Portfolio Optimization using Multi-objective Genetic Algorithms

Prisadarng Skolpadungket[1], Keshav Dahal[1] and Napat Harnpornchai[2]
[1]School of Informatics, University of Bradford BD7 1DP, UK
[2]College of Arts, Media and Technology, Chiang Mai University, Thailand
p.skolpadungket; k.p.dahal{@bradford.ac.uk}； tomnapat@camt.info, tomnapat@gmail.com

***Abstract*—A portfolio optimisation problem involves allocation of investment to a number of different assets to maximize yield and minimize risk in a given investment period. The selected assets in a portfolio not only collectively contribute to its yield but also interactively define its risk as usually measured by a portfolio variance. In this paper we apply various techniques of multiobjective genetic algorithms to solve portfolio optimization with some realistic constraints, namely cardinality constraints, floor constraints and round-lot constraints. The algorithms experimented in this paper are Vector Evaluated Genetic Algorithm (VEGA), Fuzzy VEGA, Multiobjective Optimization Genetic Algorithm (MOGA) , Strength Pareto Evolutionary Algorithm 2nd version (SPEA2) and Non-Dominated Sorting Genetic Algorithm 2nd version (NSGA2). The results show that using fuzzy logic to combine optimization objectives of VEGA (in VEGA_Fuz1) for this problem does improve performances measured by Generation Distance (GD) defined by average distances of the last generation of population to the nearest members of the true Pareto front but its solutions tend to cluster around a few points. MOGA and SPEA2 use some diversification algorithms and they perform better in terms of finding diverse solutions around Pareto front. SPEA2 performs the best even for comparatively small number of generations. NSGA2 performs closed to that of SPEA2 in GD but poor in distribution.**

***Index Terms*— Portfolio optimisation, Investment management, Multiobjective Genetic Algorithms, Vector Evaluated Genetic Algorithm, VEGA, Strength Pareto Evolutionary Algorithm, SPEA2, Fuzzy VEGA.**

I. INTRODUCTION

Different exact techniques have been used to solve the portfolio optimisation problems. These techniques usually involve the exploration of the large number of combinations of states which increases exponentially with the size of problem becoming computationally intractable [1]. Furthermore, many of these techniques are inept in handling the nonlinear objective and constraint functions and several assumptions are generally required to make the problem solvable using reasonable computational resources [2]. Alternatively, some heuristic-based and evolutionary techniques can approximate solutions for problem instances of NP-hard problems in a reasonable time [3]. Those techniques can tackle the optimisation problems in polynomial time with a traded-off of their optimality. In some circumstances of the real world problems, the speed to reach acceptable approximate solutions is very critical. Feasible near-optimum solutions are acceptable but untimely are not. Simple heuristics, based on greedy search algorithms, tend to stop in inferior local optima.

Genetic Algorithms (GA) are population based heuristic algorithms. In GA, solutions are represented as chromosomes that to be breed by crossover or modified by mutation. Selection processes are used to find optimal or near-optimal solutions imitating the natural selection of survival of the fittest [2]. Busetti [4] compared GA with tabu search and found that GA performs better for portfolio optimization problems for the problem setting considered. Streichart et al. [5] applied the Multi-Objective Evolutionary Algorithm (MOEA) to solve portfolio optimization problem. Earlier Tettamanzi et al. (cited in [6]-[8]) transformed the multi objective optimization problem into a single-objective problem by using a trade-off function (therefore not a true multi-objective). Mukherjee et al. [9] also solved the risk-return trade off problem that similar to portfolio optimisation problem by MOEA. The paper compared performance of different GA representations of portfolio optimization with several combinations of real world constraints on the Hang Seng data set with 31 assets. The representations are binary bit-string based genotypes or gray-code encoding and real-valued genotype. The constraints imposed on the optimisation problems are cardinality and integer (discrete) constraints. In another paper Streichert et al. [10] introduced an alternative hybrid encoding for evolutionary algorithms, which combines both 'continuous' real value and 'discrete' binary value together. The algorithm then compared with the different EA representations. When the algorithm and the other EAs without Larmarckism (the genetic encoding can be adapted and changed not only by mating and mutations but also during evaluations). was applied on the problem with only cardinality constraints, the algorithms performed better than those of standard EAs. Subbu et al. [11] presented a new hybrid evolutionary multi-objective portfolio optimisation problem algorithm called Pareto Sorting Evolutionary Algorithm (PSEA) that integrates evolutionary computation with linear programming. However, the aim of the paper was only to design algorithm and architecture for portfolio optimization not to measure or compare performance of the algorithm.

In this paper we propose to investigate the performance of various multi-objective genetic algorithms to solve portfolio optimization problem with some realistic constraints. We also extend genetic algorithms for multi-objective optimization by applying fuzzy logic into their evaluation and selection which are tested along with other standard multi-objective genetic

1-4244-1340-0/07/$25.00 ©2007 IEEE

algorithms. The paper is organized as follow. The next section presents the structure of portfolio optimization problem. Section III describes the algorithms to be tested in this paper. Section V gives details of the problems and experiment setting. Section V is to report the results, and the last section concludes the paper.

