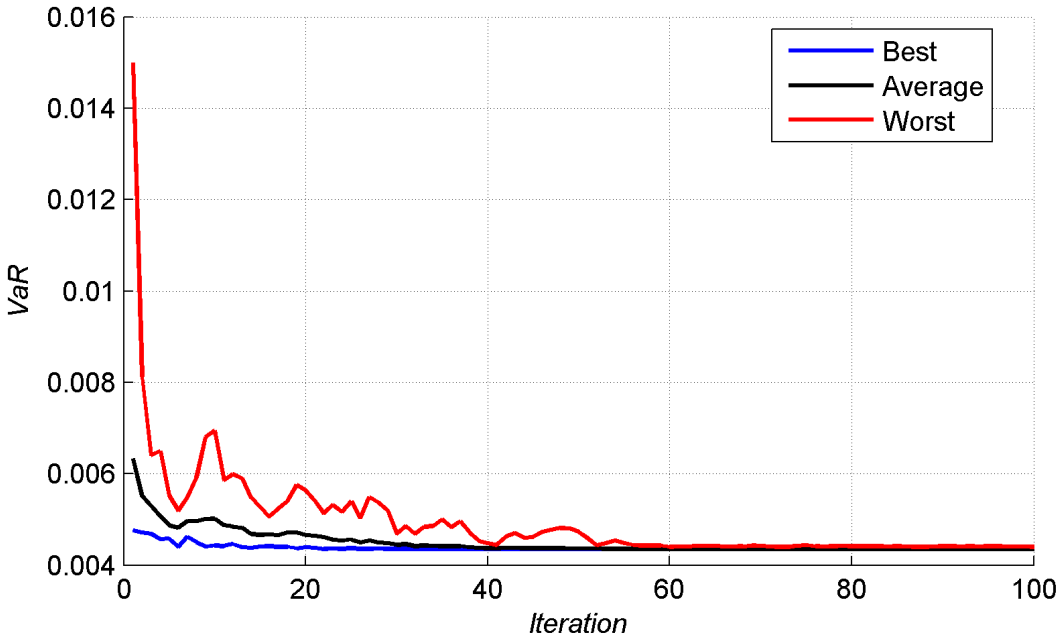


(a) Evolution of the VaR of the particles

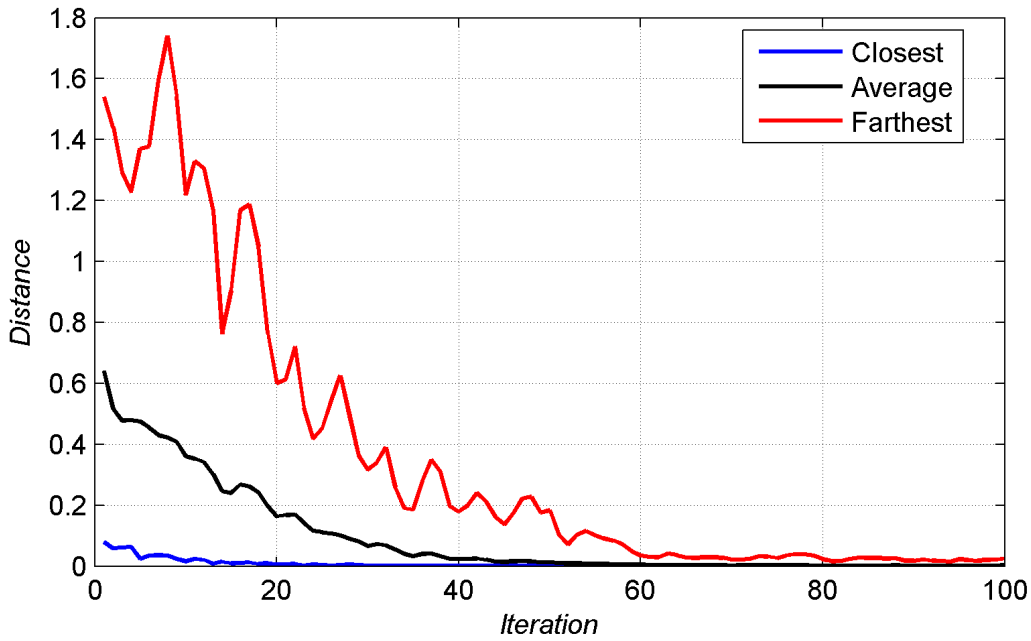


(b) Distance between the particles

Figure C.2: Evolution of solutions for a typical run of PSO using bumping strategy, for 5 assets (3M, Citigroup, Coca-Cola, Gm and Microsoft)

