A multiobjective genetic algorithm for portfolio selection problem[1]

Dan Lin[a,b,2] and **Shouyang Wang**[a]

[a]Institute of Systems Science, Academy of Mathematics and Systems Science
Chinese Academy of Sciences, Beijing 100080, P.R.China

[b]Department of Mathematics, Tianjin University, Tianjin 300072, P.R.China

lindan@iss04.iss.ac.cn swang@iss04.iss.ac.cn

Hong Yan

Department of Management, The Hong Kong Polytechnic University, Kowloon,
Hong Kong

mshyan@polyu.edu.hk

Abstract

A mean-variance models is proposed for portfolio selection with fixed transaction costs and minimum transaction lots. The portfolio seleciton problems are modeled as a non-smooth nonlinear integer programming problem with multiple objective functions. A new genetic algorithm based on NSGA-II and Geconop is designed to solve the proposed models. It is illustrated via a numerical example that the genetic algorithm can be used to solve portfolio selection problems efficiently in practice.

Keywords: Portfolio selection; Mean-variance model; Transaction costs; Minimum transaction lots; Genetic algorithm

1. Introduction

The mean-variance (MV) methodology originally proposed by Markowitz (1952) has been playing a crucial role in the theory of portfolio selection and also gained widespread acceptance as a practical tool for investment analysis. The classical

[1] This work was supported by Chinese Postdoctoral Funds, NSFC, MADIS , CAS and CERG Research Grant B-Q343.

[2] The corresponding author.

formulation of Markowitz's model for portfolio selection is a quadratic programming problem with a positive semidefinite matrix which is solvalbe in polynomial time. However, when taking into account a practical situation such as fixed transaction costs and/or minimum transaction lots (or minimum lots for simplicity), portfolio selection becomes more complicated because the model is a mathematical programming problem with mixed-integer variables and nonlinear objectives.

A number of authors have discussed the transaction costs in portfolio optimization e.g., Mao (1970), Pogue (1970), Chen, Jen and Zionts (1971), Jacob (1974), Levy (1978), Patel and Subhmanyam (1982), Mulvey and Vladimiron (1992), Dantzig and Infanger (1993), Gennotte and Jung (1994), Yoshimoto (1996), Li , Wang and Deng (2000). Most of them incorporated the transaction costs into the single-period or multi-period portfolio selection models. It is generally believed that an optimal portfolio selection problem with fixed transaction costs is a complex mathematical programming problem. Because of the complexity, the convex quadratic programming technique cannot be applied to solve such a problem (Yoshimoto 1996).

Very recently some authors included minimum lots into portfolio selection problems (Speranza 1996, Kellerer *et al*. 1997, Mansini and Spernza 1999, Konno and Wijayanayake 2001), while some of them considered transaction costs at the same time. It was shown that a portfolio selection problem with minimum lots and without any fixed costs is an NP-complete problem (Mansini and Speranza 1999), so a few heuristics have been developed and shown to be quite effective for solving a problem with minimum lots.

The genetic algorithms (GAs) were growing in popularity over the last few years. GA is a stochastic, heuristic technique based on the natural selection principle, and it does not need specific information to guide the search. Therefore, the GA can be applied in a wide variety of domains. A comprehensive discussion on GAs can be found in Holland (1975), Goldberg (1989) and Bäck *et al.* (1995).

Since the mid-eighties of the last century, multi-objective genetic algorithms (MOGAs) have been greatly developed to solve multi-objective scientific and engineering problems (Coello 1999, van Veldhuizen 2000). Till to date, most of the successful genetic type implementations for multiobjective optimization rely on the concept of nondomination firstly proposed by Pareto, and among them are Niched-Pareto-GA(NSGA) (Srinivas and Deb 1994) and its new version NSGA-II (Deb *et al.* 2000).

As a "Black-box" search algorithm, MOGA can deal with the multiple optimization

problems with non-smooth and even non-continuous objective or constraint functions. Furthermore, being a population-based technique, it makes possible to lead to the efficient front in a same evolution run.

In this paper, we will propose a model for portfolio selection with fixed transaction costs and minimum lots(ML). A NSGA-II based heuristic is designed to solve these two models. The paper is organized as follows. In Section 2, we formulate the models. A genetic algorithm is designed in Section 3. Computational results are given in Section 4. Finally, a few concluding remarks are made in Section 5.

2. Model

The portfolio selection model with fixed transaction costs and minimum lots based on a different risk measure was firstly introduced by Kelererer *et al.* (1997). In this section, we will present a new portfolio selection model with fixed transaction costs and minimum lots in a traditional mean-variance framework.