II. THE PROBLEM

Modern portfolio theory originated in a paper by Harry M. Markowitz in 1952 [12]]. The theory stated that an investor should not select assets due to only characteristics that are particular to the assets but she/he need to consider how each asset co-moved with all other assets. Moreover, by taking into account of these co-movements, an investor can construct a portfolio that has less risk given the same expected yield than a portfolio constructed by ignoring the interaction between securities [13].

The Markowitz model is a well-defined optimisation model and with some modification later by Black [14]] to allow short-selling (allowing negative weights of assets) the model has a closed form solution. By removing some realistic assumptions such as the non-negativity constraints (i.e. no short sell on any assets are allowed); the integer constraints (i.e. shares of assets cannot be divided into lower than their trading units,) etc., the model has a general form which has only the assets' expected yields, the variance and covariance of the assets are parameters. On the other hand, if we impose the non-negativity constraint, there exists no general form (closed form) solution for the optimisation problem. Although the model with non-negativity constraint can be solved efficiently by specialised algorithms and other ad hoc methods, imposing other constraints (e.g. the integer constraint or maximum number of asset constraint) will cause large-scale problems became unable to be solved by mixed integer non-linear programming or other exact solution algorithms, within a reasonable time [4] . The portfolio optimisation problems with realistic constraints are NP hard problem especially for those of exact solutions. The methods require complete enumeration where all possible and valid values for the decision variables are tested.

The Markowitz model assumes that investors make their decision in portfolio construction by choosing assets that maximise their portfolio *yield*s at the end of investment period (expected *yield*s). By assuming that investors are risk averse, the simplest model with a number of unrealistic constraints namely, perfect market without taxes, no transaction costs, no short sales, assets are infinitely divisible, the Markowitz portfolio optimisation can be stated mathematically as follows:

$$Min_{X_i} \quad \sigma_P^2 \tag{1}$$

subject to

$$\sigma_P^2 = \sum_i \sum_j x_i x_j \sigma_{ij} \tag{2}$$

$$r_P = r^* \tag{3}$$

$$r_P = \sum_i x_i r_i \tag{4}$$

$$\sum_i x_i = 1 \tag{5}$$

$$x_i \in \Re_0^+ \quad \forall i \tag{6}$$

Where σ_{ij} is covariance between asset i and j, if i = j, it is variance of asset i.

σ^2_P is variance of the portfolio of assets.

r_i is expected yield of asset i

r_P is the expected yield of the portfolio

r^* is a predefined level of yield

These additional conditions must hold so that the optimisation has a solution:

$$\min_i r_i \leq r_P \leq \max_i r_i \tag{7}$$

$$\sigma_i > 0 \quad \forall i \tag{8}$$

$$\rho_{ij} > -1 \quad \forall (i, j) \tag{9}$$

$$\exists (i \neq j) \quad such \ that \quad r_i \neq r_j \tag{10}$$

Where σ_i is standard deviation (square root of variance) of asset i.

ρ_{ij} is correlation coefficient of asset i and asset j

r_j is expected yield of asset j

The Markowitz model is a simplified model to focus only a theoretical point of view. In the real world of investment management, portfolio managers face a number of realistic constraints those arise from normal business practices, practical matters and industry regulations. The realistic constraints that are of practical importance include (not exhaustively) round-lot constraints, cardinality constraints, floor constraints, turnover constraints, trading constraints, buy-in threshold and transaction cost inclusions. For this paper, we will concern only round-lot constraint, cardinality constraint and buy-in (floor) constraint.

Round-lot constraint make the number of any asset include in the portfolio must be multiples of normal trading lot. The round-lot constraints can be expressed as

$$x_i = \frac{n_i}{\sum_{i=1}^{N} n_i} \tag{11}$$

and

$$n_i \bmod l_i = 0 \quad \forall i \tag{12}$$

Where n_i is number of unit of asset (share) and l_i is trading lot of the asset i.

Cardinality constraints are the maximum number and minimum number of assets that a portfolio manager wishes to include in the portfolio due to monitoring reasons or diversification reasons or transaction cost control reasons [14]. The constraints can be expressed as follows

$$C_l \le \sum_{i=1}^{N} b_i \le C_u \qquad (13)$$

$$where b_i = 1 \ \ if \ x_i > 0 \ \ and \ b_i = 0, otherwise$$

where C_l and C_u are the lowest number of assets and the highest number of assets required to include in a portfolio respectively.

Floor constraint defines lower limits on the proportion of each asset, which can be hold in a portfolio. These constraints may result from institutional policy in order to diversify portfolio and to rule out negligible holding of assets for the ease of control [1]. They can be expressed mathematically as follows

$$f_i \le x_{ii} \qquad \forall\, i \qquad (14)$$

Where f_i is the lowest proportion and the highest proportion that asset i can be held in the portfolio.

