Portfolio Optimization by Heuristic Algorithms

Collether John

A thesis submitted for the degree of PhD in Computing and Electronic Systems

School of Computer Science and Electronic Engineering

University of Essex

2014

ABSTRACT

Portfolio optimization is a major activity in business. It is intensively studied by researchers. Conventional portfolio optimization research made simplifying assumptions. For example, they assumed no constraint in how many assets one holds (cardinality constraint). They also assume no minimum and maximum holding sizes (holding size constraint). Once these assumptions are relaxed, conventional methods become inapplicable. New methods are demanded. Threshold Accepting is an established algorithm in the extended portfolio optimization problem.

In this thesis, we take into consideration the cardinality and holding size constraints. We have developed five hill climbing algorithms, namely HC-S, HC-S-R, HC-C HC-C-R and Guided Local search (GLS), for the extended portfolio optimization problem.

The new hill-climbing algorithms produced are first tested in standard portfolio optimization problem. In solving standard portfolio optimization problem, we retain Markowitz's constraints that the investor has a fixed budget, and no short-selling is allowed. Results are compared (benchmarked) with the Threshold Accepting algorithm, a well-known algorithm for portfolio optimization, and quadratic programming (QP).

The new algorithms are next applied to the extended portfolio optimization problem. First, we take into consideration the cardinality constraint. Then we take on the holding sizes constraint. Results suggest that the algorithms developed in this thesis also out-performed Threshold Accepting in the extended portfolio optimization. This establishes the usefulness of the five hill climbing algorithms developed in this thesis.

Acknowledgements

I thank my God for each and everything. I thank my supervisor Prof Edward Tsang for supervision, support, guidance and encouragement. I also thank all members of staff for support and encouragement. I thank my family for their patience and support.

Table of Contents

1. INTRODUCTION

1.1 Background

The portfolio optimization problem is a problem concerning asset allocation and diversification for maximum return with minimum risk. The problem is to find the portfolio weights, i.e. how to distribute the initial wealth across the available assets, in order to meet the investor's objectives and constraints as well as possible [11, 12, and 13].

Harry Markowitz [12, 13] in 1952 came up with a parametric optimization model for the problem of asset allocation and diversification for maximum return with minimum risk, which has become the foundation for Modern Portfolio Theory (MPT) or Markowitz theory or Mean-Variance model. To apply the Markowitz model to practical problems using the standard/traditional methods like quadratic programming, strong assumptions and simplifications of the real market situations have to be made. Markowitz model considers what is termed as standard portfolio optimization. In the standard portfolio optimization problem, the constraints taken into account are budget and no-short selling. In reality however, portfolio optimization has realistic constraints to be incorporated, such as holding sizes, cardinality, transaction cost, portfolio size or additional requirements from investors and authorities. When these realistic constraints are added to portfolio optimization, the problem quickly becomes too complex to be solvable by standard optimization methods. When the assumptions and simplifications of the real market situations are relaxed and realistic constraints added, now we have an extended portfolio optimization problem. And here the Markowitz solution and the conventional methods like quadratic programming become inapplicable. Heuristic methods are usually used to deal with this extended portfolio optimization problem [5, 10, 11, 17, 19, 20, 21, 41, and 50]. The most established heuristic algorithm used in extended portfolio optimization problem being Threshold Accepting [5, 15, 17, 23, 58, 48, and 40].

The core of heuristic methods is an iterative principle that includes stochastic elements in generating new candidate solutions and in deciding whether these replace their predecessors, while still incorporating some mechanism that prefers and encourages improvements [57, 24]. In portfolio optimization, when a realistic setting is considered technically, the search space is usually discontinuous and discreet with numerous local optima.

1.2 The Objective

The objective of the research is to produce more effective and more efficient heuristic algorithms for the extended portfolio optimization problem.

In this research, heuristic methods are designed, investigated and then applied to portfolio optimization under some realistic constraints of the market. The produced algorithms are implemented in solving both the standard and the extended portfolio optimization problem.

In this thesis, the constraints taken into account in extended portfolio optimization problems are first cardinality, and then cardinality and holding sizes (maximum holding size and minimum holding size). The problem is to find the portfolio weights, i.e. how to distribute the initial wealth across the available assets, in order to meet the investor's objectives and constraints as well as possible. The significance of the research lies in efficient portfolio selection/optimization and also in efficient investment management [13].

2. LITERATURE REVIEW

2.1 Introduction

Portfolio optimization is about identifying the combination of financial assets that fits an investor's needs and requirements the best [1, 2, 4, 6, and 7]. It is about finding a combination of assets that can offer ideal trade-off between expected profit and the risk associated with it [12, 13]. This is because not all assets will show the same unexpected deviations between the expected and actual returns; while one asset increases in price another one might fall at the same time and vice versa. So in the long run, splitting one's capital and holding both or more assets will offset some of the deviations and achieve actual returns closer to the expected return [2, 3, 9, 11, and 13]. In order to find an optimal combination of financial securities, managing a financial portfolio includes determining fair prices for these securities, assessing the relationships between securities, estimating future profits of financial securities and the risk associated with it, also the analysis of investor's attitude towards risk, expected return and consumption. Usually, to model portfolio optimization and portfolio management, the assumption that markets are frictionless has to be made. Although unrealistic, this has been long considered as the only way to model the frameworks [12, 13]. But these simplifying assumptions are no longer necessary and instead more complex scenario and settings can be investigated with heuristic type of optimization and search methods [5].

2.2 Area of Study

The area of study is portfolio optimization by heuristic methods. The problem in portfolio optimization is often to reduce as much risk as possible, or to achieve the highest possible returns or both under constraints [1, 2, 4, 6, and 7]. Considering realistic situations/constraints of the market turns portfolio optimization into a too highly demanding optimization problem for standard methods. The focused area of the study is portfolio optimization by heuristic techniques. The purpose of the study is to tackle realistic portfolio optimization by making the Markowitz model more applicable in real situations and constraints of the market [11, 15].

2.3 Modern Portfolio Theory (MPT) Or Markowitz Theory or Markowitz Model

Harry Markowitz in [12,13] in 1952 came up with a parametric optimization model for the problem of asset allocation and diversification for maximum return with minimum risk, which has become the foundation for Modern Portfolio theory (MPT) or Markowitz theory or Mean-Variance theory. Others came up with ways to implement the model like Capital Asset Pricing Model (CAPM) in [1] where the model was developed that shows that rational investors with homogenous expectations will hold a portfolio that somehow emulates the market with a safe or risk free asset. Another way to implement the model is Arbitrage Pricing theory (APT) in [14].

In the mean variance framework, [12, 13], the selection problem can be split into two steps. From a universe of feasible portfolios, the majority can be classified as inefficient and should not be held by any investor for whom the usual assumptions of risk aversion apply. Risk plays an important role in modern finance, including risk management, capital asset pricing and portfolio optimization. Which of the remaining efficient portfolios ought to be picked, however, depends on the investor's preferences.

Markowitz's standard portfolio optimization model is a mathematical framework for describing and assessing return and risk of a portfolio of assets, using returns, volatilities and correlations. Markowitz introduced what is known as the mean-variance principle, where future returns are regarded as random numbers and expected value (mean) of the returns E(r) and their variance (whose square root is called standard deviation/ risk) capture all the information about the expected outcome and the likelihood and range of deviations from it [12,13].

2.3.1 Objective Function

In the standard Markowitz model below, the goal is to maximize the expected return, R, while diminishing incurred risk, σ, (measured as standard deviation/variance) [5].

Given return ($R_{\mathbf{p}}$) of a portfolio and variance ($\sigma^2_{\mathbf{p}}$) of portfolio, the equation to maximize is

Max $(\lambda.E(R_{\mathbf{p}}) - (1-\lambda).\sigma^2_{\mathbf{p}})$ (1)

Subject to

- Expected return:

  $E(R_{\mathbf{p}}) = \sum_i w_i \, \mathrm{E}(R_i)$ (2)

- Portfolio return variance:

  $\sigma^2_{\mathbf{p}} = \sum_i \sum_j w_i \, w_j \sigma_i \sigma_j \, \rho_{ij}$ (3)

  $\rho_{ij} = 1$ for $i=j$

- $\sum_i w_i = 1$ (4)
- $0 \leq w_i \leq 1$ (5)

Where the expected return of each asset is $E(R_i)$, each asset variance is σ_i, and each asset weight is w_i.

From the equation (1) the trade-off between return ($R_{\mathbf{p}}$) and risk ($\sigma_{\mathbf{p}}$) of portfolio is reflected. The efficient line/frontier is then identified by solving the above problem for different values of $\lambda \in [0, 1]$: If $\lambda=1$ the model will search for the portfolio with highest possible return regardless of the variance. With $\lambda=0$, the minimum variance portfolio (MVP) will be identified. Higher values of λ put more emphasis on portfolio's expected return and less on its risk. [5]. Equation (4) and (5) are the constraints on the weights that they must not exceed certain bounds.

2.3.2 Asset Return

Asset return is the payoff of financial security after investment in that security. From the Markowitz model, two moments, mean and variance, describe the asset returns and utility of a rational risk averse investor. The basis for decisions on investment is the trade-off between the higher return and higher variance. Returns are assumed be normally distributed. This assumption of symmetry does not capture extreme events and may apply mostly for small set of stocks [4, 6, and 7]. The equation (2) is the mathematical expression for the overall expected return from the assets in a portfolio.

$$E(R_{\mathbf{p}}) = \sum_i w_i \, \mathrm{E}(R_i) \tag{2}$$

2.3.3 Risk

Risk explains a situation where the exact outcome is not known [7]. The value or measure of risk shows the magnitude of deviations from the expected value/outcome. This results in positive or negative news. Risk can be measured in several ways, the most popular being volatility which is the square root of variance [4]. Semi-variance measures only the negative deviations from the expected value. Another measure of risk is VAR (value at risk) which indicate the maximum loss within a given period of time with a given probability. VAR is due to findings and assumptions that investors put additional weight on losses. That is decisions are driven by loss aversion [5]. The equation (3) is the mathematical expression for the portfolio variance whose square root is called risk.

$$\sigma^2_{\mathbf{p}}=\sum_i \sum_j w_i \, w_j \sigma_i \sigma_j \, \rho_{ij} \tag{3}$$

2.3.4 Diversification

One of the outcomes of mean-variance theory is that investors will want to hold as many different assets as possible. This is if there is no constraint on short selling and no transaction costs [5]. Markowitz showed that riskiness of a portfolio can be reduced by diversification [8, 9]. If the correlation between assets is lower, it means the diversification of assets is larger. The less the stocks are correlated the more risk can be eliminated and so can result in better investor's utility. So portfolio investment is not only advantageous over a single stock investment but also the more different stocks the better. In reality, however, there are transaction costs and administration costs. It is more work to monitor a portfolio of a large number of different assets. In practice, deciding on the right weight for an asset is done together with deciding whether the asset should be included at all, as most of the risk diversification can be realized with a well-chosen small set of assets [5].

2.3.5 The efficient frontier

For every level of return, there is one portfolio that has the lowest possible risk and for every level of risk there is a portfolio that offers the highest return. This combination when plotted on a graph of the curve/line is the efficient frontier. The curve is usually a convex one but may change depending on the constraints imposed on the investor. The portfolios of this combination of return, risk values, plotted on the efficient frontier make up the set of efficient portfolios [8, 9].

2.3.6 Shortcomings of Markowitz model

The Markowitz model is one period or static (independent of the actual length of time) and he had to make unrealistic assumptions, like there are no realistic constraints like cardinality, maximum holding size, minimum holding size, transaction costs, regulations and securities are perfectly divisible. To apply the Markowitz model to practical problems using the standard/traditional methods like quadratic programming, strong assumptions and simplifications of the real market situations, have to be made. In reality, however, portfolio optimization has realistic constraints to be incorporated as mentioned before. When these realistic constraints are added to portfolio optimization, the problem quickly becomes too complex to be solvable by standard optimization methods. Here the Markowitz solution becomes inapplicable [5, 11, 17, 19, 20, and 21].

We have stated that optimization has to consider the above realistic constraints to be realistic. This thesis will handle cardinality, maximum holding size, and minimum holding size incorporating them in the Markowitz model.

In minimum holding size, assets either are above a certain lower bound, or they are not part of the portfolio at all. This is to prevent the assets with small weights from being included in the portfolio [18]. The reason behind this being to avoid the cost of administrating very small portions of assets and transaction costs [5].

Maximum holding size constraint is when there is a limit to the maximum proportion allowed to be held for each asset in a portfolio. The purpose of this, which can also be there because it is imposed by law, is to avoid excessive exposure to a specific asset in a portfolio [34].

The cardinality constraints limit a portfolio to have a specified integer number of assets. Cardinality constraints are there for monitoring or management reasons and in order to reduce management and transaction costs [35].

2.4 Approaches to Portfolio Selection Problem

Some of the approaches to modelling the portfolio selection problem are the Mean-Variance approach and Scenario Generation approach. These will be elaborated below.

2.4.1 The Mean-Variance Approach

Optimization by mean-variance by Harry Markowitz [12, 13] is the most popular approach to the portfolio selection problem. In this structure, the investor faces a trade-off between the gain from his portfolio, described as the expected return, and the risk, measured by the variance of the portfolio returns. These first two moments, mean and variance, of the portfolio future return are taken to be sufficient to define a complete ordering of the investors' utility functions. This strong result is due to the simplistic hypothesis that the investors' utility functions are quadratic and the distribution of returns is normal.

The efficient portfolios of the mean-variance are defined as having the highest expected return for a given variance and the minimum variance for a given expected return [1, 12, and 13]. Efficient algorithms exist to compute the mean-variance portfolios.

2.4.2 Scenario Generation Approach

Another approach to the above optimization setting is the scenario analysis where uncertainty about future returns is modelled through a set of possible realizations called scenarios. A model, experts' opinions, or past returns are used to generate scenarios of future outcomes.

A straightforward approach is to use empirical distributions computed from past returns as equally likely scenarios. Observations of returns over overlapping periods of a certain length are considered as the possible outcomes, or scenarios, of the future returns and a probability S is assigned to each of them [17].

2.5 Heuristic Portfolio Optimization Techniques

2.5.1 Portfolio Optimization Problem

Optimization problem is about finding the values for decision variables that meet the objectives the best without violating the constraints. Optimization problems might have multiple solutions depending on the objective function. Some of these solutions might be local optima. A solution is global optimum if it yields the best overall value for the objective function. If the solution space is too complex, it is often difficult to determine whether an identified solution is a local or global optimum. Finding an efficient portfolio in the Markowitz model, equation (1) in section 2.3.1 is an optimization problem. The objective is to maximize equation (1) under the constraints that the asset weights must not exceed certain bounds (equations (4) and (5) in section 2.3.1) [5].