First we introduce some notations. We denote by S the set of securities in which the investor intends to invest a capital C, and suppose that C is fixed to range between range C_0 and C_1, i.e. between the minimum and the maximum amount of money available for the investment. Let $s = |S|$ be the number of the securities. Let r_i be the random rate of return on security i, where $i \in S$. Suppose that all the securities in S must be acquired in multiplies of a minimum transaction lot. Let N_j be the minimum transaction lot of security j. The quantity d_j represents the proportional cost associated with security j, and p_j represents the quoted price of security j at the time of its acquisition on the market. We denote by c_j the purchasing price for the minimum lot of security j. For each security, the minimum lot is expressed in terms of money and is equivalent to $c_j = N_j \cdot p_j$. Thus, a portfolio can be represented as $k = (k_1, ..., k_s)$, where k_j represents the multiple minimum lots of security j in the portfolio. We denote $R_i = E(r_i)$ the expected rate of return on security i, and $\sigma_{ij} = \mathrm{cov}(r_i, r_j)$ the covariance between r_i and r_j. For security $j \in S$, we impose a maximum amount of capital which can be invested in it, denoted by u_j. Consider the case in which the investor is obliged to pay the corresponding fixed cost f_j when security j is chosen in a portfolio, and the fixed costs are supposed to be incurred at the end of the period and are deduced from the portfolio return. Hence, the net expected return $R(x)$ of a portfolio k can be written as

$$R(x) = (\sum_{j \in S} (R_j - d_j) c_j k_j - \sum_{j \in S} f_j z_j) / C,$$

where $C = \sum_{j \in S} c_j x_j$ is the total sum of capital invested in the portfolio and z_j is a $0-1$ variable taking value 1 if and only if $k_j > 0$. The variance $V(x)$ is

$$V(x) = \sum_{i \in S} \sum_{j \in S} \sigma_{ij} y_i y_j$$

where $y_j = (k_j c_j) / C$ is the fraction of the capital C invested in security j.

The portfolio selection model with fixed transaction costs and minimum lots proposed in this paper is a two objective programming problem. The problem can be stated as follows:

(P1)

$$\begin{aligned} &\text{minimize } (\text{-}R(k), V(k)) \\ &\text{subject to} \\ &C_0 \le \sum_{j \in S} c_j k_j = C \le C_1 \qquad (1.1) \\ &0 \le c_j k_j \le u_j \qquad (1.2) \\ &z_j \in \{0,1\} \\ &\text{for all } j \in S \end{aligned}$$

A portfolio $k = (k_1, ..., k_s)$ is said to be feasible to (P1) if it satisfies all the constraints of (P1). A feasible portfolio k^* is said to be efficient if it is Pareto-optimal solution to (P1), i.e., there exists no other feasible portfolio k such that $R(k) \ge R(k^*)$, $V(k) \le V(k^*)$ with at least one strict inequality. The set of all the efficient portfolios is called the efficient frontier of (P1). In GA community, the solutions are traditionally called as chromosomes.

The fixed transaction costs may take on different forms. One model is proposed in Kellerer *et al.* (1997) where the fixed cost is charged if the sum of money invested in the individual security exceeds a bound, represented by the parameter M1, and otherwise no such cost is imposed. Another example is of more reality because at the Japanese stock market where different amount of fixed transaction cost is imposed according to the total amount of investment capital (Yoshimoto 1996). Mathematically the fixed transaction cost function can be plotted as a step function with two or more constant values.

Some attentions paid to the portfolio selection models with fixed transaction costs

and minimum lots. When the fixed transaction cost is of the type as a step function with two values, the problem (P1) can be transformed into a mixed-integer programming problem by adding an additional constraint and then the linear programming based heuristics MILP (Mansini and Speranza 1999) can be used to solve the problem. But for a step function which takes more than two values, this approach will encounter difficulties. In Yoshimoto (1996), although the multi-value step function typed fixed cost was discussed, the classical V-shaped transaction cost function was considered for calculations instead. Kanno and Wijayanayake (2001) considered the portfolio optimization problem with convex transaction cost function and minimum lots. A branch and bound algorithm was used to calculate the optimal solution (not integers in general). The solution is rounded to the nearest solution to satisfy the minimum lots constraint. In all the above methods, the risk and return measure functions are chosen in linear forms so that some linear programming algorithms can be applied. But for our model (P1) with quadratic risk terms, those methods do not work.

There are two types of constraints in (P1): the constraint (1.1) of the amount of invested capital and the bound constraint (1.2). Constraint (1.1) implies that a fund will be nearly fully invested to insure fully investment, like the constraint $\sum x_i = 1$ in classical M-V models, while (1.2) implies that short sales and borrowings are not allowed and a maximum amount of capital is imposed on each security.

Because finding a feasible solution tothe model (P1) is NP-hard (Mansini and Speranza 1999), it is significant to design some heuristic algorithms to solve (P1).

3. Genetic algorithms

GAs have been used for solving portfolio selection problems. Chang *et al.* (2000) extended the classical M-V model to include cardinality constraints and useed GA as one heuristic algorithm to find the efficient frontier. However, neither transaction costs nor minimum lots is considered. In Xia *et al.* (2000), GA is used to solve an extended M-V model with ordered expected returns. A V-shaped transaction cost function is considered there.