In this paper, we are to solve multi-objective portfolio optimization problem by using genetic algorithm as follow:

Max Portfolio Expected Yield
Min Portfolio Variance of Yield
Max Sampling Distance (only VEGA_Fuz2)
Subject to:
$\Sigma x_i = 1$ and $x_i >= 0$ for all i;
Cardinality Constraint (max only) according to (13)
Floor Constraint according to (14)
Round-lot Constraint according to (11) and (12)

For all constraints handling, we use repair algorithms as described in section III A, so there are only two objectives for multi-objective genetic algorithms to optimize. Sampling Distance is adopted to be one of objective functions as a mechanism to preserve distribution along the Pareto front.

III. The Algorithms

Four GA based approaches are implemented to solve the portfolio optimization model discussed in the previous section.. We first experiment with Vector Evaluated Genetic Algorithm (VEGA) which is an extended version of single objective GA to handle multi-objective in a single run. Although VEGA is usually outperformed by other newer algorithms such as MOGA, SPEA and SPEA2, it is the best in term of complexity and easy to implement [17],[20]. We aim to improve VEGA by using Fuzzy Logic to enable joint evaluation of the both objectives as well as try to integrate a simple distribution preservation mechanism which does not increase complexity, as described in subsection C. We compare the results to MOGA which is wildly used multi-objective GA and SPEA2 which is recently considered among the best of multi-objective GAs for its performances.

A. Algorithms Design

Problem representation: The problem is represented by hybrid encoding [10],[15] A pair of genetic strings stands for a particular portfolio (an individual of population). The binary value string represents which stocks (or assets) are included in portfolio (0 stands for not included and 1 stands for included.) The real value string represents weights of each stock in portfolio. So, the lengths of both strings are equal to the number of stocks in the market (or the stocks of interest.) Before, repair process by the repair algorithm begins both strings combine by scalar product of the Binary string and the Real value string. Then after the repair process ended, the combined string separates into new and normalized Binary string and Real value string.

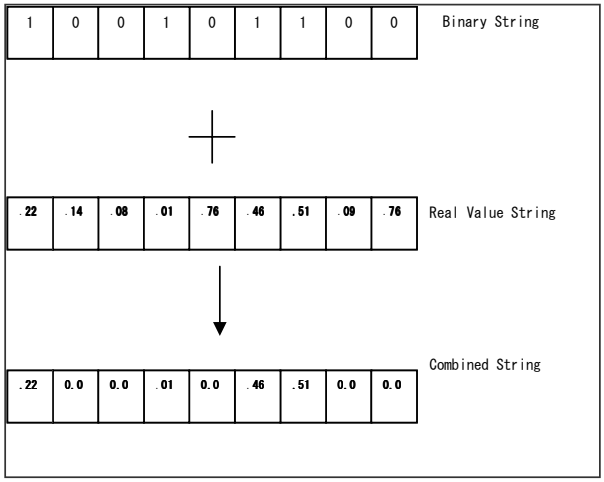

Fig 1: Problem Representation: Binary String, Real Value String and Combined String

GA operators: Crossover and mutation operations are performed independently for both strings. But before evaluation both strings need to be combined so that the objective values can be calculated. Crossover operation for all GAs is three-point crossover by randomly select three points for the string independently. Mutation operation for all algorithms in this paper is one-point mutation by randomly selecting the mutation point. For Binary strings, the mutation is flip-flop mutation by changing from 1 to 0 and 0 to 1 respectively. For Real value strings, the mutation point is added by random number (between 0 and 1) multiply by 0.05 (5% weight).

Repair Algorithms and Constraint Handling: All constraints are handled through a repair algorithm. The algorithms were proposed and used in [5],[10] and [15]. The constraints in this setting are unity constraint (the sum of weights must be equal to one), cardinality constraints, floor (buy-in) constraints and round-lot constraints.

The repair algorithm first handles the cardinality constraints by setting smaller (S-K) values (from S values) of combined string to zero, where S is the number of selectable stocks (equal to the length of the strings) and K is the maximum number of

stocks permitted in a portfolio (cardinality constraint.) Then, it handles floor constraint (buy-in threshold) by setting stocks whose weights below the buy-in threshold to zero. Next, it normalizes those remaining non-zero weights to make all weight sum to 1 by setting $w_i^{/} = l_i + (w_i - l_i) / \Sigma(w_i - l_i)$, where w_i is non-zero weight of stock i and l_i is the buy-in threshold (the minimum weight amount that can be purchase) for stock i. Then, the round-lot constraints are handled by rounding the non-zero weights to the next round-lot level such that $w_i^{//} = w_i^{/} - (w_i^{/} \bmod c_i)$, where, c_i is the smallest volumes can be normally purchased from the stock market for stock i. The remainder of the rounding process ($\Sigma\, w_i^{/} \bmod c_i$) is allocated in quantity of c_i to $w_i^{//}$ which has the biggest value of $w_i^{/} \bmod c_i$ until all of the remainder is depleted.