Although the Markowitz model is a well defined optimization problem, there exists no general solution for the optimization problem, because of the non-negativity constraint on the asset weight. Though the Markowitz model cannot be solved analytically, numerical methods exists by which the model can be solved for a given set of parameters [5, 11]. The capacity of these traditional/standard methods, rely on strong assumptions and simplifications which do not reflect the real market situations [1, 14]. Examples of real market situations are the existence of regulations and/or trading restrictions, transaction costs and other fees. For reliable results that reflect the effects of the real market situations, alternative optimization techniques that are capable of dealing with these real market situations have to be employed. These are heuristic optimization techniques [5].

2.5.2 Characteristics of Heuristic Optimization Techniques

The core of heuristic methods is an iterative principle that includes stochastic elements in generating new candidate solutions and/or in deciding whether these replace their predecessors – while still incorporating some mechanism that prefers and encourages improvements [11, 15, 59, and 57]. They seek to converge to the optimum in the course of the iterated search. They are flexible and not so restricted to certain forms of constraints. Heuristic techniques solve optimization problem by repeatedly generating new solutions and testing them. Therefore heuristic techniques address problems with a well defined objective function and model [11].

The following is an explanation on heuristic techniques.

2.5.2.1 Initial Solution

The choice of an initial solution for heuristics for the portfolio selection problem is randomly generated, or a solution constructed by a means of simple heuristic procedure [11, 15, 59, 57]. The requirement to this starting solution is for it to conform to the constraints to ensure feasibility of the initial portfolio. A separate mechanism can be used to ensure the feasibility [40]. In this thesis, the initial solution was a set of randomly generated integers to conform to the constraint of no short selling, that is no negative values, and they were all scaled to 1 (100%) to conform to the constraint of budget (100% of capital is to be invested in the portfolio).

2.5.2.2 Iterative Improvement

Improving from the initial solution to the required global optimal solution is achieved iteratively [11, 15, and 40]. Iterative improvement can be considered as the simplest neighbourhood search, as it performs a path in the search space by moving from one solution to a neighbouring one according to the already set neighbourhood tuning parameters or a certain mechanism. This neighbourhood search can be named best improvement, if the neighbour chosen is the best among the feasible neighbours, or just first improvement, if the chosen neighbour is the first solution found during the neighbourhood search that is better than the current one [34, 11]. A more complex strategy can also be used for iterative improvement, example, [39] a greedy search is used to refine solutions found by an ant colony algorithm.

2.5.2.3 Stopping Criteria

The stopping criterion of the heuristic algorithms is usually a fixed number of steps or if the quality of the solution does not improve after a given or specified number of iteration or both [15, 40].

2.5.2.4 Computation resources and stochastic solutions

The local search methods usually get candidate new solution by randomly trying out one candidate solution after another, using the objective function. It ignores the information that the derivatives of the objective function provide. This makes them less efficient than the gradient based methods as they require more computing time. But in recent years this has become less of a concern due higher speed of computers. Computational resources can also be measured using the number of objective function evaluation [40].

Also running the same technique twice normally results in different solutions. A number of runs are required to run a program, which is from a different starting point, for a convergence of a solution or to reach approximate global optimum [23, 40].

2.5.3 Some Portfolio Heuristic Optimization Techniques

2.5.3.1 Simulated Annealing

Simulated annealing [45] is a type of local search algorithm that accepts all new points that are superior to the current solution according to the objective function, but also, with a certain probability, accept inferior points. By accepting inferior points, the algorithm avoids being trapped in local minima, and is able to explore more widely for better solutions. The probability of accepting an inferior point decreases over time, following a cooling schedule on the "temperature". When the temperature falls to 0, SA behaves exactly like hill climbing. SA has been applied for portfolio selection [20, 21], and with constraints and trading restrictions in [19].

Definition: $f(x)$ is the objective function value due to solution x.

Pseudo code for Simulated Annealing [45]

```
Generate initial solution $xc$,
 Initialize maximum number of rounds/steps, $R$max and
Temperature, $Temp$.
for $r = 1$: $R$max do
while stopping criteria not met do
        Compute $x$n   (neighbour to current solution $x$c)
        Compute Difference Diff=$f(x\text{n}) - f(x\text{c})$ and generate $u$ (uniform random variable)
 if (Diff< 0) or ($e$−Diff/$Temp > u$) then
        $x\text{c} = x\text{n}$
 end while
        Reduce $Temp$
end for
```

2.5.3.2 Threshold Accepting

Threshold Accepting [22] can be seen as a variation of simulated annealing, except that there is no introduction of temperature. Instead of accepting inferior new points with a certain probability, it accepts only the points that fall below a fixed threshold. TA was originally proposed by [22] as a deterministic and faster variant of the original Simulated Annealing algorithm.

As Threshold Accepting avoids the probabilistic acceptance calculations of simulated annealing, it may locate an optimal value faster than the actual simulated annealing technique.

In Threshold Accepting algorithm, the best solution obtained depends on some parameters such as the initial threshold value, the threshold decreasing rate and the number of permutations. The initial threshold and threshold decreasing rate are fixed such that a number of iterations can be carried before the algorithm stops.

In [15, 17, 23, 58, 40, and 48], Threshold Accepting is applied for constrained portfolio optimization. Different utility functions can be optimized because of the flexibility of the TA algorithm implemented for portfolio selection. These include transaction costs, multiple-currency portfolios, cash-flow control, depreciation and losses and income taxes.

Definition: $f(x)$ is the objective function value due to solution x.

Pseudo code for Threshold Accepting [48]

```
Initialize number-of-Rounds, $n$Rounds and number-of-steps, $n$Steps
Compute threshold sequence $\tau r$
Randomly generate current solution $x$c in the search space X
for $r$ = 1: $n$Rounds do
for $i$ = 1: $n$Steps do
Generate $x$n  neighbour to ($x$c)
compute Difference D = $f(x\text{n}) - f(x\text{c})$
 if D < $\tau r$ then
        $x$c = $x$n
end for
end for
        $x$soln = $x$c
```

In this thesis, Threshold Accepting is used as a benchmark algorithm to the proposed hill climbing algorithms in solving the standard Markowitz model.

2.5.3.3 Evolutionary algorithms (EA)

These are population based heuristics from the inspiring Darwin's theory of natural evolution and selection. At each iteration, these search techniques change and manipulate a set of solutions combining the best solutions of the current set to generate the solutions of the next set, while saving the best solution found during all iterations [34].

There has been a trend of hybrid heuristics of evolutionary algorithm and local search to get the benefits of both, so often EA-based heuristics are enhanced by hybridizing EAs with local search strategies and/or advanced constructive procedures, for example in [50]. The name memetic algorithm (MA) is used to describe strategies where local search runs are executed to improve the quality of the solutions constructed by the EA [39]. Examples are [38] the local search procedure that is used for enhancing the performance of standard differential evolution (DE) algorithm. In [10], the paper evaluates the hybridization of a multi-objective evolutionary algorithm and a quadratic programming (QP) local search on multiple instances of the constrained and unconstrained portfolio selection problem, using a problem specific representation. The memetic algorithm proves to be a two-edged approach, on one hand, it improves the convergence rate for some problem instances. While on other hand of problem instances, the local search causes a neutral search space and eventually premature convergence. The paper investigates this behaviour, offers some explanation and also outlines a possible remedy.

Evolutionary methods also include all the various forms of genetic algorithms and genetic programming. One successful evolutionary method is Differential evolution [51]. The method is easy to implement and has few parameters to tune when applied [52]. Also the parameters are more or less standard in that the values produce good result to different set of problems [53].

Among the Evolutionary methods that have been successfully used for portfolio optimization are described in [16, 29, and 41]. Also, in [32], evolutionary strategy was applied to tackle portfolio optimization. The strategy employs *k*-means cluster analysis to eliminate the cardinality constraint and thereby simplify the mathematical model and the evolutionary optimization process. The strategy also employs refined weight standardization algorithms to tackle the bounding constraints and class constraints. Authors in [36] apply Multi-objective evolutionary algorithm (MOEA) on the constrained portfolio selection problem based on the Markowitz mean-variance model, and suggest a new hybrid encoding of the portfolio selection that proves to be more efficient than a standard encoding. They showed that the suggested hybrid encoding is able to solve the portfolio optimization problem more efficiently than the standard encoding based on a single real-valued vector of decision variables.

Basic Structure for EA [5]

```
        Generate $P$ random solutions $x_1 \ldots x_p$
repeat
        for each parent individual $i=1 \ldots P$
                Generate offspring $x`_i$ by randomly modifying the "parent" $x_i$
                Evaluate new solution $x`_i$
        end
        Rank parents and offspring
        Select the best $P$ of these solutions for new parent population
until halting criteria met
```

2.5.3.4 Ant Colony Optimization

The unique behaviour of ants inspired this population-based heuristic known as Ant colony optimization (ACO). Solutions are generated component by component, following a probabilistic procedure that biases the choice of the next solution component on the basis of the quality of the previous constructed solutions. Successful application of ACO in a portfolio selection problem modelled with the cardinality constraint is in [43, 34].

2.5.3.5 Particle Swarm

Particle swarm optimization approach is the nature-inspired search algorithm that is useful when solving continuous optimization problems. It is for both discreet and continuous problems.

Its application to the portfolio selection problem has been demonstrated by [44], in which results show that it is only when dealing with problem instances that demand portfolios with a low risk of investment, that the particle swarm optimization model gives better solutions than genetic algorithms, Tabu search and simulated annealing [34].

2.5.3.6 Spin Glass

Spin glass optimization is a distributed technique inspired by the interactions in spin glasses in nature. Spin glasses are the lattices of spins where each spin is only a part of the entire solution [37].

This technique was applied to the Markowitz standard portfolio model. Although the algorithm is computationally intensive, it was found to be superior to SA (Simulated Annealing) regarding accuracy. However, experiments showed that the use of local search significantly increased the speed of the technique at the cost of decreased accuracy. The algorithm aimed at achieving global optimization by parallel local search [37].

2.6 Some of the Realistic Constrained Portfolio Optimization Problems

Here the optimization problem can be of Single objective, Multi-objective or Dynamic [49]. A major difference between single-objective optimization and multi-objective optimization is that in the single-objective optimization we obtain a single solution and in the multi-objective optimization we have a number of non-dominated solutions (Pareto Front) [54]. For example, Single objective portfolio optimization is when you intend to either maximize return or minimize risk. In multi-objective portfolio optimization, risk and return are simultaneously considered.

2.7 Summary of Some of the Related Heuristic Techniques

The heuristics for the portfolio selection problem are mostly either trajectory based strategies, such as simulated annealing [45], Threshold accepting [48], and Tabu search [46], and population-based heuristics, such as evolutionary algorithms where there are methods like genetic algorithms and differential evolution algorithm, particle swarm and ant colony optimization. So the development of heuristics has mainly been in using two principles, as local search and as population-based search. The population-based search consists of maintaining a pool of good solutions and combining them so as to produce better solutions. Examples are the genetic algorithms. In local search methods, an intensive exploration of the solution space is performed by moving at each step from the current solution to another feasible solution in its neighbourhood as explained below. Some of the famous local search methods are simulated annealing, originally proposed by [45], and Tabu search [46]. Each of these heuristics have their own principles for implementation, [47] attempt to give guidelines for adaptation of all local search and population based search methods.

2.7.1 Local Search

Local search is the basis of many heuristic search methods for solving computationally hard combinatorial problems. Local search starts a search with a randomly or heuristically generated candidate solution of a given problem instance. It then iteratively improves this solution to a neighbour solution usually by means of minor modification according to the objective function. Neighbour solutions are a set of candidate solutions. When all neighbouring solutions are no better than current candidate solution, the local search stops. This means local search can get stuck in a local optimum, although it usually finds good solutions very fast. This situation, where no direct improvement is possible can be handled in many ways, which has led to many variations of local search methods [55, 56, and 58]. Randomization in generating new, neighbouring, solutions is used by many stochastic local search methods to overcome stagnation with unsatisfactory solutions [55, 56].

Below is the general pseudo code for Local Search [58]. f(x) is the objective function value due to solution *x*.

```
Procedure Local-Search ()
        Initialise number-of-steps, nSteps
        Randomly generate current solution xc from the search space X
for i = 1 : nSteps do
        Generate xn which is neighbour to (xc)
compute Diff = f (xn) − f (xc)                if Diff < 0 then
        xc = xn
end for
        xsol = xc
```

2.7.2 Hill Climbing

One technique that belongs to the class of local search methods is Hill Climbing. It is an algorithm that requires two functions, which is evaluation function or objective function and adjacency function or neighbourhood function. From a random focal point in the search space Hill-Climbing uses the adjacency function to get the next solution which is to be evaluated by the evaluation function to determine if it is a better solution [57, 24]. Another strategy used to overcome local minima that is used by many local search methods is the acceptance of a candidate solution that does not improve objective function. For maximization problem, a solution that does not maximize the objective function is also accepted as a new candidate solution, likewise for the minimization problem, a solution that does not minimize the objective function may be accepted as a new candidate solution [57].

2.7.3 Guided Local Search

Guided Local Search (GLS) [55, 57] is a meta-heuristic search method that uses penalties to help local search algorithms escape local minima or plateaus. Guided Local search sits on top of a local search algorithm hence called a meta- heuristic. It works by building up penalties during a search [55, 57]. The solution features are defined to distinguish between problems with different characteristics. For a given problem, a set of features for candidate solution need to be defined. Some of these features, the poor characteristics, are selected and penalized when a local search is trapped in local optima. Each feature, *i*, is associated with a penalty *pi*. The objective function is augmented by the accumulated penalties. The local search searches the solution space using the augmented objective function [55, 57].

Below is the pseudo code of GLS [55, 57] applied in finding the optimum portfolio of n number of assets.

In the pseudo code below, *p* is the problem, *g* is the objective function, *h* is the augmented objective function, λ is a parameter, *Ii* is the indicator function of feature *i*, *ci* is the cost of feature *i*, *M* is the number of features, *pi* is the penalty of feature *i*,.

```
Procedure Guided Local Search (p, g, λ , [I1, . . . , IM], [c1, . . . ,cM], M)

begin
        k=0;
        s0 is randomly generate initial solution (p);
        % set all penalties to 0 %
        for i=1:M do
                pi =0;
                % define the augmented objective function %
                h=g+ λ*∑pi *Ii;
        while StoppingCriterion do
        begin
                sk+1 =Hill-climbing method(sk,h);
                % compute the utility of features %
        for i=1: M do
                utili =Ii(sk+1) *ci/(1+pi);
```

```
            % penalize features with maximum utility %
        for each $i$ such that $utili$ is maximum do
            $pi = pi+1$;
            $k=k+1$;
        end
            $s*$ is best solution found with respect to objective function $g$;
        return $s*$;
end
```

In this thesis the method Guided local search (GLS) is applied as a meta-heuristic in one of the proposed algorithms (HC-C-R).

One of the major advantages of heuristic methods over the traditional deterministic methods is that the randomness allows the escaping of the local optima, which is an important issue in many financial optimization problems [11]. That is, their solution is global optimal or at least near global optimal.

Other advantages of optimization heuristics include the fact that constraints are easily integrated and the algorithm works even if the objective function is changed [11].

The disadvantages include requiring extensive parameter tuning, and compared to other standard techniques like integer solver 0r QP (Quadratic Programming) solver, more work has to be put into designing the heuristics algorithms [18].