In Vedarajan *et al.* (2000), NSGA is used to calculate the efficient frontier of a simple portfolio that consists of five stocks. The efficient frontier can be yielded in just a run of the GA simulation, and the result seems very promising. A V-shaped transaction cost function is considered, but no minimum lots constraints have been taken into account..

In our previous work (Lin *et al.* 2001), we use a GA to solve the portfolio selection

and a similar portfolio revision model. Through scalar weighting, these two objective functions are combined into one scalar function. The proposed GA is quite similar to the GA which will be described in 3.1.

NSGA was originally proposed in Srinivas and Deb (1994), and recently Deb *et al.* (2000) proposed an improved version of NSGA called NSGA-II. From the simulation results of the NSGA-II on a number of difficult test problems, quite good performance has been observed. For the details of an NSGA-II algorithm, one can refer to Deb *et al.* (2000).

3.1 GA based on NSGA-II

In this section, we will design a genetic algorithm based on NSGA-II to solve problems (P1). Because (P1) includes constraints of discrete variables, some components of NSGA-II should be revised and a new mutation operator will be used. Since a detailed description of the GA will occupy too much space, in the following we will only discuss the representation structure, initialization process, cross and mutation operators, which are different from those of NSGA-II.

Representation structure: When using NSGA, one is allowed to choose binary or real-coded representation. The binary representation traditionally used in GAs has some drawbacks when it is applied to the integer programming problems ((P1) and (P2). For example, if for some security i the k_i is within $[0,100]$, then it is difficult to use a fixed length binary chromosome to exactly represent all the integers lying within $[0,100]$. A binary chromosome with length of 6 can represent all the integers in [0,63], while a binary string of length 7 represents the integers in [0,127]. Even if one could manage to deal with that, treating constraints in (P1) and (P2) would be a tough task. Adopting the idea of real-coded representation used in GAs for mathematical programming problems with real variables (Baeck *et al*. 1995), we will represent a solution by the integer implementation in which each chromosome is coded as a string of integer numbers

$$k = (k_1, ..., k_s),$$

and each x lies in the range $[0, upper(i)]$, where $upper(i) = [u_i / c_i]$ for each $i \in S$. Here $[\cdot]$ denotes the truncation function. This promises that the boundary constraint (1.2) holds for every $i \in S$.

Initialization process: As will be remarked later, in NSGA-II, there is no need to treat the constraint (1.1) separately. Thus attention will be paid to keeping the solutions from the initialization and throughout the whole simulation process to satisfy the boundary

constraint (1.2). For doing that, we generate random integer $k_i \in [0, upper(i)]$ for each $i \in S$, and then an initialized solution $k = (k_1, ..., k_s)$ is generated. Repeat it for pop_size times, where the parameter pop_size is the number of the population of chromosomes. Then the initialized population is generated.

Crossover operator: In NSGA-II, SBX crossover (Deb and Agrawal 1995) operator is used for real-coded representation chromosomes. In order to make it suit for our integer-coded representation chromosomes, we suggest a tiny revision to it as follows. After each time the crossover process happens, every variable x_i (not an integer value in general) of the children solution will be made to take an integer value k_i according to the following form

$$k_i = \begin{cases} [x_i] \text{ or } [x_i + 1] \text{ randomly}, & \text{if } [x_i] + 1 \le upper(i) \\ upper(i), & \text{else} \end{cases} \tag{3.1}$$

Here again, k_i will be in the range $[0, upper(i)]$.

Mutation operator: The Parameter Based Mutation (PM) Operator used in NSGA-II can be modified through truncating to suit for our integer-coded representation chromosomes in the similar way. It works as follows: for a parent chromosome $k = (k_1, ..., k_s)$, if for some $i, 1 \le i \le s, k$ has been chosen to be mutated, it can be mutated into x by using the traditional PM operator for real coded value. Then x is truncated to a integer k_i' by (3.1).

The main loop and other key issues of the GA are the same as that of NSGA-II.

There are two features of NSGA-II and the proposed GA. For the constraint (1.1), any possible solution k to (P1) which satisfies (1.2), the measure of the constraint violation $g(k)$ with regard to constraint (1.2) defined as

$$g(x) = \begin{cases} 0, & \text{if } C0 \le C \le C1 \\ C0 - C, & \text{if } C < C0 \\ C\text{-}C1, & \text{if } C > C1. \end{cases} \tag{3.2}$$

Obviously, a solution is feasible if an only if $g(x) = 0$. By guaranteeing that (1): a feasible solution is always better than an infeasible solution and (2): an infeasible solution with a small $g(x)$ is always superior to an infeasible solution with a large $g(x)$, an tournament selection method and elitism strategy (Deb *et al*. 2000) implemented in NSGA-II is able to find and keep feasible solutions in the population quickly and efficiently.