All pairs of strings first are filled with random number, so, they need to be repaired by the repair algorithm. And since crossover and mutation operations cause the string deformed, the repair algorithm need to be applied again to preserve the aforementioned constraints before the evaluations and selections.

B. Vector Evaluated Genetic Algorithm (VEGA)

The Vector Evaluated Genetic Algorithm is proposed by [16] as an extension of a simple genetic algorithm to handle multiple objectives in a single run. VEGA is a criterion-based fitness assignment, which is filling equal portions of mating pool according to the different objectives [17]. For m optimization objectives and a fixed population size as P, VEGA randomly selects m subpopulations with the size of P/m each so there are i = 1 to m subpopulation. Each individual in subpopulation i will be evaluated based on the optimization objective i. After probabilistic selections based on relative objective values (two for this experiment i.e. relative yield which is defined by yield of asset i / maximum yield of any asset in the population and relative variance which is defined by variance of asset i / minimum variance of any asset in the population), the selected individual from each subpopulation is shuffled and pooled together to form a new population of size P. The new population is then followed by crossover and mutation operations. The whole process is repeated until the predetermined condition is met (in this case, the total number of generations has been reached. VEGA is usually has O (n) complexity for each generation of population, where, n is the number of individuals in a population. VEGA main routine is exhibit in Figure 2.

```
Initialize generation counter: N = 0
Create a population, Pop.
Repeat while stopping criteria is not met.
     Initialize subpopulation counter: i = 1.
     Repeat while i <= m.
     Generate i th subpopulation, SubPop(i), by randomly
     selecting P/m individuals from Pop.
         Remove any individuals SubPop(i) from Pop.
     Generate the i th objective function values, F(i), for
     individuals   in SubPop(i).
          Perform genetic selection on SubPop(i) based on
F(i).
          i = i + 1.
      End Repeat
     Pop = Integrate all SubPop(i).
     Shuffle the individual sequence in Pop.
     N = N + 1.
End Repeat
Evaluate Pop for all objective function values for all F(i).
Return (Pop, all F(i),...)
```

Fig 2: VEGA Main Routine

C. Fuzzy VEGA

VEGA tends to converge towards one objective best solution, thus it is quite incompatible with multiobjective optimization in which we try to trade-off between objectives. Introducing fuzzy logic into VEGA may facilitate the traded-off between objectives. We incorporate a fuzzy decision rule to combine optimization objectives together. The fuzzy decision rule dictates the probability of selection for each individual. For the first Fuzzy VEGA (VEGA_Fuz1), Fuzzy decision rule combines two objective functions i.e. portfolio yield and portfolio variance of yield. The Fuzzy rule is exhibit in Table 1. VEGA_Fuz1 is then modified to incorporate distribution preservation mechanism by setting Sampling distribution (with 25 samples for all populations) as an additional objective function. We call this version as VEGA_Fuz2. Fuzzy logic with 3 objectives is quite cumbersome and unable to be displayed in a 2 dimension table like Table 1. The Fuzzy logic is composed of 6^3 or 216 rules (Sampling distribution is graded to max, very high, high, moderate, low and very low). The main loop of Fuzzy VEGA is exhibited in Figure 3 below.

```
INITIALIZE GENERATION COUNTER: N = 0
Create a population, Pop.
Repeat while stopping criteria is not met.
     Repeat while i <= m.
     Generate the i th objective function values, F(i), for
      individuals   in Pop.
    End Repeat
   Fuzzify of the objective function values.
    Apply Fuzzy rules to the Fuzzified objectives giving Fuzzy
value.
    Defuzzify the Fuzzy value to get selecting probability p.
          Perform genetic selection on Pop by probability p.
     N = N + 1.
End Repeat
Evaluate Pop for all objective function values for all F(i).
Return (Pop, all F(i)   )
```

Fig 3: Fuzzy VEGA Main Routine

TABLE I
FUZZY RULE FOR FUZZY VEGA (VEGA_FUZ1)

Y | Var	Min	Very Lo	Low	Moderate	High	Very High
Max	Certain	Highly Likely	Highly Likely	Likely	Likely	Probably
Very High	Highly Likely	Highly Likely	Likely	Likely	Probably	Probably
High	Highly Likely	Likely	Likely	Likely	Probably	Probably
Moderate	Likely	Likely	Likely	Probably	Unlikely	Highly Unlikely
Low	Likely	Probably	Probably	Unlikely	Unlikely	Highly Unlikely
Very Low	Probably	Probably	Probably	Unlikely	Highly Unlikely	Never

In Table 1, we derive Fuzzy decision rules based on joint optimality of the objective functions (yield and variance in VEGA_Fuz1, and yield, variance and sampling distance inn VEGA_Fuz 2). The most desirable is when there are maximum yield and minimum variance (and maximum sampling distance) . While least desirable one is when there are very low yield and very high variance (and very low sampling distance). The subjective desirable of combined objective values is then assigned probabilities of selection. The probabilities of selection are set to 7 levels accordingly. Certain, Highly Likely, Likely, Probably, Unlikely, Highly Unlikely and Never represent probabilities of selection of 1.0, 0.9, 0.75, 0.5, 0.35, 01 and 0.0 accordingly. The second version of fuzzy VEGA (VEGA_Fuz2) is different from VEGA_Fuz1 by combining an additional objective, sampling distance by randomly selecting 25 different individuals from populations (we use sampling instead of plain distance to reduce the complexity of the algorithm.) into fuzzy rule. The sampling distance here is used the same formula as Sharing Distance in MOGA as described in subsection D but does not calculate from all individuals, only 25 randomly selected from all individuals in the population. The fuzzy rule prefers more sampling distance to less to correct clustering problems of both VEGA and VEGA_Fuz1.

