メメティックアルゴリズムを用いた金融ポートフォリオの再構築

Use of Memetic Algorithms for Portfolio Rebalancing

アランニャ　クラウス[1]*　　伊庭　斉志[1]
Claus Aranha[1]　　Hitoshi Iba[1]

[1] 東京大学電気系工学学科
[1] University of Tokyo, Department of Electrical Engineering

Abstract: We present a system that uses Memetic Algorithms to perform long-term rebalancing of financial portfolios. This allows for a greater resilience to changes in the market. In our experiments, we achieved a number of portfolios which show stable return values even during market crash environments.

1 Introduction

Investment Portfolios are used by financial institutions in the management of long term investments, like savings accounts, retirement funds, etc. The idea of an investment portfolio is that if it is unwise to keep money still, for it loses value over time, it is also too risky to put everything into a few investment, for it exposes the capital to sudden changes in the market.

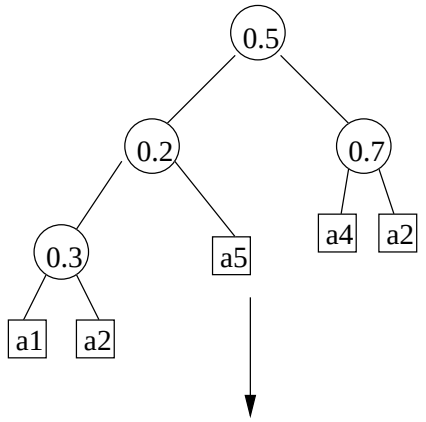
The Markowitz Portfolio Model describes how to minimize the risk of an investiment, by distributing the capital into multiple, counter correlated assets. This model can be used to calculate the optimal distribution of capital in order to minimize the risk of an investiment, for a given target return.

This optimization problem is called the Portfolio Optimization Problem, and has been heavily studied in the fields of Operational Research and Financial Engineering. In its original form, it is possible to solve the Portfolio Optimization problem using techniques such as Linear or Quadratic Programming. However, as the number of asset grows, and constants such as lots and limits are added, the problem require more elaborate techniques to be solved, like Genetic Algorithms.

While the Markowitz Model for portfolio optimization concerns itself with the optimal portfolio structure for a given market state, in real life we are also concerned with fluctuations in the market and how to adjust the Portfolio in response to those fluctuations.

However, because of trading costs, this response must be limited so that gains due to it are not negated by the cost of changing the portfolio's position. This problem, known as *Portfolio Management* or *Rebalancing*, is very appropriate to Evolutionary Algoriths, whose main characteristic is their robustness to changes in their environment.

In this work, we extend the MTGA (Memetic Tree Based Genetic Algorithm) to deal with the Dynamic Portfolio Optimization problem. The MTGA was previously described in [2]. It is a Genetic Algorithm enhanced by a tree structure, which produced good results for optimizing static portfolios (See figure 1).



	a1	a2	a3	a4	a5
Name	AMZN	GOOG	INTL	MSFT	YHOO
Weight	0.03	0.22	0	0.35	0.4

Figure 1: A tree genome and its corresponding portfolio. The values in the intermediate nodes indicate the weight of the left sub tree. The complement of that value is the weight of the right sub tree. The final weight of each asset (a_x) is given by the sum of the weights of all occurrences of that asset in the tree.

The tree structure for the genome allows the use of local optimization in the evolutionary process. In our previous work, this local optimization was used to find the optimal values for the weights. At that time, we suggested that the local optimization could also be used to rebalance the portfolio according to the changes in the market conditions. In this work we continue that suggestion by developing a portfolio strategy based on the use of the local optimization on a dynamic market scenario.

We validate our proposal by means of simulated

*連絡先：東京大学電気系工学学科
E-mail: caranha@iba.t.u-tokyo.ac.jp

experiments. We use the MTGA with local optimization to generate a portfolio strategy for a period of one year, based on an optimal portfolio generated at the start of the period. We compare this strategy with others, one where the optimal portfolio is re-generated monthly, and the benchmark performance monthly.