Besides it should be mentioned that since the GA does not utilize any particular properties of the objective functions such as linearity, convexity or differentiability, the fixed transaction cost with multi-value step function does not add any difficulty to calculation of the objective values.

For simplicity, we will call the GA proposed above GA1.

3.2 GA based on NSGA-II and Genocop

When using GA1 to solve (P1), there still exists a drawback for its genetic operators. As the constraint (1.1) is hard to satisfy, the genetic operators can not guarantee the feasibility of any children even the parents all feasible. So many infeasible individuals are generated in the evolution and quite a lot of computation effort has been wasted. In the preliminary work of this paper, the investigation shows that preserving the feasibility of individuals is of great importance to improve the efficiency of GA (Lin *et al.* 2001).

The Genocop (for GEnetic algorithm for Numerical Optimization of COnstrained Problems) system (Michalwicz ,1998) assumes linear constraints only and a feasible starting point (or feasible initial population). A closed set of operators maintains feasibility of solutions. For example, when a particular component x_i of a solution vector $X = (x_1, ..., x_n)$ is mutated, the system determines its current domain dom(x) (which is a function of linear constraints and remaining values of the solution vector X) and the new value of x is taken from this domain by using some mutation operator. The offspring solution vector is always feasible. Genocop 4.0, the fourth version of Genocop system, handles also integer and Boolean variables. Uniform, boundary, non-uniform and whole non-uniform mutation operators, as well as whole arithmetical, simple arithmetical and heuristic crossover operators are used.

As in (P1) there are only linear constraints, Genocop 4.0 will also work here. But it deserves more consideration about the constraint (1.1). Although it takes on a form of an inequality, it is essentially an equality constraint to ensure the total investment. For a feasible individual $k = (k_1, k_2, ..., k_s)$ which satisfies constraint (1.1), when, say k_1 is chosen to be mutated when using uniform mutation operator, k_1 can uniformly takes an integer value from the range

$$[\frac{C_0 - \sum_{i=2}^{s} k_i p_i}{p_1}, \frac{C_1 - \sum_{i=2}^{s} k_i p_i}{p_1}] \cap [0, upper(1)]$$

When the constraint is tight, i.e., $(C_1 - C_0)$ is small, k_1 can hardly been mutated into different values, especially in the case of large minimum lots. This eventually undermines the search power of mutation operators dramatically.

Based on the above discussion, now we are able to present a GA which combines the ideas of GA1 and Genocop and can preserve the feasibility of all solutions generated within the whole evolution generations. We call the new GA GA2 .

The main specific components of GA2 different from GA1 can be described as below:

Initialization process: When implementing Genocop, it requires that all individuals in the initial population are feasible. However, as mentioned above, finding feasible solutions to (P1) under the constraints is a NP-hard problem. As a good heuristic to solve such a constrain satisfaction problem, GA can be used to find the feasible individuals.

All feasible individuals to (P1) are solutions to the following single objective problem:

$$\begin{aligned} & \text{minimize } \mathrm{g}^2(k) \\ & \text{subject to} \\ \text{(P2)} \quad & 0 \le c_j k_j \le u_j \\ & \text{for all } j \in S . \end{aligned}$$

Viewed as a special case of the multi-objective optimization problems, (P2) can also be solved by using GA1. The run will be terminated until all individuals in the population are all feasible and this population will serve as the initial population for GA2.

Fitness scaling: First we will solve the following two single objective programming problems

$$\begin{aligned} \text{(P3)} \quad & \text{maximize} \quad \mathrm{V}(k) \\ & \textit{s.t.} \text{ constraints in (P1)} \end{aligned}$$

and

$$\begin{aligned} \text{(P4)} \quad & \text{maximize } \mathrm{R}(k) \\ & \textit{s.t.} \text{ constraints in (P1)} \end{aligned}$$

The solutions to (P3) and (P4) are of global minimum variance and maximum return respectively, and their fitness value are denoted as $(R_{\min}, V_{\min})$ and $(R_{\max}, V_{\max})$. These two solutions will be inserted into the initial population by substituting two randomly selected individuals.

Now the fitness function in (P1) will be transformed by scaling and we obtain a new model (P5)

(P5) minimize $\left(\frac{-R(k)}{R_{mzx}-R_{\min}}, \quad \frac{V(k)}{V_{\max}-V_{\min}}\right)$

s.t. constraints in (P1).

The purpose of scaling is to eliminate the "range dependent" effect of the rank method (Bently and Wakefield, 1997). It will not affect the result of dominance comparison of any two individuals at all, while at the same time it does change the value of the quantity for density estimation and eventually cause the different rank of the individuals in the same efficient front (Deb *et. al.* , 2001).

Constraint handling: In GA2, only the boundary constraint (1.2) need to be considered. The genetic operators will always keep the individuals satisfying (1.2).