2.8 Constrained Portfolio Optimization

Practical optimization problems, like portfolio optimization, are expected to include constraints. There are equality and inequality constraints. There are methods to handle constrained optimization problem. The algorithm must seek to accomplish two principal outcomes, to satisfy all constraints and for it to be optimal, with feasibility being more important than optimality. So the optimal solution must be feasible. There are indirect methods and direct methods. Indirect methods include exterior penalty function (EPF) [61], and the Augmented Lagrange multiplier method [61]. Direct methods include expansion of functions, sequential linear programming (SLP), and sequential quadratic problem [61].

In tackling the Markowitz model under some of the realistic constraints of the market, like cardinality and holding sizes constraints, the goal is to maximize the expected return while diminishing incurred risk, measured as standard deviation/ variance, [5] under the realistic constraints; cardinality and holding sizes constraints. In the maximization problem below, the cardinality and holding sizes constraints are defined.

Given return ($R_{\mathbf{p}}$) of a portfolio and variance ($\sigma^2_{\mathbf{p}}$) of portfolio, the equation to maximize is

Maximize $(\lambda.E(R_{\mathbf{p}}) - (1-\lambda).\sigma^2_{\mathbf{p}})$ (1)

Subject to:

- Expected return:

$$E(R_{\mathbf{p}}) = \sum_{i=1}^{K} w_i E(R_i) \tag{2}$$

- Portfolio return variance:

$$\sigma^2_{\mathbf{p}} = \sum_i \sum_j w_i \, w_j \sigma_i \sigma_j \, \rho_{ij} \qquad for\ all\ i,j = 1,2,\dots,K \tag{3}$$

$$\rho_{ij} = 1 \qquad \text{for } i{=}j$$

- Basic Constraints:

$$\sum_{i=1}^{K} w_i = 1 \tag{4}$$

$$0 \le w_i \le 1 \tag{5}$$

- Cardinality Constraints:

$$\sum_{i=1}^{N} Z_i \leq K \text{ where } Z = \begin{cases} 1 & if\ w_i > 0 \\ 0, & if\ w_i = 0 \end{cases} \tag{6}$$

- Holding sizes Constraints:

  $\varepsilon_i \leq w_i \leq \delta_i,\ i = 1,2,\dots,K$

  $$0 \leq \varepsilon_i < \delta_i \leq 1 \tag{7}$$

Where the expected return of each asset is $E(R_i)$, each asset's variance is σ_i, and each asset's weight is w_i.

From the equation (1) the trade-off between return (R_p) and variance (σ^2_p) of portfolio is reflected. Standard deviation (Risk) is obtained as the square root of Variance. The efficient line/frontier is then identified by solving the above equation (1) for different values of $\lambda \in [0, 1]$: If λ=1 the model will search for the portfolio with highest possible return regardless of the variance. With λ=0, the minimum variance portfolio (MVP) will be identified. Higher values of λ put more emphasis on portfolio's expected return and less on its risk. [5].

K in equation (6) is the Cardinality constraint in which the investor decides to invest in K or less number of assets, out of N assets.

ε_i and δ_i in equation (7) are minimum and maximum holding sizes respectively of weight of assets.

2.8.1 Budget and Return Constraints

The most important constraints are budget and return constraints since they characterize the main part of the portfolio problem [34]. The return constraint is when the investor requires a certain level of profit from his investment with minimum risk [1]. The budget constraint is when the investor has to invest all the money/capital in the portfolio. However return constraints can only be satisfied for a historical portfolio [5]. The unconstrained Markowitz model includes these constraints which are also used to define the solution's feasibility. The equation (4) is the mathematical expression for the budget constraint.

$$\sum_{i=1}^{K} w_i = 1 \tag{4}$$

2.8.2 No Short Selling

With the constraint that short selling is not allowed, asset weights must be positive numbers. This is part of the original Markowitz model. But this is sometimes not realistic, as short-selling happens and it is known to be a common practice with the investors. The relaxation of the constraint in the model, to analytically allow short selling, was first introduced in [42]. The equation (5) above is the mathematical expression for the no short selling constraint.

$$0 \leq w_i \leq 1 \tag{5}$$

2.8.3 Cardinality Constraints

The cardinality constraints limit a portfolio to have a specified integer number of assets [35]. Cardinality constraints are there for monitoring and management reasons and in order to reduce management and transaction costs. The first to tackle cardinality constraint in portfolio optimization problem by heuristics were the authors of [35]. The equation (6) is the mathematical expression for the cardinality constraint.

$$\sum_{i=1}^{N} Z_i \leq K \text{ where } Z = \begin{cases} 1 & if\, w_i > 0 \\ 0, & if\, w_i = 0 \end{cases} \tag{6}$$

2.8.4 Minimum Holding Size or Buy-In Threshold

Here assets either are above a certain lower bound, or they are not part of the portfolio at all. This is to prevent the assets with small weights from being included in the portfolio [18]. The reason behind it being to avoid the cost of administrating very small portions of assets and transaction cost [5]. The equation (7) above is the mathematical expression for the minimum and maximum holding size constraints. If only minimum holding size is considered, the equation for the constraint reads as follows.

$$\begin{aligned} w_i \geq \varepsilon_i, \qquad i = 1,2,\dots,K \\ 0 \leq \varepsilon_i < 1 \end{aligned} \tag{7a}$$

2.8.5 Maximum Holding Size or Ceiling Constraint

This is when there is a limit to the maximum proportion allowed to be held for each asset in a portfolio. The purpose of this, which can also be there because it is imposed by law, is to avoid excessive exposure to a specific asset in a portfolio [34]. The equation (7) above is the mathematical expression for the minimum and maximum holding sizes constraints. If only maximum holding size is considered, the equation (7) is reduced to following equation.

$$\begin{aligned} w_i \leq \delta_i, \qquad i = 1,2,\dots,K \\ 0 < \delta_i \leq 1 \end{aligned} \tag{7b}$$

2.8.6 Some Other Constraints

2.8.6.1 Transaction Costs

Transaction costs refer to the amount to be paid in order to buy assets. The assets have fixed transaction costs, proportional transaction cost or both [5]. It has been shown that if transaction costs are ignored, this results in an inefficient portfolio [63]. It also leads to a non-diversified portfolio since the transaction costs are not included in the original Markowitz model [64]. Including transaction costs makes the problem non-convex, so cannot be solved by convex optimization methods. Instead other techniques like relaxation methods and heuristics have to be applied [65]. In [5], it is shown that the higher the transaction cost the lower the performance of a portfolio, and also the lower the cardinality (total integer number of assets) of the portfolio, especially for small investors.

2.8.6.2 Class Constraints

In class constraints, the assets are categorized in classes or sets with common characteristics so that the investors are able to limit the proportion of each class in the portfolio. This is for easy class management and diversification [34]. Optimization is then based on selected the best representative of each class.

2.9 Handling Constraints

Many approaches exist for handling constraints in heuristic optimization, e.g. see [56, 60]. For all the iterative techniques above, the simplest approach used to handle constraints is to "throw away" infeasible new solutions. If a neighbour violates a constraint, a new neighbour is picked. The approach is efficient for a model with few constraints.

Information of the constraints can be directly used to create a new solution from a previous solution. This approach can be applied in budget constraint where a new solution is created by increasing some weights in the portfolio and decreasing others such that the weight change is zero.

Another approach is to introduce a mechanism that corrects violations, effectively repairing a solution. One mechanism for budget constraint could be scaling the weights such that they all sum to one.

Depending on the problem, one of the many approaches is used. Also different approaches are often used to tackle different constraints in the same problem [40], as is the case in this thesis.

2.10 Discussion on Other Works in Portfolio Optimization

Here we look at some of the other published work that requires more analysis. Some of the areas of analysis, or research are on multi-objective optimization, incorporating of constraints and the dealing with real world return data used and their discrepancies.

The portfolio optimization problem may be formulated in many ways depending on the choice of the objective functions, the description of the decision variables, and the constraints underlying the specific situation. The expected return and variance of return are traditionally the main objectives considered as in the Markowitz portfolio model [12, 13]. However, there are additional objective functions which can be incorporated: dividend, number of securities in a portfolio, amount of short selling, excess return over a benchmark random variable, liquidity [26]. While in the bank portfolio management, the additional objectives such as expected default rate, the prime rate, processing cost, can be incorporated [27]. As a real example, the multi-objective portfolio selection problem can include the following objectives to be minimized: deviations from asset allocation percentages, number of securities in portfolio, turnover (i.e. minimize costs of adjustment), maximum investment proportion weight, amount of short selling. The following objectives are to be maximized: portfolio return, dividend, growth in sales, liquidity, excess portfolio return over that of a benchmark [28].

In dealing with the investors' wish to hold a small number of diversified assets in the portfolio, cardinality constraint, the authors of [71] proposed a hybrid local search algorithm which combines the principles of Simulated Annealing and Evolutionary strategies. The approach was efficient in tackling the cardinality constraint in portfolio selection [71].

In showing the benefit of using intraday return instead of daily return, variance was used as a measure of risk [30, 31].

In Harry Markowitz's work [12, 13] returns of financial assets are represented by their mean, while risk is represented by variance. Using the first two moments only is not sufficient since the returns do not follow a Gaussian normal distribution [69]. Investors with non-increasing absolute risk aversion do like positive skewness as it indicates that extreme deviations from the mean tend to be on the plus side. Such investors dislike high kurtosis which indicates that extreme events have a high probability on either side [69]. Also, stylized market facts show that higher order moments do matter as empirical data is skewed and more importantly, exhibit excess kurtosis and fat tails [69].

An extension of the Markowitz model by incorporating the optimization of higher–order moments is considered in [69, 67]. The inclusion of higher order moments has been proposed as one possible augmentation to the model in order to make it more applicable to real situation [66, 69]. In [68] shows that the applicability of the model can be broadened by relaxing one of its major assumptions, that is the rate of returns is normal.

As one of the cases for portfolio constraints handling, as a pre-requirement, an investor may wish some specific assets to be included in the portfolio, in proportion that is fixed or to be determined. Assets which must be in the portfolio can be accommodated in constrained formulation [35].

[62] Investigated portfolio optimization problem with real-life market constraints of transaction costs and integer constraints at the same time. The two objectives are of very difference scales. Their approach was promising in tackling the disparately scaled objectives [62].

The use of heuristic methods in estimation and modelling econometric appears to be limited compared to other fields of sciences where they have become more standard [70]. The authors in

[70] give an introduction to the heuristic optimization methods and also present some examples for which heuristic optimization techniques work efficiently.

This thesis combines both return and risk in a single objective, thus portfolio optimization is dealt with as a single objective optimization problem. We also observe cardinality, maximum holding sizes and minimum holding sizes constraints.

3. DESIGN OF THE ALGORITHMS

The following are the algorithms that are proposed and their notations:

HC-S: Hill Climbing-Simple

HC-C: Hill Climbing-Complete

HC-S-R: HC-S with reducing ThP

HC-C-R: HC-C with reducing ThP

GLS: Guided Local search [55, 57] sitting on top of HC-C-R

HC-S is a basic hill-climbing algorithm. In each step, it attempts to improve the solution by changing the relative weight of a single asset. In contrast, HC-C explores the possibility of changing the relative weight of every asset in each step, hence the name "complete".

ThP stands for Threshold Percentage. It refers to the size of a step in the proposed hill climbing methods. In HC-S and HC-C, ThP is fixed (to 0.5%). HC-S-R and HC-C-R improve HC-S and HC-C, respectively. R denotes the case where ThP is reduced after a set of iterations. In HC-S-R and HC-C-R, ThP is reduced from 5% to 0.5% during the course of hill-climbing. That has the effect of refining the step-size in the search.

3.1 Representation of Solution

In our approach, a solution is represented by a vector of numbers [y_i, …, y_n]. The element in position *i* represents the relative weight of the capital invested in stock *i*. The vector of numbers [y_i, …, y_n] are normalized to make sure that the weights in all the assets add up to 1.

The percentage/weight to be invested in stock *i* is x_i, where:

$x_i = y_i / \sum_{i=1}^{n} y_i$

One advantage of using this representation is that the vector, y, may take any number without violation of budget constraint that the weights add up to 100%.

3.2 Objective Function

Below is the extended Markowitz model tackled in this thesis (the standard Markowitz model plus the cardinality and holding sizes constraints) [Section 2.9]. Here the goal is to maximize the expected return while diminishing incurred risk (measured as standard deviation/ variance) [5].

Given return (R_p) of a portfolio and variance (σ^2_p) of portfolio, the equation to maximize is

Maximize $(\lambda . E(R_p) - (1-\lambda) . \sigma^2_p)$ (1)

Subject to:

- Expected return:

  $E(R_p) = \sum_{i=1}^{K} w_i E(R_i)$ (2)

- Portfolio return variance:

  $\sigma^2_p = \sum_i \sum_j w_i w_j \sigma_i \sigma_j \rho_{ij} \quad for\ all\ i, j = 1,2, \dots, K$ (3)

  $\rho_{ij} = 1$ for $i=j$

- Basic Constraints:

$\sum_{i=1}^{K} w_i = 1$ (4)

$$0 \leq w_i \leq 1 \tag{5}$$

- Cardinality Constraints:

$$\sum_{i=1}^{N} Z_i \leq K \text{ where } Z = \begin{cases} 1 & if\ w_i > 0 \\ 0, & if\ w_i = 0 \end{cases} \tag{6}$$

- Holding sizes Constraints:

$$\varepsilon_i \leq w_i \leq \delta_i,\ i = 1,2,\dots,K$$

$$0 \leq \varepsilon_i < \delta_i \leq 1 \tag{7}$$

Where the expected return of each asset is $E(R_i)$, each asset's variance is σ_i, and each asset's weight is w_i.

From the equation (1) the trade-off between return (R_p) and variance (σ^2_p) of portfolio is reflected. Standard deviation (Risk) is obtained as the square root of Variance. The efficient line/frontier is then identified by solving the above equation (1) for different values of $\lambda \in [0, 1]$: If $\lambda=1$ the model will search for the portfolio with highest possible return regardless of the variance. With $\lambda=0$, the minimum variance portfolio (MVP) will be identified. Higher values of λ put more emphasis on portfolio's expected return and less on its risk. [5].

K in equation (6) is the Cardinality constraint in which the investor decides to invest in K or less assets, out of N assets.

ε_i and δ_i in equation (7) are minimum and maximum holding sizes respectively of weight of assets.

3.3 Design of HC-S

In this contribution, the hill-climbing algorithm is denoted as HC-S. Here HC stands for Hill Climbing, S stands for Simple search of neighbourhood, according to the neighbourhood functions defined below.

3.3.1 Neighbourhood Function for the Hill Climbing Algorithm HC-S

Following, HC-S algorithm is proposed for portfolio optimization.

The current solution has two neighbours or possible candidate solutions. Elements of vector y in the range of 0 to 100 are randomly generated. The number of elements of y is equal to the number of asset/stocks (as explained in Section 3.1). The randomly picked position in y is denoted as pos. ThP is a small percentage, which we refer to as *threshold percentage*, by which elements of y will be varied to get the next neighbour.