The results shows us how using the local optimization to rebalance the portfolio to adapt to new market conditions generate a stable level of return, even during the depression scenario experienced last year.

2 The Portfolio Problem

The resource allocation problem is a traditional optimization problem, which consists of distributing a limited "resource" to a number of "jobs", in order to satisfy one or more utility functions [4].

The Portfolio Optimization problem falls in this category. The limited resource is the capital available for investment, and the jobs are the varied assets in which this capital can be invested (for example, company stock or foreign currency. The utility functions in this problem are the Portfolio Estimated Return, to be maximized, and the Portfolio Risk, to be minimized.

The model for the Portfolio Optimization problem was formally proposed by Markowitz [6]. Markowitz's Portfolio Model could be solved by numerical methods, like Quadratic Programming [12]. However, when adding real world constraints to the problem (for example, large number of assets, restrictions to the values of weights, trading costs, etc), the search space becomes large and non-continuous, and heuristics, such as evolutionary computation, must me used to solve the problem.

An extra layer of complexity is added to this problem when dynamic market behavior is considered.

2.1 The Markowitz Model

A portfolio P as a set of N real valued weights $(w_0, w_1, ...w_N)$ which correspond to the N available assets in the market. These weights must obey two basic restrictions [12]: The total sum of the weights must be equal to one; and all weights must be positive.

The utility of a portfolio is measured by its *Estimated Return* and its *Risk*. It is calculated as:

$$R_P = \sum_{i=0}^{N} R_i w_i \tag{1}$$

Where N is the total number of assets, R_i is the given estimated return of each asset, and w_i is the weight of each asset in the portfolio.

The risk of an asset is given as the variance of its return over time (variability). The risk of the portfolio is defined as:

$$\sigma_p = \sum_{i=0}^{N} \sum_{j=0}^{N} \sigma_{ij} w_i w_j \tag{2}$$

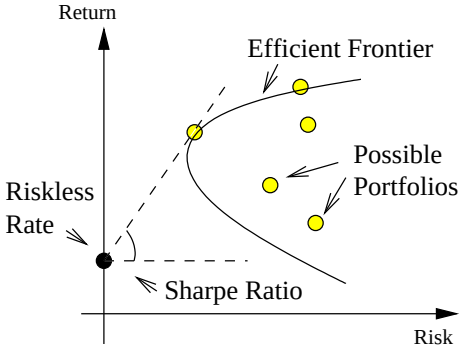


Figure 2: Risk-return projection of candidate portfolios. The search space is bounded by the Efficient Frontier. Sharpe ratio is the angle of the line between a portfolio and the risk-free rate.

Where $\sigma_{ij}, i \neq j$ is the covariance between i and j. While the risk is usually stated as the variance of the return of a given asset, there are other definitions of risk that have been used to bias the resulting portfolios towards certain kinds of investment strategies. For other risk metrics, see the works of Harish[9] and Shu[7].

These two utility measures can be used separately to determine the optimal portfolio, or they can be combined. The *Sharpe Ratio* measures the trade off ratio between risk and return for a portfolio, and is defined as follows:

$$Sr = \frac{R_P - R_{riskless}}{\sigma_p} \tag{3}$$

Where $R_{riskless}$ is the risk-free rate, an asset which has zero risk and a low return rate (for example, government bonds). The relationship between these three utility measures is illustrated in Figure 2.

2.2 Real World Constraints

The Markowitz Model, as described above, can be solved by optimization techniques such as Quadratic Programming [12]. However, when real world constraints are added, the problem becomes too complex for simple optimization techniques. Practical portfolios are composed from markets with hundreds to thousands of available assets, and the calculation of risk measures grows quickly in relation to the number of assets.

Also, real world applications have constraints related to the values of weights, and to trading. Weight constraints include maximum and minimum weights and lots (indivisible unit of a held asset). These constraints turn the search space non-convex, making the problem harder.

Trading constraints include minimum and maximum trading volume (how much of an asset you can buy at once) and trading cost (proportional to the amount of asset traded). These constraints take effect when multiple scenarios (time periods) are con-

sidered, and affect greatly the outcome of the optimization process.