Mutation operators: Instead, we could treat (1.1) as an equality and then extract an variable as one wish as a non free variable. Without loss of generality, in the following we will assume that k_1 has been extracted. For the sub-individual $k^{'}=(k_2,...,k_s)$, when k_j is chosen to be mutated into $k_j^{'}$ which belongs to its current domain dom(k_j), then we make $k_j^{'}$ must satisfy the follow inequalities in addition to the boundary constraint (1.2):

$$C_0 - p_1 \cdot upper(1) - \sum_{\substack{i=2 \\ i\neq j}}^{s} p_i k_i \le k_j^{'} \cdot p_j \le C_1 - \sum_{\substack{i=2 \\ i\neq j}}^{s} p_i k_i \ . \tag{3.3}$$

Thus the inequality

$$C_0 - p_1 \cdot upper(1) \le \sum_{\substack{i=2 \\ i\neq j}}^{s} p_i k_i + k_j^{'} p_j \le C_1 \tag{3.4}$$

holds, and we have

$$C_0 - \sum_{\substack{i=2 \\ i\neq j}}^{s} p_i k_i - k_j^{'} p_j \le p_1 k_1 \le C_1 - \sum_{\substack{i=2 \\ i\neq j}}^{s} p_i k_i - k_j^{'} p_j \tag{3.5}$$

From (1.4) it is easy to see that

$$C_0 - \sum_{\substack{i=2 \\ i\neq j}}^{s} p_i k_i - k_j^{'} \cdot p_j \le p_1 \cdot upper(1)$$

and

$$0 \le C_1 - \sum_{\substack{i=2 \\ i\neq j}}^{s} p_i k_i - k_j^{'} p_j \ ,$$

so at least one integer $k_1^{'} \in [0, upper(1)]$ exists, and the new individual $k^{''} = (k_1^{'}, k_2, ..., k_j^{'}, ...k_s)$ is a feasible solution to (P1).

When implementing boundary, uniform and non-uniform operators, only a single variable of an individual is mutated and the mutation is implemented according to the process just described. For whole non-uniform operator which mutates all the variables of the individual, we need to adjust the current value of k_1 just after each mutation happened variable by variable.

Crossover operators: The implementation of all three crossover operators is just the same as Geconop 4.0. When two individuals are chosen to crossover using whole arithmetical crossover operators, they produce children individuals as in the real-code situation, then each variable of the children is set to an integer by (3.1) and then two integer valued children are created. The feasibility of these two children to (P1) are tested, and they will be accepted if they are both feasible. If not, the two parent will crossover once more. This process is repeated until two feasible individuals have been created or the number of trials proceeds a prespecified maximum trial number. Similar implementations are done to other crossover operators.

We can now summarize the GA based on the SBX and PM genetic operators which solves the portfolio optimization problems (P2) —(P4) as follows:

Step 0. *Input all parameters such as* pop_size, *crossover probability* p_c *and mutation probability* p_m *of all genetic operators, total evolutionary generations* gen.

Step 1. *Use GA1 to solve (p2) to generate the initial population consists of all feasible solutions to (P1).*

Step 2. *Use GA1 to solve (P3) and (P4) to obtain the global minimum risk and the maximum return solution, insert them into the initial population. Do fitness function scaling.*

Step 3. *Update the individuals in the current population by crossover and mutation operators.*

Step 4. *Calculate the scaled fitness values for all individuals.*

Step 5. *Select the individuals by using the binary tournament selection strategy, carry out elitism strategy to generate the next generation for evolution.*

Step 6. *Repeat steps 3 to 5 until the given* gen *generations.*

Step 7. R*eport the efficient solutions in the final population.*

4. Simulation results

4.1. Problem description and GA settings

In this section, the proposed GA2 is illustrated by solving (P1) using the data publically available from OR-Library at http://mscmga.ms.ic.ac.uk/jeb/orlib/portinfo.html. The data consist of 31 observations of weekly prices for the period from March 1992 to September 1997 for asset from Hang Seng index. The top six mean returns of these stocks are 0.0109, 0.0071, 0.0058, 0.0053, 0.0052, 0.0050 respectively. The prices of all stocks in the last period are randomly generated in the range [10,100].

The minimum lots ML is set to be 100 or 1 when purchasing and selling stocks. In constraint (1.1), $C_0 = 1{,}000{,}000$ and $C_1 = 1{,}00{,}5000$. For each security $i \in S$, $u_i = 0.4$ or 1. Denote by C the amount to be invested in stock i, the proportional transaction costs $P(C)$ and fixed transaction costs $F(C)$ are defined as

$$\begin{cases} P(C) = 0.0115C, F(C) = 0 \\ \quad \text{if } C < 10{,}000 \\ P(C) = 0.009C, F(C) = 25 \\ \quad \text{if } 10{,}000 \le C < 50{,}000 \\ P(C) = 0.007C, F(C) = 125 \\ \quad \text{if } 50{,}000 \le C < 100{,}000 \\ P(C) = 0.00075C, F(C) = 785 \\ \quad \text{if C} \ge 100{,}000 \end{cases}$$

These settings are rather arbitrary for our purpose of this section is merely to test the efficiency of the proposed GA for model (P1).