The neighbourhood definition is to pick just one position (pos) in the current solution, y, at random. After picking the random position in the current solution, one neighbour is obtained by adding ThP to that position and another is obtained by subtracting ThP on the same position. This gives two neighbours (two possible candidate solutions) to be compared with the current solution, at random. The first better candidate solution (neighbour) to be picked is taken to be the current solution out of the possible candidate solutions. If no better solution is found, another position, pos, in y is picked at random. The procedure is repeated for positions picked at random for a preset number of iterations, or until local maximum.

Given mean returns of all stocks in column vector denoted as *retasset*, given assets' co-variances/deviations matrix, denoted as *dev*, and given λ as the level of risk aversion; below is the procedure for **HC-S**.

Pseudo Code for HC-S

```
Procedure HC-S (ThP, λ, retasset, dev)

        Randomly generate initial
        current solution y
    Begin
        Repeat
            Pick random position, (pos), in current solution y

            yplus = y                                % Generate yplus from current solution %

            yminus = y                               % Generate yminus from current solution %

            yplus(pos) = yplus(pos)*(1 + ThP)        % Get a neighbour of current solution %

            yminus(pos) = yminus(pos)*(1 - ThP)      % Get second neighbour of current solution %

            y = move_to_neighbour (y, yplus,
            yminus, λ, retasset, dev)                % Pick a better neighbour solution %

            At this point call function for
            Cardinality constraint, if applicable,
            (after every pre-set number of
            iterations).

        Until stopping criterion                     % Stopping criterion was; no neighbour is
                                                     better than current solution or preset
                                                     maximum number of iterations reached%

    End
```

Fig1. Hill climbing procedure of **HC-S**

Pseudo Code for a **function for searching for better neighbouring solution**

```
Function Move_to_neighbour (y, yplus, yminus, λ, retasset, dev)
Begin
    $x_i = y_i / \sum_{i=1}^{n} y_i$                         % Find weights, x, of all the assets n in portfolio%
    $xplus_i = yplus_i / \sum_{i=1}^{n} yplus_i$             % Find weights, xplus, of all the assets n in portfolio%
    $xminus_i = yminus_i / \sum_{i=1}^{n} yminus_i$          % Find weights, xminus, of all the assets n in portfolio%

    xvalue = objectvalue (x, λ, retasset, dev)               % Calculate objective value of portfolio x and denote as xvalue. %

    xplusvalue = objectvalue (xplus, λ, retasset, dev)       % Calculate objective value of portfolio xplus and denote as xplusvalue. %

    xminusvalue = objectvalue (xminus, λ, retasset, dev)     %Calculate objective value of portfolio xminus and denote as xminusvalue. %

    if xplusvalue >xvalue then y=yplus                       % Return yplus if it is better than y. %
    end if

    if xminusvalue>xvalue then y=yminus                      % Return yminus if it is better than y. %
    end if
    return y
End
```

Fig 2. Function to search for better neighbouring solution

The function below measures the quality of a portfolio. The function calculates the objective/objective value from equation (1) in section 3 above. The mean returns and co-variances of all assets/stocks are initially calculated from the daily prices in the main program. They are used to calculate the expected return and risk of a portfolio. The return and risk of a portfolio calculated are used to measure the quality of a portfolio.

Pseudo Code for calculating Objective function value

```
    Function Objectvalue (x, λ, retasset, dev)
Begin
     retport=scalar/dot product(retasset, x)        % Calculate effective expected return of
                                                    portfolio. %
      risk = x*dev*x'                               % Calculate effective risk/variance of
                                                    portfolio. %
     fitvalue = λ* retport – (1 - λ)*risk
                                                    %Calculate objective/objective value
    return fitvalue                                 according to equation (1) in section 3 above. %
End
```

Fig3. Function to calculate objective/fitness value

3.4 Design of HC-C

3.4.1 Neighbourhood Function for HC-C

Following, HC-C algorithm is proposed for portfolio optimization.

The sequence of all the positions of the elements of initial random solution y is randomized (so that the elements are not sequentially picked). If first position in the random sequence gives no better solution, next position is picked and so on. This is in contrast with HC-S, where only one position is examined in each hill-climbing step. Thus, HC-C searches a larger space. This will potentially help it to find better solutions. The cost of doing so is increased computation over HC-S. For the same amount of computation time, HC-S will be able to restart more times than HC-C.

The randomly picked position in y is denoted as pos. ThP is a small percentage, which we refer to as *threshold percentage*, by which elements of y will be varied to get the next neighbour.

The neighbourhood definition is to pick one position (pos) in the current solution. After picking the random position in the current solution, one neighbour is obtained by adding ThP to that position and another is obtained by subtracting ThP to the same position. This gives two neighbours (two possible candidate solutions) at a time to be compared with the current solution, at random order. The first better candidate solution (neighbour) to be picked becomes the current solution out of the possible candidate solutions. On getting a better solution, the sequence of the positions of the elements of y is again randomized. The overall procedure is repeated for a number of iterations, or until local maximum.

Given mean returns of all stocks in column vector denoted as *retasset*, given assets' co-variances/deviations matrix, denoted as *dev*, and given λ as the level of risk aversion; below is the procedure for **HC- C**.

Pseudo Code for HC-C

```
Procedure HC-C (ThP, λ, retasset, dev)
Randomly generate initial current solution y
Begin
    Repeat
        pick random position, (pos), in current solution y
        yplus = y                                   % Generate yplus from current solution. %
        yminus = y                                  % Generate yminus from current solution. %
        yplus(pos) = yplus(pos)*(1 + ThP)           % Get a neighbour of current solution. %
        yminus(pos) = yminus(pos)*(1 - ThP)         % Get second neighbour of current solution. %
        yb4=y                                       % keep record of current solution y. %
        y = move_to_neighbour (y, yplus,            % Pick a better neighbour solution. %
            yminus, λ, retasset, dev)
        Randomly change the sequence of             % Provides randomness. %
            the positions
        while y=yb4 do                              % Looks for better solution in the random
                                                      sequence. (pos) is any position. %
            yplus = y                               % Generate yplus from current solution %
            yminus = y                              % Generate yminus from current solution. %
            yplus(pos) = yplus(pos)*(1 + ThP)
            yminus(pos) = yminus(pos)*(1 - ThP)     % Get a neighbour of current solution. %
            if (all positions in the sequence          % Get second neighbour of current
    have been checked for better solution) then        solution. %
    break while loop
            end if

        end while

    At this point call function for Cardinality
    constraint, if applicable, after every pre-set
    number of iterations.

            Until halting criterion is met

                                                       % Halting criterion was; no neighbour is
    End                                                better than current solution or maximum
                                                       number of iteration is reached.
```

Fig 4. Hill climbing procedure of **HC- C**

The function y = Move_to_neighbour (y, yplus, yminus, λ, retasset, dev) is similar to that used in HC-S above.

The function fitvalue = Objectvalue(x, λ, retasset, dev) is similar to that used in HC-S above.

3.5 Design of HC-S-R

3.5.1 Neighbourhood Function for the Hill Climbing Algorithm HC-S-R

HC-S-R is like HC-S, except that the ThP is reduced over time. In other words, it searches the neighbourhood with finer and finer steps. Following, HC-S-R algorithm is proposed for portfolio optimization.

Elements of vector y are randomly generated. The number of elements of y is equal to number of asset/stocks. The randomly picked position in y is denoted as pos. ThP is a small percentage, which we refer to as threshold percentage, by which elements of y will be varied to get the next neighbour.

Similar to HC-S, the neighbourhood definition of HC-S-R is to pick just one position (pos) in the current solution, y, at random. After picking the random position in the current solution, one neighbour is obtained by adding ThP to that position and another is obtained by subtracting ThP on the same position. This gives two neighbours (two possible candidate solutions) to be compared with the current solution, at random. The first better candidate solution (neighbour) to be picked is taken to be the current solution out of the possible candidate solutions. If no better solution is found, another position, pos, in y is picked at random. The procedure is repeated for positions picked at random for a preset number of iterations, or until local maximum.

In HC-S-R ThP is reduced over time. That is after a preset number of iterations or on reaching local maximum, ThP is repeatedly reduced to be half the previous value until it reaches the preset minimum ThP value, denoted as minThP.

Given mean returns of all stocks in column vector denoted as *retasset*, given assets' co-variances/deviations matrix, denoted as *dev*, and given λ as the level of risk aversion; below is the procedure for **HC- S-R**.

Pseudo Code for HC-S-R

```
Procedure HC-S-R (ThP, minThP, λ, retasset, dev)
        Randomly generate initial current solution y

        set minThP
    Begin
        Do while ThP>minThP

        Repeat

      Pick random position, pos, in current solution y      % Generate yplus from current solution %

        yplus = y                                           % Generate yminus from current solution %

        yminus = y                                          % Get a neighbour of current solution %

        yplus(pos) =  yplus(pos)*(1 + ThP)
                                                            % Get second neighbour of current solution %
         yminus(pos) = yminus(pos)*(1 - ThP)

                                                            % Pick a better neighbour solution %
    y = move_to_neighbour (y, yplus, yminus, λ, retasset, dev)

    At this point call function for Cardinality
    constraint, if applicable, (after every pre-set         % Stopping criterion was: no
    number of iterations).                                  neighbour is better than current
                                                            solution or preset maximum number of
        Until stopping criterion                            iterations reached%

            ThP=ThP/2

        End while
    End
```

Fig5 Hill climbing procedure of HC-S-R

The function Move_to_neighbour (y, yplus, yminus, λ, retasset, dev) is the same as that used in HC-S above.

The function Objectvalue (x, λ, retasset, dev) is the same as that used in HC-S above.

3.6 Design of HC-C-R

3.6.1 Neighbourhood Function for the Hill Climbing Algorithm HC-C-R

HC-C-R is like HC-C, except that the ThP is reduced over time. In other words, it searches the neighbourhood with finer and finer steps. Following, HC-C-R algorithm is proposed for portfolio optimization.

Elements of vector y are randomly generated. The number of elements of y is equal to number of asset/stocks. The randomly picked position in y is denoted as pos. ThP is a small percentage, which we refer to as threshold percentage, by which elements of y will be varied to get the next neighbour.

Similar to HC-C, the sequence of all the positions of the elements of initial random solution y is randomized (so that the elements are not sequentially picked). If first position in the random sequence gives no better solution, next position is picked and so on. This is in contrast with HC-S, where only one position is examined in each hill-climbing step. Thus, HC-C searches a larger space. This will potentially help it to find better solutions. The cost of doing so is increased computation over HC-S. For the same amount of computation time, HC-S will be able to restart more times than HC-C.

The randomly picked position in y is denoted as pos. ThP is a small percentage, which we refer to as *threshold percentage*, by which elements of y will be varied to get the next neighbour.

The neighbourhood definition is to pick one position (pos) in the current solution. After picking the random position in the current solution, one neighbour is obtained by adding ThP to that position and another is obtained by subtracting ThP to the same position. This gives two neighbours (two possible candidate solutions) at a time to be compared with the current solution, at random order. The first better candidate solution (neighbour) to be picked becomes the current solution out of the possible candidate solutions. On getting a better solution, the sequence of the positions of the elements of y is again randomized. The overall procedure is repeated for a number of iterations, or until local maximum. In HC-C-R ThP is reduced over time. That is after a preset number of iterations or on reaching local maximum, ThP is repeatedly reduced to be half the previous value until it reaches the preset minimum ThP value, denoted as minThP.

Given mean returns of all stocks in column vector denoted as *retasset*, given assets' co-variances/deviations matrix, denoted as *dev*, and given λ as the level of risk aversion; below is the procedure for **HC- C-R**.

Pseudo Code for HC-C-R

Procedure HC-C-R (ThP, minThP, λ, retasset, dev)

```
    Randomly generate initial current
    solution y
Begin
    Do While ThP>minThP
    Repeat

        pick random position, pos, in
        current solution y
                                        % Generate yplus from current
        yplus = y                       solution %
        yminus = y                              % Generate yminus from current solution %

        yplus(pos) = yplus(pos)*(1 + ThP)       % Get neighbour of current solution. %

        yminus(pos) = yminus(pos)*(1 - ThP)     % Get second neighbour of current solution. %
        yb4=y                                   % keep record of current solution y %

        y = move_to_neighbour (y, yplus,        % Pick a better neighbour solution. %
        yminus, λ, retasset, dev)

        Randomly change the sequence            % Provides randomness. %
        of the positions

        while y=yb4 do                          % Looks for better solution in the random
                                                sequence. (pos) is any position. %

            yplus = y                           % Generate yplus from current solution %

            yminus = y                          % Generate yminus from current solution %

            yplus(pos) = yplus(pos)*(1 + ThP)   % Get neighbour of current solution. %

            yminus(pos) = yminus(pos)*(1 - ThP) % Get second neighbour of current solution. %

            if (all positions in the sequence
            have been checked for better solution)
            then break while loop
            end if

        end while

    At this point call function for Cardinality
    constraint, if applicable, after every pre-set
    number of iterations.

    Until halting criterion is met

        ThP=ThP/2                               % Halting criterion was no neighbour is
End While                                       better than current solution or maximum
End                                             number of iteration is reached.
```

Fig6. Hill climbing procedure of HC-C-R

The function Move_to_neighbour (y, yplus, yminus, λ, retasset, dev) is the same as that used in HC-S above.

The function Objectvalue (x, λ, retasset, dev) is the same as that used in HC-S above.

3.7 Application of GLS

Below is the pseudo code of GLS [55, 57] applied in finding the optimum portfolio of n assets; showing GLS application using HC-C-R.

Pseudo Code of GLS

```
Procedure GuidedLocalSeach (p, g, λ, [I1, . . . , IM], [c1, . . . ,cM], M)
begin
        k=0;
        s0 is randomly generate initial solution (p);
        % set all penalties to 0 %
        for i=1: M do
                pi =0;
                % define the augmented objective function %
                h=g+ λ*∑ pi *Ii;
        while StoppingCriterion do
        begin
                sk+1 =Hill-climbing method HC-C-R (sk,h);   %the method HC-C-R is
                described above

                % compute the utility of features %
        for i=1: M do
                utili =Ii(sk+1) *ci/(1+ pi);
                /% penalize features with maximum utility %
        for each i such that utili is maximum do
                pi = pi+1;
        $k$=$k$+1;
    end
        $s*$ is best solution found with respect to objective function $g$;
    return $s*$;
end
```

Fig7 Procedure of GLS

Where *p* is the problem, *g* is the objective function, *h* is the augmented objective function, λ is a parameter, *Ii* is the indicator function of feature *i*, *ci* is the cost of feature *i*, *M* is the number of features, *pi* is the penalty of feature *i*,.

3.8 Design of a Function for Handling Cardinality Constraint

Following, a function for handling cardinality constraint in portfolio optimization problem is proposed.