D. MOGA

MOGA employed in this paper is based on the algorithm proposed by Fonseca and Flemming in 1993 [18] . MOGA uses Pareto rankings to assign the smallest ranking value to all non-dominated individuals. For dominated individuals, they are ranked by how many individuals in the population dominate them. Thus, the raw fitness of an individual is an inverse function of its Pareto rank. In order to distribute the individual in the population evenly along the Pareto front, the overall fitness function is then adjusted by sum of sharing distance. The sharing distance between individuals i and j is given by

$$SF(i,j) = 1 - \frac{d(X_i, X_j)}{\sigma_{share}} \quad if \; d(X_i, X_j) < \sigma_{share}$$
$$and \qquad (15)$$
$$SF(i,j) = 0 \quad otherwise$$

Where, $d(X_i, X_j)$ is a metric distance between two individuals in objective domain, σ_{share} is a predefined sharing distance. And, the overall fitness is defined by

$$Sharing \;\; Fit(i) = \frac{Fit(i)}{\sum_j SF(i,j)} \qquad (16)$$

where fit(i) is the inverse of Pareto rank(i) (1/rank(i) in this test).

The overall fitness values of individuals are to be used in the probabilistic selection process by the comparative overall fitness to the individual that has maximum overall fitness. The comparative fitness values are used to compare with random number. If they exceed the random number, the individual will be selected (roulette selection method.) MOGA usually has O (n^2) for a single round, because it needs to compute Pareto ranks and the sharing distance for all individuals.

E. Strength Pareto Evolutionary Algorithm 2 (SPEA2)

SPEA2 proposed by Zizler et al. [18] as an improvement of the original SPEA. Like SPEA, SPEA 2 uses two population of size P, the first (P) for the population and the second (P') for the archive. In SPEA, all non-dominated individuals in population P are copied to the archive P', so the size of the archive P' is varied from generation to generation. However, in SPEA2, the size of P' is fixed thus if the non-dominated individuals in a generation exceeds the size of P', they will be truncated, on the other hand, if they is less than P', some dominate individuals need to be added in the archive P'. The truncation and addition of dominated individuals are incorporate density information as a strategy to make the solutions distribute along the Pareto front. The density estimation of an individual i is defined as D(i) = $1/(d_i+2)$, where d_i is the distance of individual i from the nearest neighbor.

SPEA2 first selects all non-dominated individuals from the population P in the first round and then selects the combined population of P and archive of P' in the subsequent rounds. Unlike VEGAs and MOGA, the selection is deterministic rather than probabilistic. If the number of non-dominated individuals exceeds the fixed size of the archive P', the excess individuals will selects based on the density estimation. And if the non-dominated individuals fall short of the size of P', then the remaining non-dominated individuals (the next best Pareto front) will be selected until the archive has been filled. If however, the last selected Pareto front that exceed the size, the same truncation method will be employed. SPEA2 usually has O (n^2 log n) a single round, due to the density estimation calculation [17]

F. Nondominated Sorting Genetic Algoerithm 2 (NSGA2)

NSGA2 proposed by Deb et al. [19] as an improvement of the original NPGA. NPGA 2 uses two population of size P, the first (P) for the parent population and the second (P') for the offspring population. The two populations are combined for the selection process. All individuals in combined population are to be pass through the Fast Non-dominate Sorting (for Pareto ranking) and the Crowding Distance Assignment (for density (for distribution preservation). The algorithm first select all individuals that have Pareto rank smaller than P^{th} element in the sorted vector of combined population (always less than P individuals.) The remaining individuals need to fill up the selected population to P individuals are to be selected based on the Crowding Distance Assignment (more to less.) The Fast Distance Assignment and the Crowding Distance Assignment have the overall complexity of $O(MP^2)$ and O(MPlogP) respectively, where M is the number of objective functions.

IV. The Experiments

We run experiment on data from OR library that maintained by Prof. Beasley as a public benchmark data set (derived from Heng Seng data set with 31 stocks.) The data can be found at http://people.brunel.ac.uk/~mastjjb/jeb/orlib/portinfo.html
Two problem instances are considered for the experimentation: the case of cardinality constraint is the number of stocks in a portfolio is not larger than 10 (K = 10) and the case of cardinality constraints is the number of stocks in a portfolio is not larger than 5 (K = 5). For each case experiments have been run using VEGA, VEGA_Fuz1, VEGA_Fuz2, MOGA SPEA2, and NSGA2. The general approach adopted during the tests of each of the GA was to conduct 10 runs until the stopping criterion (a given number of generations) is reached. VEGA, VEGA_Fuz1, VEGA_Fuz2, MOGA, SPEA2 and NSGA2 results were recorded for 500, 1000 and 5000 generations with the number of population is 400.