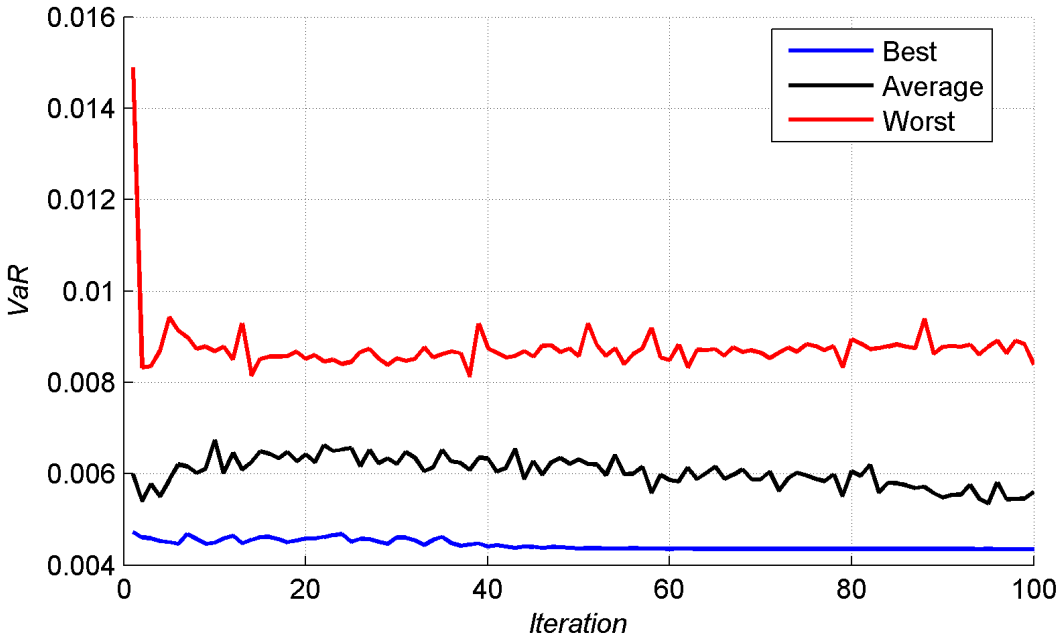


(a) Evolution of the VaR of the particles

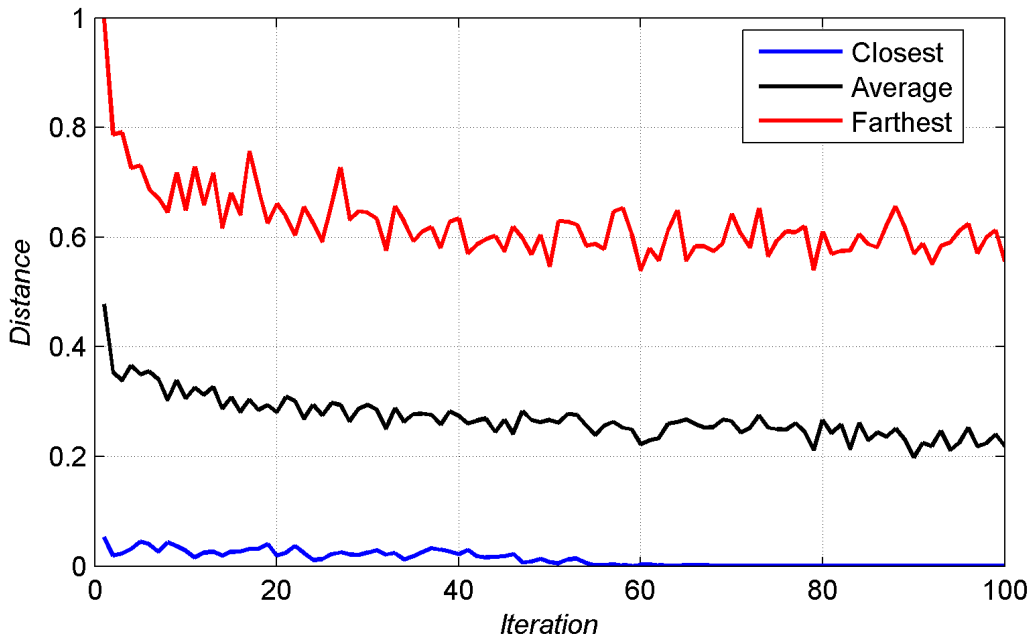


(b) Distance between the particles

Figure C.3: Evolution of solutions for a typical run of PSO using amnesia strategy, for 5 assets (3M, Citigroup, Coca-Cola, Gm and Microsoft)