To solve (P1) when using GA1 in step 1 and 2, we let the probability of crossover P_c be 0.95 and the probability of mutation P_m be 1/(chromosome length) as proposed in NSGA-II. $pop\_size = 200$ in both steps, and $gen = 100$ and 1500 in step 1 and 2 respectively. For step 3 to 5, $pop\_size = 200$ and $gen = 3000$. $P_C = 0.4$ for all crossover operators and $P_m = 0.2$ for all mutation operators.

4.2. Computational results

In all simulations, GA1 can always attain a feasible population within 50 generations. The NP-hard problem for finding feasible solutions has not been cause difficulties to GA.

The solid lines in all the following figures are the efficient solutions for the classical Markowitz M-V model in the return-risk plane. These efficient frontier values are provide in the data set we used. In some sense, they can serve as benchmark.

The efficient frontier obtained when using GA2 for (P1) with $ML = 1$ and $u_i = 1$

and without transaction costs is shown in Figure 1. As $ML = 1$ is the smallest transaction lots as possible while u_i takes the largest possible value 1, in this case (P1) is the most accurate approximation of the M-V model. In Figure 1 we can find that the efficient solutions obtained almost exactly lies in the efficient frontier for M-V model. This fact proves the efficiency of GA2.

We also solve (P1) without changing any settings except for omitting fitness scaling implemented in step 2. Figure 2 shows that the distribution of the obtained solutions composing the efficient frontier is less even than that in Figure 1 for the part when the e return of the solutions is relatively high. Similar phenomena occur in other simulations of which the results are not presented here. We can draw a conclusion that the efficiency of GA2 always been undermined without fitness scaling.

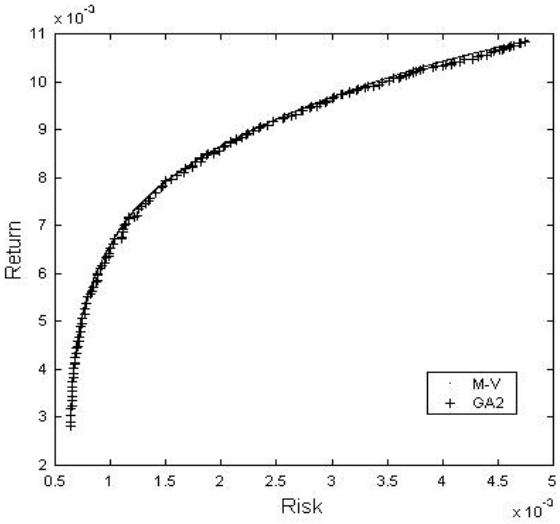

Fig1. GA2 for (P1) with fitness scaling, $ML = 1, u_i = 1$, without transaction without transaction costs

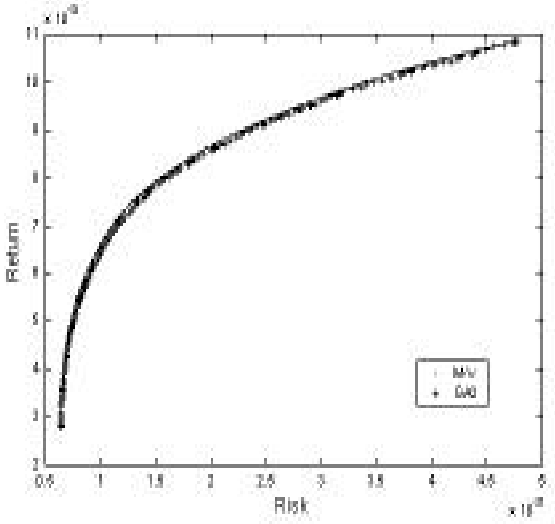

Fig2. GA2 for (P1) without fitness scaling, $ML = 1$, $u_i = 1$, without transaction costs

The results for $u_i = 0.4$ and $ML = 1$ are shown in Figure 3. We can see that by investing in more stocks, the maximum risk has decreased significantly. Once more the efficiency of GA2 can be proven from the coincidence of these two frontiers.

In Figure 4, the obtained frontiers with $u_i = 0.4$ and $ML = 1$ and with and without transaction costs are plotted for comparison. Although we lack benchmark to test the quantity of the solutions where transaction costs are considered, we have faith in it.

Figure 5 and 6 show the results corresponding to those in Figures 3 and 4 while $u_i = 0.4$ and $ML = 100$, and the results are quite similar.