The performance is measured by Generation Distance (GD) [19] for the last generation of the tests. GD is given as follow:

$$GD = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2} \qquad (17)$$

Where, d_i is the nearest distance between Pareto front of the results (PF known) and Pareto front of the benchmark solutions provided by the OR-library (PF true) and n is the number of population.

V. The Results

The graphical results of known Pareto fronts for VEGA, VEGA_Fuz1, VEGA_Fuz2, MOGA, SPEA2 and NSGA2 (N = 5000) are shown in Figure 4. SPEA2 seems to be the best when compare to the true Pareto front proximate by no constraint efficient front (NC Eff. Front) both in the terms of the closeness and distribution along the true front. VEGA does not perform well both in closeness to the true Pareto front (NC_Eff Front) and distribution along the known Pareto front. The results tend to cluster in two separate areas and not forming a line. This is caused by break up the population into subpopulations and each subpopulation is selected based on only single subpopulation alone. Also there is no any distribution preservation mechanism. For VEGA_Fuz1, the Fuzzy logic that combines the two objective together help improving the performance, especially for the closeness from the true Pareto front. In figure 4, the results from VEGA_Fuz1 tend to be clustered in a single area, also not forming a line along the Pareto front. But they seem to be quite close to the true Pareto front in its clustering area. Also, when we consider the Generation Distance (GD), in VEGA_Fuz1 the results are better than VEGA and are closed to those of MOGA. Thus, using Fuzzy logic to combine the two objectives and make evaluation by using Fuzzy rules does improve the performance in the term of the accuracy (as measured by GD). On the other hand, make selections based on Fuzzy rules of only two objective functions, the results tends to be clustered on the moderate areas not to the extreme ends, the two objectives are collapsed into a single combined objective. VEGA_Fuz2 is designed to make the results to be more distributed along the true Pareto front by including a third objective (sampling distance) into the Fuzzy rules of selection. The results of VEGA_Fuz2 are more distributed along the Pareto front than those of VEGA and VEGA_Fuz1 (see Fig 4). However, its performances in the term of GDs are not impressive, even though, they show some improvement from those of VEGA. This can be concluded that the inclusion of Sampling Distance into the Fuzzy rules helps to make the results more distributed but compromises the accuracy of the results. In figure 5, NSGA2's performance is almost inline with the Efficient Front but the distribution along the front is poor because.only a few points exist. .

This supports the claim that SPEA2 is among the best MOEAs. We can conclude that SPEA2 is also well applicable to portfolio optimization problems with realistic constraints as in this paper. Figure 6, 7 and 8 show GDs for all algorithms at 500, 1000 and 5000 populations (rounds) respectively, it also confirms the graphical results and the conclusion above of SPEA2.

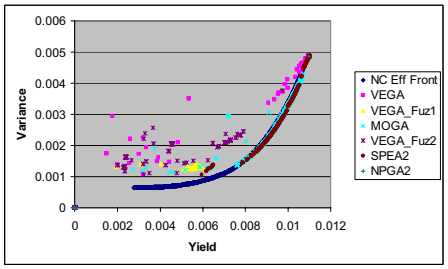

Fig 4: The Results for N= 5000

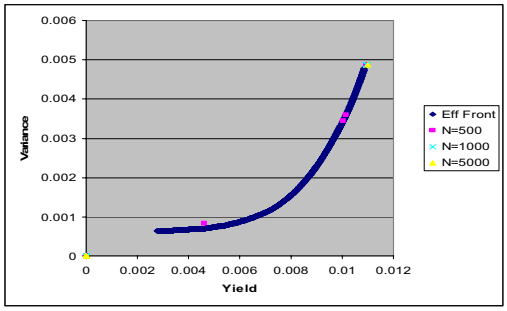

Fig 5: NSGA2 with N=500, 1000 and 5000

Comparing performance by using Generational Distances (GD) for different setting of cardinality constraint (K as the number of maximum stocks can be held in a portfolio), (see figures 6 and 7) we found that for N= 500, SPEA2 performs the best for both of instances (K = 5 and K =10). MOGA and NSGA2 perform roughly the same. VEGA_Fuz 1 performs moderately. VEGA_Fuz2 is the second worst and VEGA is the worst. However, Figure 4 shows that VEGA_Fuz 1 does not evenly distributed while VEGA_Fuz 2 improves the distributions but has to traded off with performance.