Given a cardinality constraint, K, the function below takes a number-of-assets given by (K + 1) assets. Out of all the (K+1) assets, after a set number of iterations, one asset with minimum weight is replaced by a new asset from the stock market pool/list. Therefore the best K number of assets will finally remain when at the end of the searching process an asset with minimum weight out of all assets will be removed. The weights of the remaining K assets are then scaled to 1 to keep the budget constraint. To further reduce the administration cost, the assets with trivial weight can be removed before scaling to one: example, by rounding up all the values of the weight of the assets to say five decimal places, the ones resulting in zero weight can be removed from the investment list.

Below, y is vector of K+1 integer assets, Number-of-assets is therefore length(y) and x represents vector of asset weights.

Pseudo Code for the Function for Handling Cardinality Constraint

```
Function CardinalityConstraint (y, return-data, total-returndata, asset-position, new-asset)
        Begin
        $x_i = y_i / \sum_{i=1}^{n} y_i$                % Find weights, x, of all the assets n in portfolio.
        minweight=min (x)                               % Find the most minimum weight, in all the assets in portfolio.
    for i=1: number-of-assets
        If x (i)=minweight                              % If asset has the minimum weight then
    return-data (i) = total-returndata (new-asset)      % Replace the asset y(i) in return-data i.e in portfolio (by taking a new-asset from total- returndata i.e. stock list).
    x (i)=rand()                                        % Initialize random weight for new-asset y (i).
    asset-position (i) = new-asset                      % Keep record of positions of new-assets as they are in the stock market list.
    new-asset = new-asset+1                             % Access the next asset from stock market list and make it new-asset.
        end if
    end for
    return [y, asset-position, return-data, new-asset]  % Outputs of the function.
    end
```

Fig8. Function for Cardinality Constraints

3.9 Design of a Function for Handling Holding Size Constraints

Following, a function for handling maximum and minimum holding size constraints in portfolio optimization problem is proposed.

Here the weights of all assets are to be adjusted so that each asset conforms to maximum and minimum holding size constraints. Maximum Holding size constraint reduce exposure of the capital to one asset. Minimum holding size avoids unnecessary administration costs of assets with negligible weight budget wise.

The function/procedure below iteratively takes a minimal portion, e, off the weight of assets with weight above the maximum holding size and adds it on assets with weights below the minimum holding size. If there are no assets below minimum holding size then e is added on asset with weight below maximum holding size which will result in best solution compared to all assets with weight below maximum holding size. During this process the weight of the asset must not exceed maximum holding size when e is added to it.

In this thesis, cardinality constraint was first incorporated. When cardinality constraint was met then minimum and maximum holding size were incorporated.

Pseudo Code for the Function for Handling Holding Size Constraints

```
Function holdingsizesConstraint (y, xmax, xmin)
Begin
        Number-of-asset = length (y)     % Find size of vector y (that is the number of
                                         assets in portfolio).
                                         % Find weights, x, of all the assets n in
        $x_i=y_i/\sum_{i=1}^{n} y_i$     portfolio.

For iter =1: number-of-steps do

For i =1: number-of-assets do

        set e                            % Set portion of weight to decrease or
                                         increase to asset weight (e.g 0.5)%.
        If x (i) > xmax                  % If asset is exceeding maximum limit.

        x (i) = x (i) - e                % sell/decrease some of shares of the asset
                                         exceeding maximum limit.
```

```
          x (j) = x (j)+e              % buy/increase shares of the best performing
                                       asset.
          else if  x (i) < xmin        % If asset is below minimum limit.

          x (i) = x (i)+e              % buy/increase more shares of the asset
                                       below minimum limit.
          x (k) = x (k) - e            % sell/decrease shares of the worst
                                       performing asset.
          end if
     End For
     End For
     return x
     end
```

Fig9. Function for Holding Size Constraints

4. BENCHMARKING THE ALGORITHMS ON THE MARKOWITZ MODEL

The algorithms are applied on a benchmark problem of solving standard Markowitz model as described in equations (1), (2), (3) under basic constraints (4) and (5). This problem has exact solution by standard methods. The standard method used for comparison is Quadratic Programming. Then the results from the five algorithms proposed are compared with the results by Threshold Accepting. This is a well established heuristic algorithm in portfolio optimization. The effectiveness, efficiency and reliability of the algorithms are further analyzed.

The assets and their return data used for applications in the algorithms are from DAX stock exchange. The data used were daily returns over 1001 days.

4.1 Efficient Frontier

The standard Markowitz model [Section 2.3.1], equations (1) to (5), was used to find an optimum portfolio of 230 assets.

Below is the efficient frontier obtained by tackling the Markowitz model using HC-S. It was applied on 230 assets portfolio. The 230 assets are from DAX stock exchange. The data used were daily returns over 1001 days.

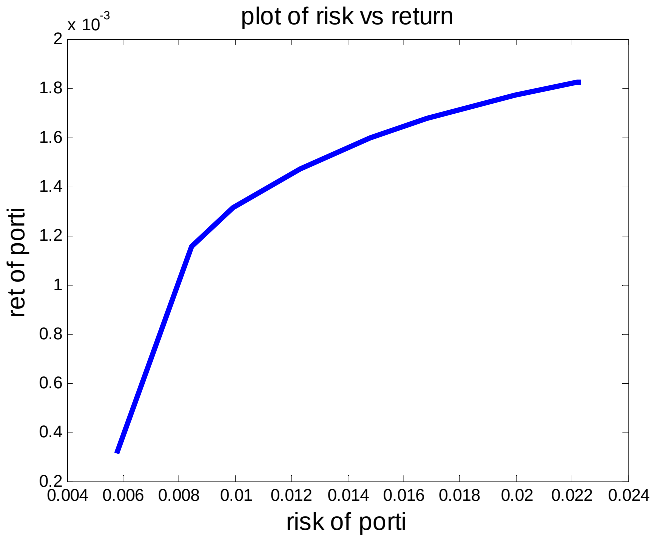

Fig10. Efficient Frontier of 230 assets portfolio using HC-S

4.2 Benchmarking HC-S and HC-C with Quadratic Programming method

Here HC-S and HC-C are benchmarked on the Markowitz model (section 4.1 above). They are tested on 10 assets portfolio. The results are compared with Quadratic Programming (QP), which is a standard method.

The following are the algorithms.
HC-S: Hill Climbing-Simple (Section 3.3)
HC-C: Hill Climbing-Complete (Section 3.4)

Below are experimental results on benchmarking HC-S and HC-C with QP. They are the percentage values in a table and corresponding bar charts of the weights of 10 assets portfolio. They were obtained by finding minimum variance portfolio (Markowitz model with $\lambda=0$ in expression (1) above) by quadratic programming method and by the hill-climbing algorithms

HC-S and HC-C. Quadratic programming (QP) produces exact solution so results by HC-S and HC-C are compaired with its results to see how accurate methods they are.The values show the relative weights (of total bugdet) to be invested in each asset . The results (weights) by algorithms HC-S and HC-C are from the best solution after 100 runs.

Table 1. Experimental results on benchmarking HC-C and HC-S with Quadratic Programming

algorithm	**Weight in each asset**
Quadratic progamming	0.0053 0.0802 0.1150 0.3191 0.1622 0.0599 0.0419 0.0067 0.0356 0.1741
HC-S	0.0053 0.0801 0.1150 0.3193 0.1620 0.0601 0.0419 0.0067 0.0357 0.1739
HC-C	0.0053 0.0802 0.1150 0.3192 0.1622 0.0599 0.0419 0.0067 0.0356 0.1740

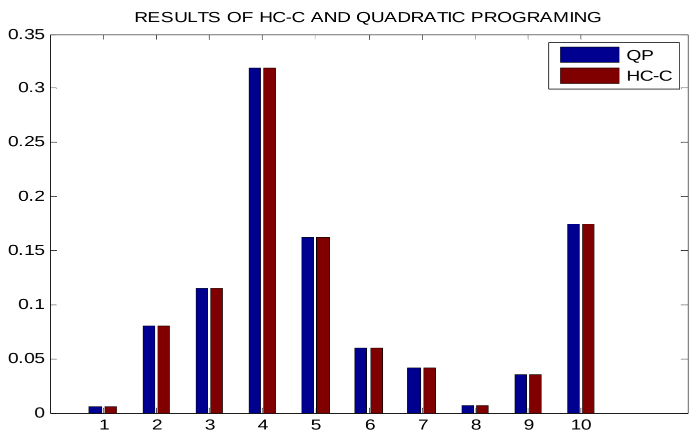

Fig11. HC-C (red) and Quadratic Programming (blue)

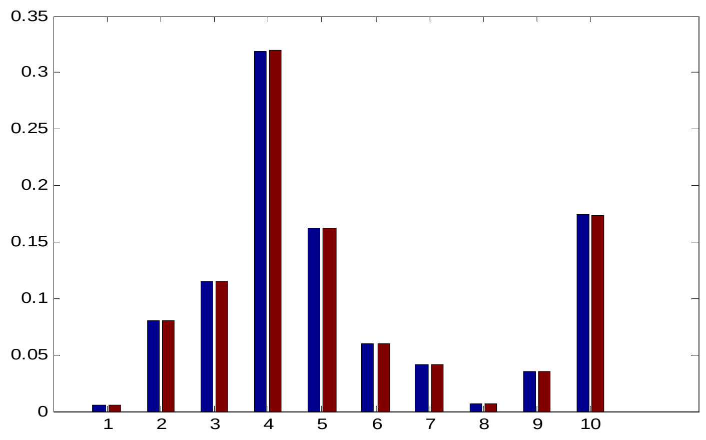

Fig12. Comparison of HC-S (red) with Quadratic Programming (blue)

It is observed from the results (in table and bar chart form) that solutions obtained by HC-C and HC-S do not differ much from the exact solution by quadratic programming (QP).

Variance/risk was calculated from the weights obtained by the methods QP, HC-S and HC-C. The three methods attained the same low portfolio risk of 6.9751e-005. Attaining the same value of risk as QP depicts that the algorithms HC-S and HC-C attain very accurate solutions.

The bar charts shows the results of the algorithms HC-C and HC-S in comparison to QP. In the two figures, the blue bars are that of Quadratic Programming (QP) and the red ones are of HC-S and HC-C respectively.

The similar height bars of HC-C compared to QP and those of HC-S compared to QP also depict that the algorithms HC-S and HC-C give very accurate solutions.

4.3 Benchmarking the Algorithms using Threshold Accepting

4.3.1 Experimental Results

HC-S, HC-C, HC-S-R, HC-C-R and GLS are benchmarked on the Markowitz model, equation (1) in section 4. They are tested on 100 assets portfolio. The results are compared with Threshold Accepting, which is a well established Hill Climbing algorithm in portfolio selection and optimization.

The following are the algorithms that are evaluated;
HC-S: Hill Climbing-Simple [section 3.3]
HC-C: Hill Climbing-Complete [section 3.4]
HC-S-R: HC-S with reducing ThP [section 3.5]
HC-C-R: HC-C with reducing ThP [section 3.6]
GLS: Guided Local search [section 3.7 & 3.6]

The following are the explanations of notations of the algorithms used in presenting results;

HC-C (9e+5) is HC-C with 9e+5 iterations

HC-C-R (0.1, 0.01, 9e+5) is HC-C-R with starting ThP = 0.1, half ThP every 9e+5 iterations, until ThP is below 0.01. The above number of iterations was given on every ThP but the program was to stop on reaching a local optimum.

HC-S (9e+6) is HC-S with 9e+5 iterations.

HC-S-R (0.1, 0.01, 9e+5) is HC-S-R with starting ThP = 0.1, half ThP every 9e+5 iterations, until ThP is below 0.01. The above number of iterations was given on every ThP but the program was to stop on reaching a local optimum.

GLS (700): Guided Local search with 700 iterations sitting on HC-C-R (0.1, 0.01, 500)

Below is a table with experimental results on the portfolio optimization on 100 stocks from DAX stock exchange; taken after 100 runs. The results show the values of objective function, number of functional evaluations required to reach final objective value, and average time in seconds for one run to converge to local maximum (final solution). The Best Final Objective value is the highest objective function value obtained in all 100 runs. Final objective values obtained in each run were recorded and so below is the Mean, STD and Worst of Final objective values in all the 100 runs. The Mean and STD of Number of functional evaluations to reach final objective value, of the 100 runs, are also given.

Table 2 Experimental results on Portfolio optimization on 100 stocks, after 100 runs.

Algorithm		**GLS** (700) (on **HC-C-R** (0.1, 0.01, 500))	**HC-C-R** (0.1, 0.01, 9e+5)	**HC-S-R** (0.1, 0.01, 9e+5)	**HC-C** (9e+5)	**HC-S** (9e+5)	**Threshold Accepting** (9e+5)
Best Final Objective value		0.000596	0.000596	0.000596	0.000596	0.000596	0.000588
Final objective value	Mean	0.000596	0.000595	0.000594	0.000595	0.000594	0.000563
	STD	1.4e-10	3.32e-6	7.32e-6	2.47e-6	6.46e-6	3.46e-5
	Worst	0.000596	0.000572	0.000572	0.000572	0.000559	7.2563e-5
No. of Functional evaluations to final objective value	Mean	2.1e+5	3.2e+4	3.2e+4	3.0e+5	2.7e+5	3.0e+5
	STD	2.8e+3	943	850	7400	6800	1770
Average time for 1		43.37	28.05	10.84	100	39.0	704.7

run (in sec)							

STD =Standard Deviation

4.3.2 Statements on the Results above

4.3.2.1 HC-S is better than T.A

HC-S managed to attain higher best final objective value (0.000596) than Threshold Accepting (0.000588). The best final objective values are higher and similar in GLS, HC-C-R, HC-S-R, HC-C and HC-S showing that the five methods are more robust than Threshold Accepting as they better escape local optima.

To understand the significance of the difference in final objective value we look at the best final objective value of HC-S which is 0.000596. This translates to a return of 0.14% and a risk of 1.34% one day after investment, of the 100 stocks considered. The best final objective value of Threshold Accepting, 0.000588, translates to a return of 0.13% and a risk of 1.54% one day after investment. The following days could include compounded interest on the original capital. From the return and risk figures, it is observed that you incur more risk but expect less return when you use the Threshold Accepting rather than HC-S to find an optimal portfolio.

The mean of final objective value of HC-S is higher (0.000594) than that of Threshold Accepting (0.000563). The worst final objective of HC-S is a lot better (0.000559) than that of Threshold Accepting (7.2563e-5). The STD of mean of final objective value of HC-S (6.46e-6) is 10 times less that of Threshold Accepting (3.46e-5).

The number of functional evaluations for HC-S was 2.7e5 while that of Threshold Accepting was 3.0e5. HC-S was faster as it required less number of functional evaluations. The STD of the number of functional evaluations of HC-S (6800) is more than that of Threshold Accepting (1770). Considering the time in seconds for one run to converge to best final objective value, Threshold Accepting (704.7), required more time than HC-S (39.0). This shows that it is far more expensive (time wise) to compute neighbourhood function of Threshold Accepting than that of HC-S.

A t-test was performed on final objective values and on the number of functional evaluations to final objective of the 100 runs. Both outcomes displayed a rejection of the null hypothesis at the 5% (default value) significance level. The t-test was performed using Mat-lab (R2010a).