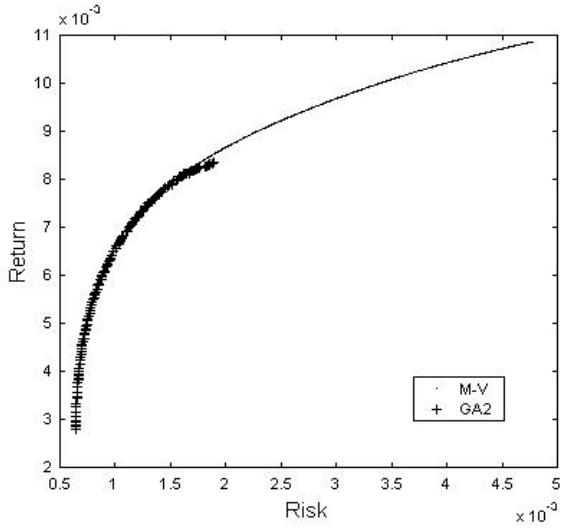

Fig3. GA2 for (P1) with $ML = 1$, $u_i = 0.4$, without costs

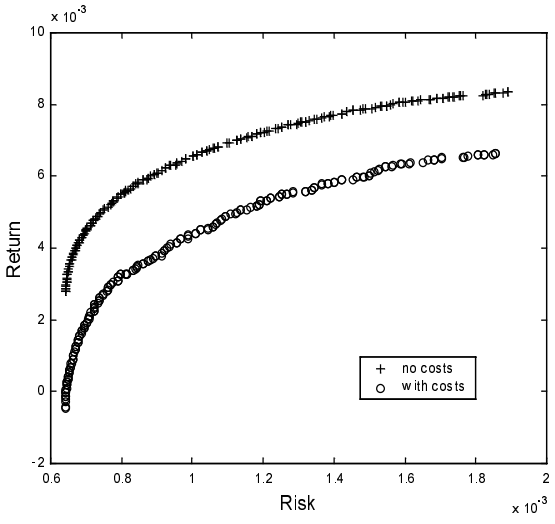

Fig4. GA2 for (P1) with $ML = 1$, $u_i = 0.4$, with and without costs

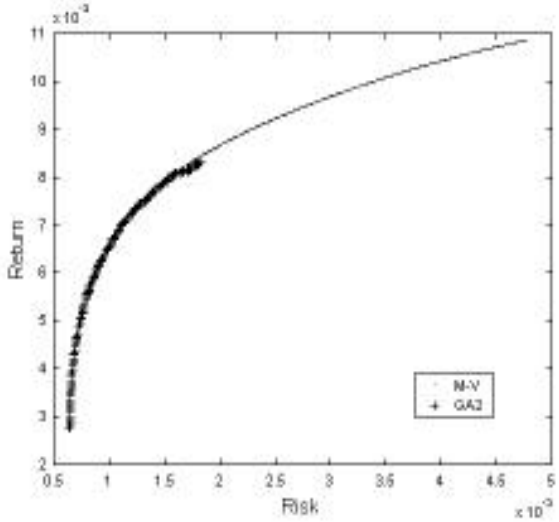

Fig5. GA2 for (P1) with $ML = 100$, $u_i = 0.4$, without costs

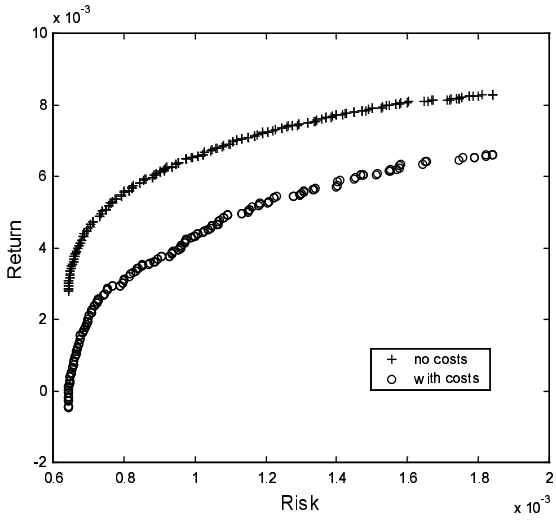

Fig6. GA2 for (P1) with $ML = 100$, $u_i = 0.4$, with and without costs

It is worthwhile to highlight that as GAs are highly parallel algorithms in nature (Goldberg 1989), it is easy to implement GAs on parallel architectures and the efficiency can be improved drastically.

5. Conclusions

In this paper, we have proposed a new mean-variance model for portfolio selection with both fixed transaction costs and minimum lots. As the model is a nonlinear, multiple objectives integer programming problem, we design a GA based on NSGA-II and Geconop to solve the programming problems. Computational results show that the proposed GA is a promising tool to solve portfolio optimization problems.

Acknowledgements

The authors would like to thank Prof. Kalyanmoy Deb of Indian Institute Of Technology Kanpur and Prof. Zbigniew Michalewicz of University of North Carolina for making their genetic algorithm code publicly available.