The results can be concluded that Pareto selections (used in MOGA, NSGA2, SPEA2 and NSGA2) are among the outperforming group vector selections as used in VEGA, however, incorporating Fuzzy logic into the vector selections do make an improvement. By mixing distribution preservation mechanism with objective functions (in the term of sharing fit in MOGA and Fuzzy rule in VEGA_Fuzzy) worsens the closeness performance (as measured by GD). Thus the distribution preservation mechanism should be set as a separate mechanism so that it will not interfere with the objective evaluation (as in SPEA2). Although VEGA_Fuz1 is not appropriate for find a Pareto front because of lacking distribution preservation mechanism, it could still be fine for any problems that do not require a set of traded off solutions. The advantage is that it has only O (n) time complexity and subjectively flexible of setting preferences for each objective.

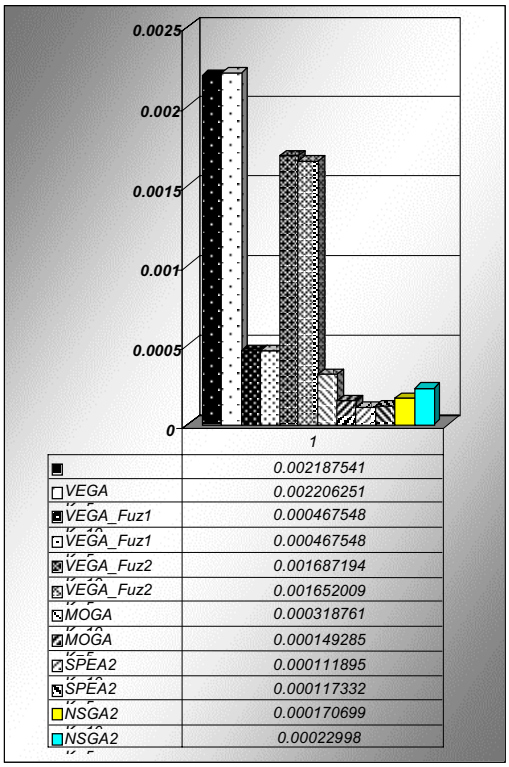

Fig 6: Generational Distance (GD) for N = 500

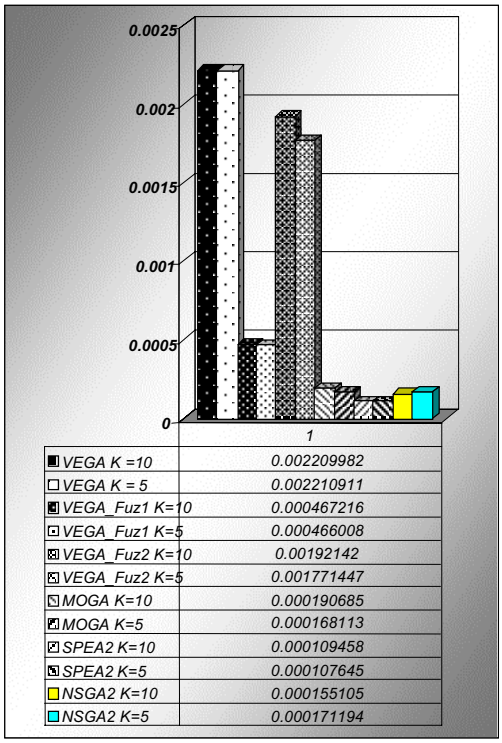

Fig 7: Generational Distance (GD) for N = 1000

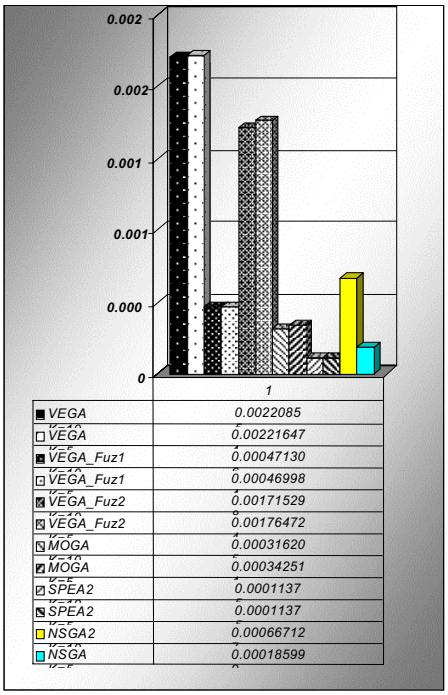

Fig 8: Generational Distance (GD) for N = 5000

VI. Conclusion

In this paper we apply various techniques of multiobjective genetic algorithms to solve portfolio optimization with some realistic constraints, namely cardinality constraints, floor constraints and round-lot constraints. We apply fuzzy logic to see whether it can improve performances of the Vector Evaluated Genetic Algorithm (VEGA). The results show that using fuzzy logic to combine optimization objectives of VEGA (in VEGA_Fuz1) for this problem does improve performances especially in Generation Distance from the true Pareto front to comparable of those of MOGA even though VEGA Fuz 1 is basically O(n) while MOGA is $O(n^2)$ but its solutions are tends to cluster around a few points. With additional fuzzy rules of Sampling Distance to make VEGA solutions are more distributed, the closeness performances are worsening. MOGA and SPEA2 are more complex algorithms but they perform better. SPEA2 perform the best even in comparatively small numbers of generation (N) and also has a good distribution along the Pareto front.