Furthermore, to use Threshold Accepting, one has to first calculate and sort threshold sequences according to a certain problem. These are the sequences by which poor solutions will be accepted to avoid being trapped in a local optimum. The process makes Threshold Accepting quite cumbersome.

4.3.2.2 HC-C is better than HC-S

The best final objective values are high and similar in both HC-C and HC-S. This shows that both methods are good at escaping local optimum and locating high quality solutions given enough number of runs (in this case 100 runs).

However, the mean of final objective value of HC-C is higher (0.000595) than that of HC-S (0.000594). The worst final objective of value of HC-C is a lot better (0.000572) than that of HC-C (0.000559). The STD of mean of final objective value of HC-S (6.46e-6) is more than twice that of HC-C (2.47e-6). This shows that HC-C is more accurate and reliable than HC-S. The mean of the number of functional evaluations for HC-S was 2.7e5 while that of HC-C was 3.0e5. So HC-S was faster as it required less number of functional evaluations. The STD of the number of functional evaluations of HC-S (6800) is less than that of HC-C (7400). This again shows that HC-S is reliably faster than HC-C. Of all the five proposed algorithms, HC-C has the highest mean of the number of functional evaluations followed by HC-S indicating that they are the least efficient of all the five algorithms.

Considering the time in seconds for one run to converge to best final objective value, HC-C (100) required far more time than HC-S (39).

4.3.2.3 "R" is better than No "R"

The effect of "R" in HC-C-R is seen in speed to reach final objective value. This means repeatedly reducing ThP (instead of fixing ThP) made the algorithms more efficient. The mean of number of functional evaluations in HC-C-R is 3.2e4 while for HC-C is 3.0e5. Similarly the mean of number of functional evaluations for HC-S-R is 3.2e4 while that of HC-S is 2.7e5. Average time to converge to final objective value for HC-C-R was 28.05 seconds while that of HC-C was 1000 seconds. Again, average time to converge to final objective value for HC-S-R was 10.84 while that of HC-S was 65.

Of all the above algorithms, HC-S-R and HC-C-R required less number of functional evaluations to final objective as the Mean and STD of Number of Functional evaluations to final objective of HC-S-R (3.2e4, 850) and HC-C-R (3.2e4, 943) were the least of all. The time elapsed for HC-S-R to finish the 1 run was the smallest, making it the fastest of all the algorithms. So HC-S-R is more efficient than HC-S and HC-C-R is more efficient than HC-C.

The mean of final objective value of HC-C-R (0.000595) is the same as that of HC-C (0.000595), and the mean of final objective value of HC-S-R (0.000594) is the same as that of HC-S (0.000594). This indicates that HC-C-R and HC-C are similar in effectiveness to find better solution, and also HC-S-R and HC-S are similar in effectiveness to find better solution.

4.3.2.4 GLS is better than No GLS

By adding penalties every time there was a local optimum, GLS managed to attain the best final objective value in all 100 runs. The mean of the final objective value is the same as the value of best final objective value, that is 0.000596, and the STD of the mean of the final objective value is exceedingly small (1.3997e-10). The worst final objective value was also the same as the best final objective value (0.000596). This demonstrates that GLS on HC-C-R was better in reliability to find best final objective value than HC-C-R as HC-C-R attained lower mean of final objective value (0.000595) and the worst final objective value was a lot lower than that of GLS.

The overall results demonstrate that GLS (sitting on HC-C-R) manages to find better solutions (higher mean of the final objective value) than the other four algorithms.

From the results above it is further noted that GLS (sitting on HC-C-R) was better than HC-S and HC-C, in both mean of the final objective value (effectiveness in finding better solution) and mean of number of functional evaluations (efficiency in finding better solution).

4.3.3 Conclusion on the Results Above

GLS and four novel hill-climbing algorithms, HC-C, HC-S, HC-S-R and HC-C-R have been described and implemented in portfolio optimization problem. They were tested on the Markowitz model; in finding weights for 100 stocks in portfolio optimization, where a budget constraint is imposed and no short-selling is permitted. Results demonstrate that HC-S manages to find significantly better solutions than Threshold Accepting, an established algorithm for portfolio optimization. Results also show that HC-C manages to find better solutions than HC-S. The overall results demonstrate that GLS sitting on HC-C-R manages to find better solutions than all the four algorithms. The small standard deviations observed show that GLS, HC-C, HC-S, HC-S-R and HC-C-R find solutions more robust than Threshold Accepting with GLS able to find best final objective value in all 100 runs. Also the technique of reducing ThP made HC-C-R and HC-S-R more efficient than HC-C and HC-S.

Following, the results of the experiments above on benchmarking the algorithms with T.A are summarized in a table below.

Table 3 Summary on benchmarking the algorithms with T.A

Algorithm	Effectiveness	Efficiency
T.A [22]	Well established algorithm in portfolio optimization [5, 15, 17, 23, 58, and 48].	
HC-S [section 3.3]	More effective in finding better solution than **T.A** <refer to Section 4.3.2.1>	More efficient and quite faster time wise than **T.A** <refer to Section 4.3.2.1>
HC-C [section 3.4]	More effective in finding better solution than **HC-S** <refer to Section 4.3.2.2>	A bit less efficient than **HC-S** <refer to Section 4.3.2.2>
HC-S-R [section 3.5]	Similar with **HC-S** in effectiveness to find better solution. <refer to Section 4.3.2.3>	More efficient than **HC-S** <refer to Section 4.3.2.3>
HC-C-R [section 3.6]	Similar with **HC-C** in effectiveness to find better solution. <refer to Section 4.3.2.3>	More efficient than **HC-C** <refer to Section 4.3.2.3>
GLS [55](sitting on HC-C-R**)** [section 3.7& 3.6]	More reliable in finding better solution than **HC-S, HC-C, HC-S-R, HC-C-R**<refer to Section 4.3.2.4>	More efficient than **HC-S, HC-C.** <refer to Section 4.3.2.4>

5. HANDLING CARDINALITY CONSTRAINTS

In this extended portfolio optimization problem, the Markowitz model is implemented together with Cardinality constraint. GLS, HC-C, HC-S, HC-S-R and HC-C-R are implemented in a portfolio optimization problem with cardinality constraint.

In these experiments, the problem was to get 10 best stocks out of 233 stocks in DAX and then find optimal weights for the 10 stocks chosen that would suffice the Markowitz model. In the Markowitz model, budget constraint is imposed and no short-selling is permitted (basic constraints).

The following are the algorithms that are evaluated;
HC-S: Hill Climbing-Simple [section 3.3]
HC-C: Hill Climbing-Complete [section 3.4]
HC-S-R: HC-S with reducing ThP [section 3.5]
HC-C-R: HC-C with reducing ThP [section 3.6]
GLS: Guided Local search [section 3.7& 3.6]

5.1 Experimental Results

Cardinality constraint implemented in these experiments is to choose best 10 assets out of 233 assets of the DAX stock exchange. Cardinality of 10 stocks was chosen based on investigation in literature review where most investors wanted 20 assets/stocks or less that would suffice/meet the Markowitz model to avoid high administration cost. In these experiments, Cardinality of 10 stocks was chosen because of pre-information from returns data and the trial experiments on the data of the 233 DAX stocks. Initially 20 stocks were chosen only to find that in all trial times the total number of stocks given any weight of capital was 10 or less. This means other stocks were assigned zero weight due to their poor past performances.

ThP is the Threshold Percentage by which the original random value is reduced or increased during the searching for optimal solution. R denotes the case where ThP is reduced after a set of iterations, (in this case ThP is reduced from 5% t0 0.5%). If fixed ThP is 0.5%.

Near to maximum number of For-Loop iterations provided by MatlabR2010a was given but the programs were to stop on reaching local optimum, (2.14e9 iterations were given).

Below is the table showing the experimental results. The results show the values of objective function (equation (1), section 3.2), number of functional evaluations required to reach final objective value, and average time in seconds for one run to converge to local maximum (final solution). The Best Final Objective value is the highest objective function value obtained in all 20 runs. Final objective values obtained in each run were recorded and so below is the Mean, STD and Worst of Final objective values in all the 20 runs. The Mean and STD of Number of functional evaluations to reach final objective value, of the 20 runs, are given. The table also gives the Return and Risk of Best Portfolio and Worst portfolio found by each algorithm in the 20 runs.

Table 4 Experimental results on Portfolio optimization of 10 stocks portfolio out of 233 stocks

Algorithm		**GLS** (on **H-C-C-R**)	**HC-C-R**	**HC-S-R**	**HC-C**	**HC-S**
Best Final objective value (in 20 runs)		6.9616e-4	6.9616e-4	6.9616e-4	6.9616e-4	6.9616e-4
Final Objective value	Mean STD	6.9616e-4 2.9730e-12	6.9616e-4 1.3324e-11	6.9616e-4 2.2207e-12	6.9616e-4 1.2008e-11	6.9616e-4 6.4143e-9
Worst Final objective value (in 20 runs)		6.9616e-4	6.9616e-4	6.9614e-4	6.9616e-4	6.9613e-4
Number of Functional evaluations to final objective value	Mean STD	1.4574e+6 3.6027e+4	7.1553e+5 8.8413e+3	2.8025e+5 4.0192e+3	6.0164e+6 9.1186e+4	1.9102e+6 2.1702e+4
Best Portfolio (in the 20 runs)	Return Risk	0.0017 0.0168	0.0017 0.0168	0.0017 0.0168	0.0017 0.0168	0.0017 0.0168
Worst portfolio (in the 20 runs)	Return Risk	0.0017 0.0168	0.0017 0.0168	0.0017 0.0168	0.0017 0.0168	0.0017 0.0168

STD =Standard Deviation

5.2 Statements on the Experimental Results

5.2.1 Best Final Objective Value

All five algorithms attained the same high best objective value of 6.9616e-4. This best objective value translates to best portfolio of return 0.17% and risk of 1.68% as shown above in the row for return and risk of best portfolio.

5.2.2 Final Objective Value

Again all algorithms attained the same high mean final objective value of 6.9616e-4 which is the same as the best objective value. The STD of the mean final objective value in all the five algorithms is very low (between 2.2207e-12 for HC-S-R and 6.4143e-9 for HC-S). This high Mean of final objective value (which is equivalent to best objective value) and very low STD give a very strong indication of the high accuracy and high reliability of the algorithms in dealing with cardinality constrained portfolio optimization problem.

5.2.3 Worst Final Objective Value

The worst Final Objective values in GLS and HC-C-R and HC-C are the same as the best final objective values (6.9616e-4). This worst Final Objective value translates to worst portfolio of return 0.17% and risk of 1.68% as shown above in the row for return and risk of worst portfolio.

The worst Final objective values in HC-S-R (6.9614e-4) and HC-S (6.9613e-4) are almost similar to the best final objective value and result in the same value of return of 0.17%, and risk of 1.68%. This again shows that the algorithms are accurate and reliable with GLS, HC-C-R and

HC-C a little superior to HC-S-R and HC-S in attaining better solution. It is worth noting that all algorithms attained the same high level of return and risk in all 20 runs, indicating the reliability and accuracy of the algorithms in cardinality constrained portfolio optimization problem.

5.2.4 Number of Functional Evaluations to Final Objective Value

The Mean of the number of Functional evaluations to final objective value of HC-C-R (7.1553e+5) is less than that of HC-C (6.0164e+6), and that of HC-S-R (2.8025e+5) is less than that of HC-S (1.9102e+6). This shows that the technique of reducing ThP (after a set number of iterations or on reaching local optimum) as the searching continues helped reduce the number of Functional evaluations to final objective value ten times. So this technique increased the speed of the algorithm without affecting the quality of the final solution (Final Objective values). HC-C (6.0164e+6) has the highest mean of the number of functional evaluations followed by HC-S (1.9102e+6) indicating that it is the least efficient of the five algorithms above and HC-S (1.9102e+6) is more efficient than HC-C (6.0164e+6). GLS (1.4574e+6) is more efficient than HC-S (1.9102e+6) and HC-C (6.0164e+6). HC-S-R (2.8025e+5) is the most efficient of the algorithms.

The Mean of the number of Functional evaluations to final objective value and its STD show that HC-C-R is more efficient than HC-C and HC-S-R is more efficient than HC-S.

5.3 Conclusion on the Experimental Results

GLS and four novel hill-climbing algorithms, HC-C, HC-S, HC-S-R and HC-C-R have been implemented in a portfolio optimization problem with cardinality constraint. The problem was to pick 10 best assets/stocks out of 233 stocks in the DAX stock market that would suffice/meet the Markowitz model. In the Markowitz model, budget constraint is imposed and no short-selling is permitted. In this extended portfolio optimization problem, the Markowitz model is implemented together with cardinality constraint.

Results demonstrate that all five algorithms attained the same high best objective value. The similar Worst Final Objective values in all the five algorithms, the similar means of final objective values and the small standard deviations (STD) observed in the mean of final objective values show that GLS, HC-C, HC-S, HC-S-R and HC-C-R are reliable in finding accurate solutions.

All algorithms attained the same high level of return and risk in all 20 runs, indicating the reliability and accuracy of the five algorithms in cardinality constrained portfolio optimization problem.

The technique of reducing ThP made HC-C-R and HC-S-R more efficient than HC-C and HC-S.

Following, the results of the experiments above in the portfolio optimization problem with cardinality constraint are summarized below.

Table 5 Summary of the algorithms on handling cardinality constraint

Algorithm	**Effectiveness**	**Efficiency**
HC-S(Section 3.3)	Very effective and very reliable in finding better solution <refer to Section 5.2.1, Section 5.2.2, Section 5.2.3 >	More efficient than **HC-C** <refer to Section 5.2.4>
HC-C (Section 3.4)	Very effective and more reliable than **HC-S** in finding better solution <refer to Section 5.2.1, Section 5.2.2, Section 5.2.3 >	Very efficient <refer to Section 5.2.4>
HC-S-R (Section 3.5)	similar to **HC-S** in effectiveness in finding better solution <refer to Section 5.2.1, Section 5.2.2, Section 5.2.3 >	Most efficient, (More efficient than **HC-S**) <refer to Section 5.2.4>
HC-C-R (Section 3.6)	similar to **HC-C** and **HC-S-R** in effectiveness in finding better solution <refer to Section 5.2.1,	More efficient than **HC-C** <refer to Section 5.2.4>

	Section 5.2.2, Section 5.2.3 >	
GLS (sitting on HC-C-R) (Section 3.7 & 3.6)	Very effective and reliable in effectiveness in finding better solution <refer to Section 5.2.1, Section 5.2.2, Section 5.2.3 >	More efficient than **HC-S** and **HC-C** <refer to Section 5.2.4>

6. HANDLING HOLDING SIZES AND CARDINALITY CONSTRAINTS

GLS, HC-C, HC-S, HC-S-R and HC-C-R are implemented in a portfolio optimization problem with Holding sizes and Cardinality constraints. The problem is to pick 20 best assets/stocks or less out of 233 stocks in the DAX stock market that would sufficiently meet the Markowitz model. All the chosen 20 assets must hold not more than 30% of the capital and not less than 1% of the capital. In the Markowitz model, budget constraint is imposed and no short-selling is permitted.