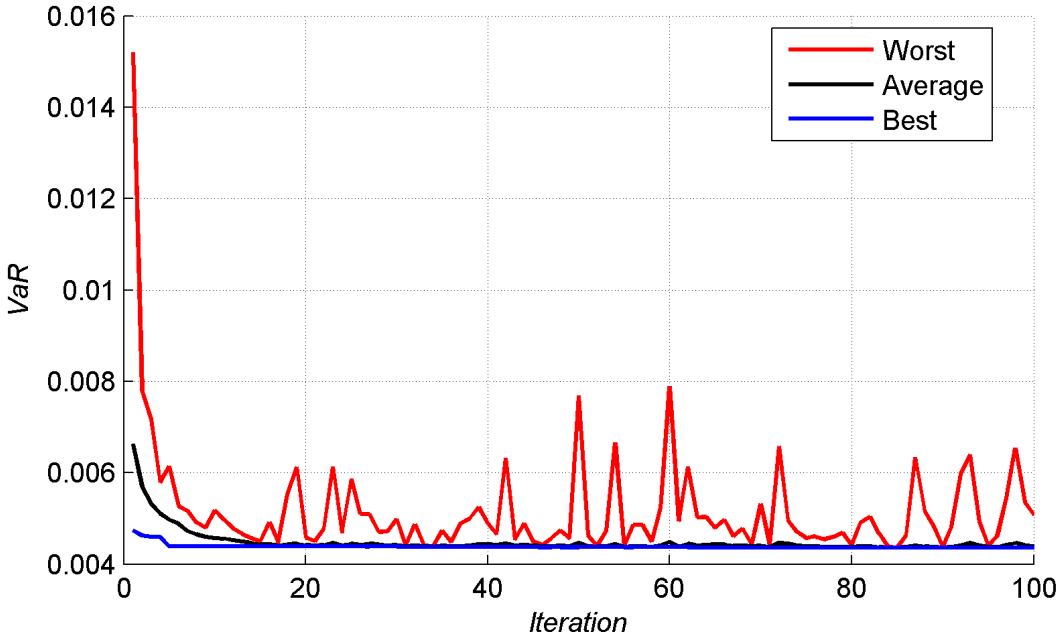


(a) Evolution of the VaR of the particles

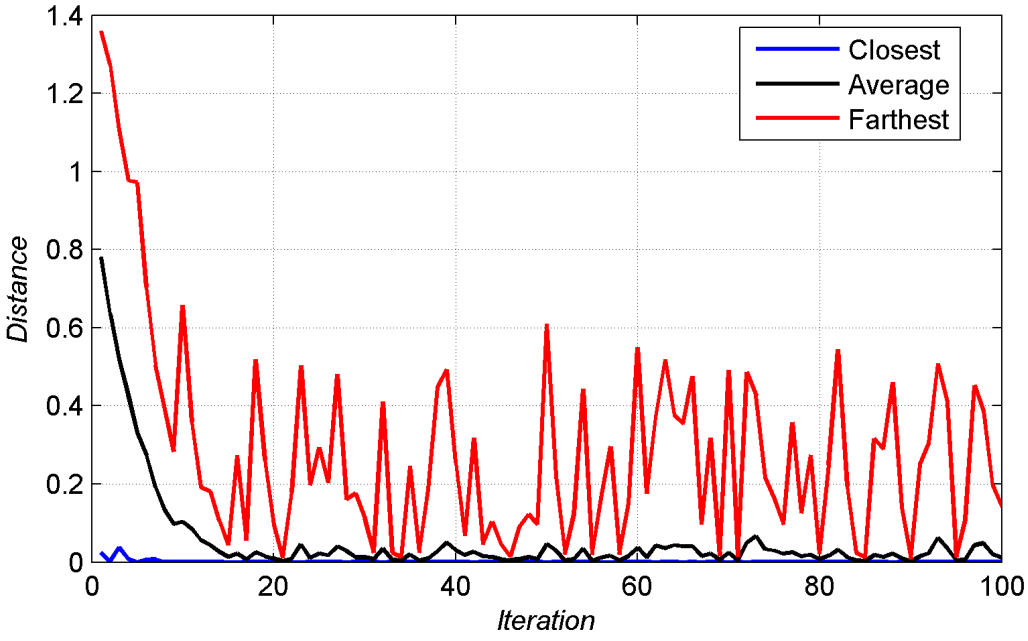


(b) Distance between the particles

Figure C.4: Evolution of solutions for a typical run of PSO using random positioning strategy, for 5 assets (3M, Citigroup, Coca-Cola, Gm and Microsoft)