References

[1] Y. Crama, and M. Schyns,"Simulated Annealing for Complex Portfolio Selection Problems," *European Journal of Operational Research, v. 150 2003* pp. 546-571

[2] D. Maringer,, Portfolio Management with Heuristic Optimization_. Advanced in Computational Management Science Series Vol. 8, Springer, 2005

[3] C.Blum and A. Roli., "Metaheuristic in Combinatorial Optimization: Overview and Conceptual Comparison". *Computing Surveys, v. 35,, 2003, pp. 268-308*

4] F. R. Busetti, "Metaheuristic Approaches to Realistic Portfolio Optimisation," MSc thesis in Operation Research, University of South Africa, 2000 Available: http://arxiv.org/ftp/cond-mat/papers/0501/0501057.pdf on 04-Jul-06

[5] F. Streichert, Ulmer, H. Zell, A. "Comparing Discrete and Continuous Genotypes on the Constrainted Portfolio Selection Problem" Available: http:///www-ra.informatik.uni-tuebingen.de/

[6] S. Arnone, A. Loraschi, and Tettamanzi, "A genetic approach to portfolio selection," *Journal on Neural and Mass-Parallel Computing and Information Systems, Vol. 3* (1993), pp. 597-604

[7] A. Loraschi and A. Tettamanzi, An evolutionary algorithm for portfolio selection in a downside risk framework,_ *Working Papers in Financial Economics, Vol. 6, (1995*), pp. 8-12

[8] A. Loraschi, A. Tettamanzi, M. Tomassini, and P. Verda, Distributed genetic algorithms with an application to portfolio selection problems_, In *Artificial Neural Networks and Genetic Algorithms, eds.* D. W. Pearson, N. C. Steele, and R. F. Albrecht, Springer, Wein, 1995 pp. 384-387

[9] A. Mukherjee, R" Biswas, K. Deb, and A. P. Mathur, "Multi-objective Evolutionary Algorithms for the Risk-return trade-off in Bank Loan Management," *International Transaction in Operations Research,* vol. 9, pp. 583-597, 2002

[10]] F. Streichert, H. Ulmer, and A. Zell, "Hybrid Representation for Compositional Optimization and Parallelizing MOEAs" Available: http://drops.dagstuhl.de/opus/volltexte/2005/251/pdf/04461

[11] R. Subbu, Bonissone, P.P. Eklund, N. Bollapragada, S. Chalermkraivuth, K, "Multiobjective financial portfolio design: a hybrid evolutionary approach," *Evolutionary Computation, 2005. The 2005 IEEE Congress on,* Vol. 2 (2005), IEEE, pp. 1722- 1729

[12]H. M. Markowitz, "Portfolio Selection" *The Journal of Finance_, 7(1),* March, 1952 pp. 77-91

[13] E.J. Elton, M.J. Gruber, "Modern Portfolio Theory, 1950 to date," *Journal of Banking & Finance 21* 1997, pp. 1743-1759

[14] Black, F., M.C. Jensen, and M. Scholes, "The Capital Asset Pricing Model: Some Empirical Tests," in *Studies of the Theory of Capital Markets,* ed By M.C. Jensen, Praeger Publishers, In., New York, 1972

[15] , F. Streichert. H. Ulmer, , A. Zell "Evaluating a hybrid encoding and three crossover operators on the constrained portfolio selection problem"*Evolutionary Computation, 2004. CEC2004. Congress on, v.1 IEEE 2004* 932-939

[16] J.D. Schaffer, ., "Multiple Objective Optimization with Vector Evaluated Genetic Algorithm," In *Proceeding of the First International Conferenceon Genetic Algorithms: Genetic Algorithms and their Applications.* Lawrence Erlbaum, pp. 93-100., *1987.*

[17] E. Zitzler, , M. Laumanns ,and S. Bleuler, "A Tutorial on Evolutionary Multiobjective Optimization," In: Gandibleux, X., M. Sevaux, K. Sorensen, and V. T'kindt, eds., *Mrtaheuristics for Multiobjective Optimisation,* Springer, Berlin 2004.

[18]C.M. Fonseca, and P.J. Flemming, "Genetic Algorithm for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Forrest S., ed., *Proceeding of the Fifth International Conference on Genetic Algorithms,* Morgan Kaufmann, San Mateo, CA, pp. 416-423, 2001.

[19] E Zitzler ., M. laumanns and L. Thiele, "SPEA2: Improving Strength Pareto Evolutionary Algorithm," *Technical Report 103*, Computer Engineering and Networks Laboratory , Swiss Federation of Technology, Zurich, 2001.

[20] K. Deb, A. Pratab, S. Agrawal and T. Meyarivan, "A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, pp. 182-197, 2002.

[21] K.C. Tan, E.F. Khor and T.H. Lee, *Multiobjective Evolutionary Algorithms and Applications,* Springer, London, 2005.