6.1 Experimental Results

Cardinality constraint in these experiments is to choose the best 20 assets or less out of 233 assets of the DAX stock exchange. Cardinality of 20 stocks or less was chosen based on investigation in literature review. Investors preferred 20 assets/stocks or less that would sufficiently meet the Markowitz model to avoid high administration cost.

Holding sizes constraint in these experiments is to have all the chosen 20 assets hold not more than 30% of the capital and not less than 1% of the capital.

It is to be noted that holding size constraint was implemented after cardinality constraint was implemented. Procedure or function for holding sizes was called when the algorithms had come to an end of searching for the best solution. So the function or subroutine for implementing the holding size constraint is a heuristic independent of the algorithms. The function or subroutine implementing holding size works on the finished results or the final output of any the algorithms above. In so saying it can be also used to enforce holding size constraint in a portfolio that has exact solution under the standard Markowitz model. The function or subroutine just requires optimal weight distribution of capital and maximum and minimum holding sizes of capital in asset weights. The function removes all stocks assigned zero weight, that is high risk and very minimal return stocks, before enforcing holding size constraints in the portfolio.

The following are the algorithms that are evaluated;
HC-S: Hill Climbing-Simple [section 3.3]
HC-C: Hill Climbing-Complete [section 3.4]
HC-S-R: HC-S with reducing ThP [section 3.5]
HC-C-R: HC-C with reducing ThP [section 3.6]
GLS: Guided Local search [section 3.7 & 3.6]

ThP is the Threshold Percentage by which the original random value is reduced or increased during the searching for optimal solution. **R** denotes the case where ThP is reduced after a pre-set number of iterations, (in this case ThP is from 5% t0 0.5%). If fixed ThP is 0.5%.

Near to maximum number of For-Loop iterations provided by MatlabR2010a was given but the programs were to stop on reaching local optimum, (2.14e9 iterations were given).

Below is the table showing the experimental results. The results show the values of objective function (equation (1), section 3.2), number of functional evaluations required to reach final objective value, and average time in seconds for one run to converge to local maximum or the final solution. The Best Final Objective value is the highest objective function value obtained in all 20 runs. Final objective values obtained in each run were recorded and so below is the Mean, and STD of Final objective values in all the 20 runs. The Worst Final Objective value is the lowest objective function value obtained in all 20 runs. The Mean and STD of Number of functional evaluations to reach final objective value, of the 20 runs, are given. The table also gives the return and risk of best portfolio and worst portfolio found by each algorithm in the 20 runs.

Table 6 Experimental results on Portfolio optimization of 20 stocks portfolio out of 233 stocks, after 20 runs.

Algorithm		**GLS** (on **HC-C-R**)	**HC-C-R**	**HC-S-R**	**HC-C**	**HC-S**
Best Final Objective value (in the 20 runs)		6.1382e-4	6.1321e-4	5.9075e-4	6.1316e-4	5.8072e-4
Final Objective value.	Mean STD	5.9054e-4 1.0386e-5	5.8190e-4 1.1774e-5	5.6764e-4 7.6004e-6	5.8192e-4 1.2851e-5	5.6714e-4 8.9096e-6
Worst Final Objective value (in the 20 runs)		5.7310e-4	5.5929e-4	5.5411e-4	5.6163e-4	5.5463e-4
No. of Functional evaluations to final objective value	Mean STD	2.7612e+6 5.8657e+4	1.2462e+6 1.4491e+4	4.4061e+05 8.1226e+3	1.0493e+7 1.8735e+5	2.9344e+6 5.1404e+4
Best Portfolio (in the 20 runs)	Return Risk	0.0014 0.0127	0.0014 0.0128	0.0014 0.0132	0.0014 0.0130	0.0013 0.0120
Worst portfolio (in the 20 runs)	Return Risk	0.0013 0.0121	0.0013 0.0126	0.0013 0.0130	0.0013 0.0125	0.0013 0.0129

STD =Standard Deviation

6.2 Statements on the Experimental Results

6.2.1 Best Final Objective Value

GLS attained the highest best final objective value of 6.1382e-4. This best objective value translates to best portfolio of return 0.14% and risk of 1.27% as shown above in the row for return and risk of best portfolio. HC-S has the lowest best objective value of 5.8072e-4, which translates to return of 0.13% and risk of 1.2% as seen in the row of Best Portfolio. It is noted that all algorithms attained the same return of 0.14% except for HC-S which has lower return of 0.13% but also a lower risk level of 1.2%. This indicates that the algorithms above are reliably accurate.

6.2.2 Mean of Final Objective Value

GLS attained the highest mean of final objective value of 5.9054e-4. The STD of the final objective value in all the five algorithms is low (between 1.1774e-5 for HC-C-R and 7.6004e-6 for HC-S-R). The similar high means and low STDs of the algorithms (GLS, HC-C-R, HC-C, HC-S-R) give a strong indication of the high accuracy and high reliability of the algorithms in dealing with cardinality constraint and holding sizes constraints in portfolio optimization. The mean of HC-C-R (5.8190e-4) is similar to that of HC-C (5.8192e-4). This shows that HC-C is similar to HC-C-R in effectiveness to find better solution. The mean of final objective value of HC-S-R (5.6764e-4) is similar to that of HC-S (5.6714e-4). This shows that HC-S-R is similar to HC-S in effectiveness to find better solution.

6.2.3 Worst Final Objective Value

GLS also has the highest worst objective value (5.7310e-4) in 20 runs. This translates to a return of 0.13% and a risk of 1.21% as seen above in the row for worst portfolio. It is observed that 'GLS on HC-C-R' is consistently better than 'HC-C-R with no GLS' in best objective values, mean of final objective values and worst objective values.

HC-S-R at 5.5411e-4 has the lowest worst objective value attained in 20 runs. This translates to a return of 0.13% and a risk of 1.29% as seen in the row for worst portfolio. The worst objective value of HC-S (5.5463e-4) is a bit higher than that of HC-S-R (5.5411e-4). The return of the worst portfolio of HC-S-R (0.13%) is the same as that of HC-S (0.13%). The risk of the worst portfolio of HC-S-R (1.29%) is lower than that of HC-S (1.30%). From the row for worst portfolio, it is observed that the small difference in Worst Final Objective values of all the algorithms resulted only in the difference in the risks. All algorithms attained the same return.

It is noted that in best objective values, mean of final objective values and worst objective values HC-C is consistently better than HC-S and GLS has the best performance of the above algorithms.

6.2.4 Number of Functional evaluations to final objective

HC-S-R (4.4061e+05) has the lowest Mean of the number of Functional evaluations to final objective value of all the five proposed algorithms. This indicates that it is the most efficient of the algorithms. HC-C (1.0493e+7) has the highest mean of the number of functional evaluations followed by HC-S (2.9344e+6), so HC-S is more efficient compared to HC-C.

The Mean of the number of Functional evaluations to final objective value of HC-C-R (1.2462e+6) is less compared to HC-C (1.0493e+7) and that of HC-S-R (4.4061e+05) is less compared to HC-S (2.9344e+6). This shows that the technique of reducing ThP (after a set number of iterations or on reaching local optimum) as the search continues helped reduce the number of Functional evaluations to final objective value ten times. This technique made HC-C-R and HC-S-R more efficient than HC-C and HC-S.

it is noted that HC-C-R (1.4491e+4) has far lower STD than HC-C (1.8735e+5). Likewise HC-S-R (8.1226e+3) has far lower STD than HC-S (5.1404e+4).

The Mean of the number of Functional evaluations to final objective value and its STD show that HC-C-R is more efficient than HC-C and HC-S-R is more efficient than HC-S.

6.3 Conclusion on the Experimental Results

Four novel hill-climbing algorithms, HC-C, HC-S, HC-S-R, HC-C-R, and GLS have been implemented in a portfolio optimization problem with cardinality constraint and holding sizes constraints. The problem was to pick 20 best assets/stocks or less out of 233 stocks in the DAX stock market that would suffice/meet the Markowitz model and then have all the chosen 20 assets hold not more than 30% of the capital and not less than 1% of the capital. In the Markowitz model, budget constraint is imposed and no short-selling is permitted.

Results demonstrate that all five algorithms attained high best objective values and high mean of final objective values. In best objective values, mean of final objective values and worst objective values HC-C is consistently better than HC-S, GLS on HC-C-R is consistently better than HC-C-R with no GLS. Also the technique of reducing ThP made HC-C-R and HC-S-R more efficient and faster than HC-C and HC-S.

Following, the results of the experiments above in the portfolio optimization problem with cardinality constraint and holding sizes constraints are summarized in the table below.

Table 7: Summary of the algorithms on handling cardinality and holding sizes constraints

Algorithm	Effectiveness	Efficiency
HC-S [Section 3.3]	Effective <refer to Section 6.2.2>	more efficient compared to **HC-C** <refer to Section 6.2.4>
HC-C [Section 3.4]	More effective in finding better solution than **HC-S** <refer to Section 6.2.3>	less efficient than algorithms **HC-S, HC-S-R, HC-C-R, GLS** <refer to Section 6.2.4>
HC-S-R [Section 3.5]	Similar with **HC-S** in effectiveness in finding better solution. <refer to Section 6.2.2>	More efficient than **HC-S, HC-C, HC-C-R, GLS** <refer to Section 6.2.4>
HC-C-R [Section 3.6]	Similar with **HC-C** in effectiveness in finding better solution <refer to Section 6.2.2>	More efficient than **HC-C** <refer to Section 6.2.4>

GLS (sitting on **HC-C-R)** [Sections 3.6 & 3.7]	More reliable in finding better solution than **HC-S, HC-C, HC-S-R, HC-C-R.** <refer to Section 6.2.1, Section 6.2.2, Section 6.2.3>	More efficient than **HC-S, HC-C.** <refer to Section 6.2.4>

7. CONCLUSION

7.1 Summary of the Work Done and Discussions

Five Hill Climbing algorithms, HC-S, HC-S-R, HC-C, HC-C-R and GLS [55, 57], have been produced. They were used to tackle the portfolio optimization problem, the standard Markowitz model, where a budget constraint is imposed and no short-selling is permitted [5]. They have been demonstrated to be more effective and efficient than Threshold Accepting [5], an established algorithm for portfolio optimization. Five Hill Climbing algorithms find solutions with significantly higher objective value and require less computing time.

HC-S is more effective in finding better solution than T.A because it has two neighbours to the current solution, the two candidate solutions, to consider instead of one at any iteration. HC-S is faster than T.A because it does not take in worse candidate solutions to be the current solution. Also it is cheaper to compute neighbourhood function of HC-S that that of T.A. T.A. has threshold sequence for allowing worse solutions to be the current solution to avoid being trapped in local optima. HC-C is more effective in finding better solution than HC-S because at any iteration it searches the whole neighbourhood completely, until a better candidate solution is found.

The threshold reduction scheme “R” in HC-S-R and HC-C-R improves the efficiency of the Hill Climbing algorithms HC-S and HC-C. HC-S-R and HC-C-R produce similar results to HC-S and HC-C respectively, but with fewer functional evaluations and less time as experimental results show in Section 4.3.2.3

By adding penalties every time there was a local optimum, GLS managed to attain better solutions than all the other algorithms. So GLS improves the reliability of the above algorithms. The overall results demonstrate that GLS sitting on HC-C-R manages to find the best solutions more reliably.

The five algorithms above were also implemented in extended portfolio optimization problem: in implementing the Markowitz model together with cardinality constraint. The results demonstrate that all five algorithms are reliably accurate in tackling portfolio optimization problem with cardinality constraint. This is because results demonstrate that all five algorithms attained the same high best objective value. Also the similar Worst Final Objective values in all the five algorithms, the similar means of final objective values and the very small standard deviations (STD) observed in the mean of final objective values show that GLS, HC-C, HC-S, HC-S-R and HC-C-R are reliable in finding accurate solutions. All five algorithms attained the same high level of return and risk in all 20 runs.

The five algorithms above were also implemented in extended portfolio optimization problem: in implementing the Markowitz model with both cardinality and holding sizes constraints. Results demonstrate that all five algorithms attained high best objective values and high mean of final objective values. In best objective values, mean of final objective values and worst objective values, HC-C is consistently better than HC-S and GLS on HC-C-R is consistently better than HC-C-R with no GLS. Also the technique of reducing ThP made HC-C-R and HC-S-R more efficient and faster than HC-C and HC-S.

To conclude, I have proposed a series of algorithms and demonstrated their effectiveness, efficiency and reliability for the extended portfolio optimization problem.

The following are the produced algorithms;
HC-S: Hill Climbing-Simple [section 3.3]
HC-C: Hill Climbing-Complete [section 3.4]
HC-S-R: HC-S with reducing ThP [section 3.5]
HC-C-R: HC-C with reducing ThP [section 3.6]
GLS: Guided Local search [section 3.7 & 3.6]

Below is the table to summarize the results on the five proposed algorithms (in comparison with T.A).

Table 8 Summary on the proposed algorithms (in comparison with T.A)

Algorithm	**Effectiveness**	**Efficiency**
T.A [22]	This is the well-established algorithm in portfolio optimization [5, 15, 17, 23, 58, and 48].	
HC-S [section 3.3]	More effective in finding better solution than **T.A** < refer to section 4.3.2.1>	More efficient and faster time wise than **T.A** < refer to section 4.3.2.1>
HC-C [section 3.4]	More effective in finding better solution than **HC-S** < refer to section 4.3.2.2, Section 5.2.1, Section 5.2.2, Section 5.2.3 and section 6.2.3>	(a bit) less efficient than **HC-S** < refer to section 4.3.2.2, Section 5.2.4, Section 6.2.4>
HC-S-R [section 3.5]	Similar with **HC-S** in effectiveness to find better	More efficient than **HC-S** < refer to section 4.3.2.3,

	solution < refer to Section 4.3.2.3, Section 5.2.1, Section 5.2.2, Section 5.2.3, Section 6.2.2>	Section 5.2.4, Section 6.2.4>
HC-C-R [section 3.6]	Similar with **HC-C** in effectiveness to find better solution < refer to Section 4.3.2.3, Section 5.2.1, Section 5.2.2, Section 5.2.3 section 6.2.2>	More efficient than **HC-C** < refer to section 4.3.2.3, Section 5.2.4, Section 6.2.4>
GLS [55, 57] **(**sitting on HC-C-R**)** [section 3.7 & 3.6]	More reliable in finding better solution than **HC-S, HC-C, HC-S-R, HC-C-R** < refer to section 4.3.2.4, Section 5.2.1, Section 5.2.2, Section 5.2.3, section 6.2.1, section 6.2.2, section 6.2.3>	More efficient than **HC-S, HC-C** < refer to section 4.3.2.4>

7.2 Summary of the Contributions

The contributions of this PhD research are summarized as follows.