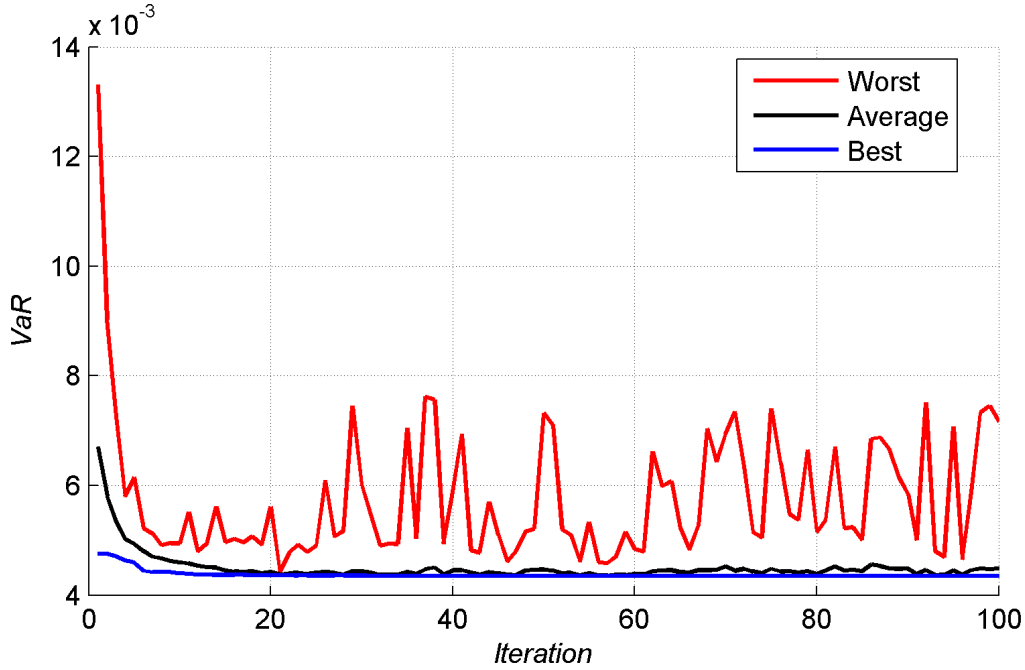


(a) Evolution of the VaR of the particles

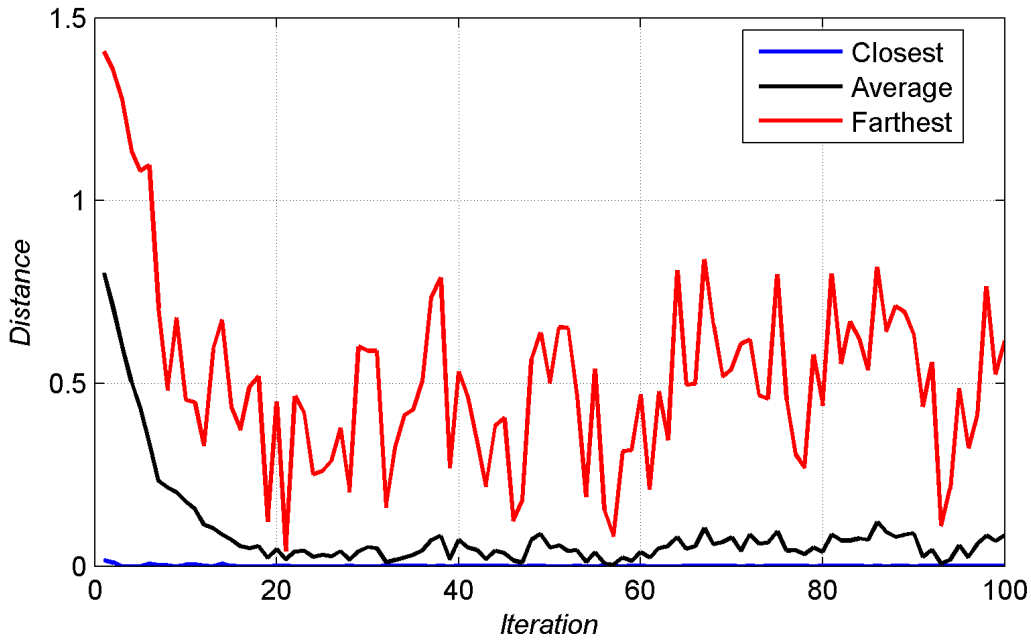


(b) Distance between the particles

Figure C.5: Evolution of solutions for a typical run of GA using roulette wheel selection and whole arithmetic crossover for 5 assets (3M, Citigroup, Coca-Cola, Gm and Microsoft)



(a) Evolution of the VaR of the particles



(b) Distance between the particles

Figure C.6: Evolution of solutions for a typical run of GA using tournament selection and basic crossover for 5 assets (3M, Citigroup, Coca-Cola, Gm and Microsoft)

References

Artzner, P., Delbaen, F., Eber, J.M. & Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, **9**, 203–228.