References

Baeck, T., Fogel, D.B., Michalewicz, Z., (Eds.), 1997. Handbook of Evolutionary Computation. Oxford University Press, Oxford.

Bentley, P. J., Wakefield, J. P., 1997. Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms. Chawdhry, in: Roy, P.K., Pant, R.K. (Eds.), Soft Computing in Engineering Design and Manufacturing. Springer Verlag London Limited, Part 5, 231-240.

Chang, T.J., Meade, N., Beasley, J.E., Sharaiha, Y.M., 2000. Heuristics for cardinality constrained portfolio optimization, Computers and Operations Research 27,1271-1302.

Chen, A.H.Y., Jen, F.C., Zinots, S., 1971. The optimal portfolio revision policy, Journal of Business 44,51-61.

Coello Coello, C.A., 1999. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. Knowledge and Information Systems 1, 269-308.

Dantzig, G.B., Infanger, G., 1993. Multi-stage stochastic linear programs for portfolio optimization. Annals of Operations Research 45, 59-76.

Deb, K., Agrawal, R.B., 1995. Simulated binary crossover for continuous search space. Complex Systems 9,115-148.

Deb, K., Agrawal S., Pratab A., Meyarivan, T., 2000. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India.

Fama, E., 1965. Portfolio analysis in a stable market. Management Science 11, 404-419.

Gennotte, G., Jung, A., 1994. Investment strategies under transaction costs: the finite horizon case. Management Science 40, 385-404.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, New York, NY.

Holland, J.H.,1975. Adaptation in natural and artificial systems. University of Michigan Press, Michigan.

Jacob, N.L., 1974. A limited-diversification portfolio selection model for the small investor. Journal of Finance 29,847-856.

Kellerer, H., Mansini, R., Speranza M.G., 1997. On selecting a portfolio with fixed costs and

minimum lots. Working paper at Dip. di Metodi Quantitativi, Universita di Brescia, C.da.S Chiara 48/b, 25122 Brescia, Italy.

Konno, H., Wijayanayake, A., 2001. Portfolio optimization problems under concave transaction costs and minimal transaction unit constraints. Mathematical Programming 89, 233-250.

Levy, H., 1978. Equilibrium in an imperfect market: a constraint on the number of securities in the portfolio. American Economic Review 68,643-658.

Li, Z.F., Wang, S.Y., Deng, X., 2000. A linear programming algorithm for optimal portfolio selection with transaction costs. International Journal of Systems Science 31,107-117.

Lin, D., Wang, S.Y., 2001, A Genetic algorithms to solve portfolio optimization problems with transaction costs and minimum transaction lots, to be published in Proceeding of Knowledge Sciences and System Sciences 2001.

Lin, D., Wang, S.Y., Hong, Y., 2001,Genetic algorithms to solve portfolio optimization problems with transaction costs and minimum transaction lots, submitted to Computers and Mathematics with Applications.

Mao, J.C.T.,1970. Essentials of portfolio diversification strategy. Journal of Finance 25: 1109-1121.

Markowitz, H., 1952. Portfolio selection. Journal of Finance 7, 77-91.

Masini, R., Speranza, M.G., 1999, Heuristic Algorithms for a Portfolio Selection Problem with Minimum Transaction Lots, European Journal of Operational Research 114(2), 219-233.

Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs, 3rd edition, Springer-Verlag, Berlin.

Mulvey, J.M., Vladimirou, H., 1992. Stochastic network programming for financial planning problems, Management Science 38, 1642-1664.

Patel, N. R., Subrahmanyam, M. G., A simple algorithm for optimal portfolio selection with fixed transaction costs, Management Science, 1982; 28:303-314.

Pogue, G.A, 1970. An extension of the Markowitz portfolio selection model to include variable transaction costs, short sales, leverage policies and taxes. Journal of Finance, 25, 1005-1028.

Speranza, M. G., 1996. A heuristic algorithm for a portfolio optimization model applied to the Milan stock market. Computers & Operations Research 23,433-441.

Srinivas, N., Deb, K., 1995. Multi-objective function optimization using non-dominated sorting genetic algorithms. Evolutionary Computation 2, 221-248.

van Veldhuizen, D.A., Lamont, G.B., 2000. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. Evolutionary Computation 8, 125-147.

Vedarajan, G., Chan, L.C., Goldberg, D., 2000. Investment portfolio optimization using genetic algorithms, in: Gonnet, G.H., Panario, D., Viola, A. (Eds.), Lecture Notes in Computer Science, vol. 1776. Springer.

Wang, S.Y., Xia, Y.S., 2000. Portfolio Optimization and Asset Pricing. Global-Link Publishers, Hong Kong.

Xia, Y., Wang, S.Y., 2000. A model of portfolio selection with order of expected returns. Computers & Operations Research 27, 409-422.

Yoshimoto, A., 1996. The mean-variance approach to portfolio optimization subject to transaction costs. Journal of Operations Research Society of Japan 39,99-117.