Five algorithms (HC-S, HC-C, HC-S-R, HC-C-R and GLS [55, 57] (sitting on HC-C-R**)**) have been produced for the extended portfolio optimization problem. They are superior to the well established algorithm in portfolio optimization known as Threshold Accepting (T.A).

The following are the produced algorithms;

HC-S: Hill Climbing-Simple [section 3.3]
HC-C: Hill Climbing-Complete [section 3.4]
HC-S-R: HC-S with reducing ThP [section 3.5]
HC-C-R: HC-C with reducing ThP [section 3.6]
GLS: Guided Local search [section 3.7 & 3.6]

HC-S is more effective in finding better solution than T.A. HC-S is also more efficient than T.A.

HC-C is more effective in finding better solution than HC-S. It is a bit less efficient than HC-S.

HC-S-R is similar with HC-S in effectiveness to find better solution. HC-S-R is more efficient than HC-S.

HC-C-R is similar with HC-C in effectiveness to find better solution. HC-C-R is more efficient than HC-C.

GLS (sitting on HC-C-R) is more reliable in finding better solution than HC-S, HC-C, HC-S-R, HC-C-R. So it is the most reliable algorithm in finding better solution of the five algorithms produced. It is also more efficient than HC-S and HC-C.

The algorithms produced attained promising results for portfolio optimization, according to the particular model they were applied to. They are also quite easy to understand and to implement. So I would argue that these algorithms also have a wider application than portfolio optimization. For instance, in other research areas, like science.

Therefore from all my work I have learnt that optimization by heuristics is a wide field now. One needs to research well to find an appropriate method for a particular problem. I also learnt that in coming up with better algorithms than the present or well-established ones, I had to study and understand the well-established algorithms first, and know their weaknesses and their strengths.

Below is the table to summarize the above algorithms and contributions of this PhD research.

Table 9 Summary of the algorithms and their contributions

Algorithm	Effectiveness	Efficiency
HC-S [section 3.3]	More effective in finding better solution than **T.A**	More efficient time wise than **T.A**
HC-C [section 3.4]	More effective in finding better solution than **HC-S**	(a bit) less efficient than **HC-S**
HC-S-R [section 3.5]	Similar with **HC-S** in effectiveness to find better solution.	More efficient than **HC-S**
HC-C-R [section 3.6]	Similar with **HC-C** in effectiveness to find better solution.	More efficient than **HC-C**
GLS [55, 57] **(**sitting on HC-C-R**)** [section 3.7 & 3.6]	More reliable in finding better solution than **HC-S, HC-C, HC-S-R, HC-C-R**	More efficient than **HC-S, HC-C**

7.3 Future work

In future more realistic, non-linear constraints like transaction costs will be incorporated.

Also the five hill climbing algorithms produced will be combined with evolutionary algorithms like genetic algorithms, to give hybrid algorithms for portfolio optimization and other applications.

8. REFERRENCES

1. Sharpe, W. F., “Portfolio theory and capital markets.”, McGraw-Hill, c2000

2. Korn, R., “Optimal portfolios : stochastic models for optimal investment and risk management in continuous time.”,2 World Scientific, 1997

3. Prigent, J., “Portfolio optimization and performance analysis”, Chapman & Hall/CRC, c2007

4. Rasmussen, M., “Quantitative portfolio optimisation, asset allocation and risk management”/ , Palgrave Macmillan, 2003

5. Maringer, D., “Portfolio management with heuristic optimization”, Dordrecht, Springer, 2005

6. Meucci, A., “Risk and asset allocation”,. Springer, 2005

7. Wilmott , P., “Paul Wilmott introduces quantitative finance.” John Wiley & Sons Ltd., 2007

8. Cuthbertson, K., Nitzsche, D., “Quantitative financial economics: stocks, bonds and foreign exchange”, (2nd ed). John Wiley, 2004

9. Brown, K. C., Reilly, F. K., “Analysis of investments and management of portfolios.” (9th ed), South-Western, c2009

10. Streichert, F., Tamaka-Tamawaki, M., “The effect of local search on the constrained portfolio selection problem.” In Proceedings of the IEEE World Congress on Evolutionary Computation (CEC2006), pages 2368– 2374, July 2006.

11. Maringer, D., “Heuristic Optimization for Portfolio management.” Nov 2008, IEEE Comp. Int. Magazine**.**

12. Markowitz, H., “Portfolio selection.”, J. Finnan, 1952. pp 77-91.

13. Markowitz, H., “Portfolio selection: Efficient diversification of investments”. John Wiley and sons, 1959.

14. Ross, S, A.: “The arbitrage theory of capital asset pricing”, the journal of economic theory, 1976, 13: 341-360.

15. Winker, P., and Maringer, D., “The Threshold Accepting Optimization Algorithm in Economics and Statistics,” in: EJ Kontoghiorges and C Gatu (eds.), Advances in Computational Economics, Finance and Management Science, Kluwer, pp 107-125.,2007

16. Korczak, J J., Roger, P., Lipinski, P., “Artificial evolution: Evolution Strategy in Portfolio Optimization, ,”; (series of Lecture Notes in Computer Science Volume 2310, 2002, pp 156-167), Springer Link

17. Gilli, M., Kellezi, E., “Heuristic Approaches for Portfolio Optimization,” 6th intl. Conf. On Computing and Economics, Barcelona, July-2000

18. Jobst, N., Horniman, M. D., Lucas, C.A., Mitra, G., “Computational Aspects of Alternative Portfolio Selection Models in the Presence Of Discrete Asset Choice Constraints.”; Quantitative Finance, 1:489–501, 2001

19. Crama, Y., Schyns, M., “Simulated annealing for complex portfolio selection problems ”, European Journal of Operational Research, (150):546–571, 2003

20. Muralikrishnan, V., “Optimization by Simulated annealing (portfolio optimization). ”, Global association of risk professionals,45-48,jun/jul 2008

21. Xiao, L., Wu, Z., “Optimization of security investment based on improved simulated annealing, ”;intl journal of business and management.; vol 3, no. 11, 2008

22. i) Dueck, G., Scheuer, T., “Threshold accepting: a general purpose optimization algorithm”, Journal of Computational Physics 1990; 90:161–175. ii) Moscato, P. and Fontanari, J. F., "Stochastic versus deterministic update in simulated annealing." Physics Letters A, 146(4):204-208, 1990

23. Winker, P., “Optimization Heuristics in Econometrics- Application of Threshold accepting”, John Wiley and sons, 2001.

24. Edward Tsang, “Foundations of Constraint Satisfaction”, Academic Press, 1993, chapter 8

25. Radziukynienė, A. Ž.; ”Evolutionary Methods for Multi-Objective Portfolio Optimization”, Proceedings of the World Congress on Engineering 2008 Vol II WCE 2008, july 2008, London, U.K

26. Ehrgott, M., Waters, C., Gasimov, R. N., Ustun, O.. “Multi objective Programming and Multiattribute Utility Functions in Portfolio Optimization”, 2006, http://www.esc.auckland.ac.nz/research/tech/esc-tr-639.pdf

27. A.Mukerjee, R. B., Deb, K., Mathur, A. P., “Multi-objective evolutionary algorithm for the risk-return trade-off in bank loan management.” International Transactions in Operational research. 9, pp. 583-597, 2002

28. Steuer, R. E., Qi, Y., and Hirschberger, M., “Portfolio Selection in the Presence of Multiple Criteria,” in Handbook of Financial Engineering, C. Zopounidis, M. Doumpos, P. M. Pardalos (Ed.), Springer, 2008, pp.3-25.

29. Li, J., Taiwo, S., “Enhancing Financial Decision Making Using Multi-Objective Financial Genetic Programming,” Proceedings of IEEE Congress on Evolutionary Computation, July 2006, pp.2171- 2178.

30. Martens, M., “Measuring and Forecasting S&P 500 Index-Futures Volatility Using High-Frequency Data”. Journal of Futures markets 22, 497-518. 2002

31. Liu, Qianqiu., “On Portfolio Optimization, how Do We Benefit from High-Frequency Data? ”, Journal of Applied Econometrics, 2009, vol. 24, issue 4, pages 560-582

32. Vijayalakshmi, G.A., Michel, T., “Evolutionary Optimisation of constrained k-means clustered assets for diversification in small portfolios”, IEEE Transactions on Evolutionary Computation, Vol.13, No.5, 2009, 1030-1053.

33. Dueck,G., Winker, P., “New concepts and algorithms for portfolio choice”, Applied

stochastic models and data analysis, vol 8, 159-178, 1992.

34. Di Tollo, G., and Roli, A., “Metaheuristics for the Portfolio Selection Problem,” IJOR Vol. 5, No. 1, 130-135, 2008

35. Chang, T.-J., Meade, N., Beasley, J. E. & Sharaiha, Y. M “Heuristics for cardinality constrained portfolio optimisation,” Computers & Operations Research vol 27, Issue 13, 1271-1302, 2000

36. Streichert, F., Ulmer, H., Zell, A., “Evaluating a hybrid encoding and three crossover operators on the constrained portfolio selection problem”. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC2004), volume 1, pages 932–939, 2004**.**

37. Jahan, M.V., Akbarzadeh-Totonchi, M.R., “From local search to global conclusions: migrating spin glass-based distributed portfolio selection”, IEEE Transactions on Evolutionary Computation, Vol.14, No.4, 2010, 591-601.

38. Noman, H., Iba, N., ”Accelerating Differential Evolution Using an Adaptive Local Search; ” IEEE transactions on Evolutionary computation 2008, vol 12; No. 1, 107-125

39. Armañanzas, R., Lozano, J.A., ”A multi-objective approach to the portfolio optimization problem”, Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, pp.1388-1395

40. Gilli, M., Maringer, D., Schumann, E., Numerical Methods and Optimization in Finance, 2011 Elsevier Inc. chp 12,13

41. Lin, D., Wang, S., and Yan, H. (2001). “A multi-objective genetic algorithm for portfolio selection.”; Technical report, Institute of Systems Science, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

42. Gilli, M., Schumann, E., Di Tollo, G., & Cabej, G., “Constructing 130/30-portfolios with omega ratio”, Journal of Asset Management, (2):94-108, 2012

43. Maringer, D., “Optimizing portfolios with ant systems”, Proceedings of the International ICSC Congress on Computational Intelligence: Methods and Applications, Bangor, Wales,

United Kingdom, pp. 288-294, 2001

44. Cura, T., "Particle swarm optimization approach to portfolio optimization, Nonlinear Analysis", Real World Applications, 2008

45. Kirkpatrick, S., Gellatt Jr., C.D., Vecchi, M.P., "Optimization by simulated annealing," Science 220, 671–680, 1983

46. Glover, F., Laguna, M., "Tabu Search", Kluwer Academic Publishers, 1997

47. Hertz, A., Widmer, M., "Guidelines for the use of meta-heuristics in combinatorial optimization", European Journal of Operational Research 151, 2003, 247–252

48. Gilli, M., Schumann, E., "Portfolio Optimization with Threshold Accepting": a Practical Guide, Ch. 9, In Stephen E. Satchell (ed), Optimizing Optimization: The Next Generation of Optimization Applications and Theory. Elsevier, 2010

49. Kung, J J. , "Multi-period asset allocation by stochastic dynamic programming," Applied Mathematics and Computation, Volume 199, Issue 1, 15 May 2008, Pages 341-348

50. Maringer, D.-G. and Winker, P., "Portfolio optimization under different risk constraints with modified memetic algorithms, " Technical Report 2003-005E, University of Erfurt, Faculty of economics, Law and Social Sciences, 2003

51. Storn, R., Price, K., "Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces," Technical report, International Computer Science Institute, Berkeley, 1995
52. Storn, R., Price, K., "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, 11(4), 341–359, 1997

53. Price, K., Storn, R., Lampinen, J., "Differential Evolution: A Practical Approach to Global Optimization, " Springer, 2005

54. Coello Coello, C., "20 yrs of Evolutionary MOOP: What has been done and what remains to be done," in computational intelligence: Principles and Practices, G. Yen and D. Fogel

(ed.) IEEE computational intelligence society; 2006

55. Voudouris, C., Tsang, E. P. K., Alsheddy, A., "Guided Local Search", published online June 2010, Wiley Online Library, Operations Research and Management Science.

56. Hoos, H. H.,   Tsang, E. P. K., "Local Search Methods", Handbook of Constraint Programming , pg 245, Ed. F. Rossi, P. van Beek and T. Walsh c, 2006 Elsevier.

57. Voudouris, C., Tsang, E.P.K. & Alsheddy, A., "Guided local search", Chapter 11, in M. Gendreau & J-Y Potvin (ed.), Handbook of Metaheuristics, Springer, 2010, 321-361

58. Gilli, M., Schumann, E., "Heuristic Optimisation in Financial Modelling", Annals of Operations Research, Vol. 193, No. 1, pp. 129-158, 2012 (http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1277114)

59. Glover, F. & Kochenberger, G.A. (ed.), Handbook of metaheuristics, Kluwer, 2003

60. Jin, N., Tsang, E. & Li, J., "A constraint-guided method with evolutionary algorithms for economic problems", Applied Soft Computing, Vol.9, Iss.3, June 2009, 924-935

61. Venkataraman, P., Applied Optimization with MATLAB Programming (2ed); John Wiley and sons, 2009

62. Zhang, Q., Li, H., Maringer, D., & Tsang, E., "MOEA/D with NBI-style Tchebycheff approach for Portfolio Management". IEEE Congress on Evolutionary Computation 2010.

63. Arnott, R.D. & Wagner, W.H., "The measurement and control of trading costs." Financial Analysts Journal, 46: 73-80.1990.

64. Jansen, R. & van Dijik, R., "Optimal benchmark tracking with small portfolios." The Journal of Portfolio Management, 28(2): 9-22. 2002.

65. Miguel, S. L., "Portfolio optimization with linear and fixed transaction costs."; Annals of Operations Research Volume 152, Number 1 / July, 2007

66. Lai, K.K., Yu, L., Wang, S.: “Mean-variance-skewness-kurtosis-based portfolio optimization.” First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS’06) Vol. 2 , 292–297 (2006)
67. Campbell R. H., Liechty, J. C., Liechty, M. W., Muller, P., “Portfolio Selection With Higher Moments”, social science research network, Dec 2004.

68. Athayde, G., & Flores,R., “Incorporating skewness and Kurtosis in portfolio optimization”, a multidimensional efficient set. In: Satchell, S., Scowcroft, A. (eds.) Advances in Portfolio Construction and Implementation, pp. 243–257. Butterworth-Heinemannpp, Oxford, 2003.

69. Maringer, D., & Parpas, Panos.: “Global Optimization of higher order moments in Portfolio selection.”, Journal of Global Optimizzation, 43:219-230, 2009.

70. Gilli, M., Winker, P., “A Review of heuristic optimization methods in econometrics”, Swiss Finance Institute Research Paper No. 08-12. Available at SSRN: http://ssrn.com/abstract=1140655, June 2008.

71. Kellerer, H., & Maringer, D., “Optimization of Cardinality Constrained Portfolios with an Hybrid Local Search Algorithm”, MIC’2001- 4th Metaheuristics International Conference, 585-589,July 2001.