Arumugam, M.S. & Rao, M. (2004). Novel hybrid approaches for real coded genetic algorithm to compute the optimal control of a single stage hybrid manufacturing systems. *International Journal of Computational Intelligence*, **1**, 231–249.

Best, P.W. (1998). *Implementing Value at Risk*. John Wiley & Sons.

Blanco, A., Delgado, M. & M.C.Pegalajar (2001). A real-coded genetic algorithm for training recurrent neural networks. *Neural Networks*, **14**, 93–105.

Bodie, Z., Kane, A. & Marcus, A.J. (2005). *Investments (International Edition)*. McGraw-Hill, 6th edn.

Chang, T.J., Meade, N., Beasley, J. & Sharaiha, Y. (2000). Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research*, 1271–1302.

Clerc, M. (1999). The swarm and queen: Towards a deterministic and adaptative particle swarm optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, 1951–1957.

Crouhy, M., Galai, D. & Mark, R. (2001). *Risk Management*. McGraw-Hill.

de Roon, F.A. (2001). Testing for mean-variance spanning with short sales constraints and transaction costs: The case of emerging markets. *Journal of Finance*, **56**, 721–742.

Eberhart, R. & Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan*, 39–43.

Eckhardt, R. (1987). Stan Ulam, John von Neumann, and the Monte Carlo method. *Los Alamos Science*, **Special Issue**, 131–137.

Goldberg, D.E. & Debb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In G.J. Rawlins, ed., *Foundations of genetic algorithms*, 69–90, Morgan Kaufmann Publishers.

Hawkins, I. (2000). Choosing appropriate var model parameters and risk-measurement methods. In M. Lore & L. Borodovsky, eds., *The Professional's Handbook of Financial Risk Management*, p. 111–151, Butterworth-Heinemann, Woburn, MA.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Holton, G.A. (2003). *Value-at-Risk: Theory and Practice*. Academic Press.

Holton, G.A. (2004). Defining risk. *Financial Analysts Journal*, **60**, 19–25.

Hu, X. & Eberhart, R. (2002). Solving constrained nonlinear optimization problems with particle swarm optimization. *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), Orlando, USA*, **5**.

Hull, J.C. (2002). *Fundamentals of Futures and Options Markets*. Prentice Hall, 4th edn.

Inui, K. & Kijima, M. (2005). On the significance of expected shortfall as a coherent risk measure. *Journal of Banking & Finance*, **29**, 853–864.

Jorion, P. (2001). *Value at Risk: The New Benchmark for Managing Financial Risk*. McGraw-Hill, 2nd edn.

Kendall, G. & Su, Y. (2005). A particle swarm optimisation approach in the construction of optimal risky portfolios. *Proceedings of the 23rd IASTED International Multi-Conference Artificial Intelligence and Applications*.

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. *Proc. IEEE International Conf. on Neural Networks (Perth, Australia)*.

Knight, F.H. (1921). *Risk, Uncertainty, and Profit*. Hart, Schaffner & Marx; Houghton Mifflin Company, Boston, MA.

Li-Ping, Z., Huan-Jun, Y. & Shang-Xu, H. (2005). Optimal choice of parameters for particle swarm optimization. *Journal of Zhejiang University Science*, **6A**, 528–534.

Markowitz, H.M. (1952). Portfolio selection. *Journal of Finance*, **7**, 77 – 91.

Szego, G. (2002). Measures of risk. *Journal of Banking & Finance*, **26**, 1253–1272.

Tasche, D. (2002). Expected shortfall and beyond. *Journal of Banking & Finance*, **26**, 1519–1533.

Xia, Y., Liu, B., Wang, S. & Lai, K.K. (2000). A model for portfolio selection with order of expected returns. *Computers & Operations Research*, 409–422.

Zhang, W.J., Xie, X.F. & Bi, D.C. (2004). Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space. *Congress on Evolutionary Computation (CEC), Oregon, USA*, 2307–2311.