2.3 Dynamic Market Behavior

We call Dynamic Portfolio Optimization, or Rebalancing, the problem of generating a trading strategy that keeps the optimized portfolio with a high level of return and risk, according to the policies of the portfolio operator. This policy must modify the optimized portfolio according to changes in the return values of the assets in the market, so that the target return is achieved in spite of those changes.

The main question when rebalancing a portfolio is how to reduce the trading cost. To change from portfolio P to portfolio P', there is an operational cost proportional to the difference between P and P'. Usually, the trading cost C takes a form similar to:

$$C_a = \begin{cases} k_c & \text{if } 0 \text{ ¡ } T_a < T_{min} \\ T_a * \delta_c & \text{if } T_a > T_{min} \end{cases} \quad (4)$$

$$C = \sum C_a \quad (5)$$

Where C_a is the cost associated with the trading of one asset. This cost is either a fixed minimum value (k_c), for transactions below a certain amount (T_{min}), or a percentage of the total transaction value (δ_c), if the transaction is above T_{min}.

In some situations, the cost to rebalance the portfolio to the new global optimum in a dynamic market environment may be higher than the improvement in the utility function. To avoid problems like this, it is essencial to add a new objective function to the Dynamic Portfolio Optimization problem: the transaction cost, also represented as the distance between two portfolios.

3 Related Research

Two important questions in the Portfolio Optimization problem are how to select the assets and the weights. The simplest answer is to use a single array with one real value for the weight of each asset [3, 5].

A more elaborated strategy to select the assets which will participate of the portfolio is to use two arrays: a binary array, which indicates whether an asset is part of the portfolio or not, and the real valued array to calculate the weights of the assets [1, 8].

A somewhat different way to assemble the portfolio is to use GP to evaluate each asset. The GP can be used to calculate the suggested weight of each asset from technical indicators [10], or to generate a ranking of assets, which will be used to select the assets to add to the portfolio [11].

4 Memetic Tree-based Genetic Algorithm

The basic idea of the MTGA is to establish a hierarchical set of relationships between the assets that belong to the portfolio, and use those relationships to improve the exploitation abilities of the Genetic Algorithm.

The tree structure leads to this exploitation by dividing-and-conquering the portfolio in two different ways: It allows the evaluation of the fitness of individual trees, which leads to the crossover based on these fitness values. It also allow the local search step to optimize many 2-variable nodes, instead of one giant portfolio with hundreds of variables at once.

4.1 Tree Representation

Each solution in the Genetic Algorithm is represented as a binary tree. Each non-terminal node holds the weight between its two sub trees. This weight is a real value, w, between 0 and 1, which indicate the weight of its left sub tree (the choice of left over right is arbitrary). The right sub tree of has weight $1 - w$. Each terminal node holds the index of an asset in the market. It is possible to have more than one terminal pointing to the same asset in the same tree. Figure 1 shows this representation.

To extract the portfolio from this representation, we calculate the weight of each terminal node by multiplying the weights of all nodes that need to be visited to reach that terminal, starting from the root of the tree. After all terminal nodes are visited, the weights of those terminals that point to the same asset are added together. The assets which are not mentioned in the tree are assigned a weight of 0.

There are some characteristics of this structure which are important to consider when implementing an Evolutionary Algorithm based on it:

First Every sub tree in an individual can be treated as if it were a normal tree. This is because the root node's structure is identical to that of any intermediate node. This allows each sub tree to have its own individual fitness, calculated in the same way as the fitness of the main tree. This is used in the specialized genetic operators.

Second A portfolio extracted from this representation is always normalized. This is because the weight on each node is limited to the 0..1 interval, and the weight of each terminal is the multiplication of the node weights. Because of the first characteristic, this also applies to sub trees.

Third The maximum number of assets in a portfolio represented by a tree is limited by the depth of the tree. As each terminal corresponds to one asset, a tree with depth d may hold at most 2^{d-1} assets. Because of incomplete trees and

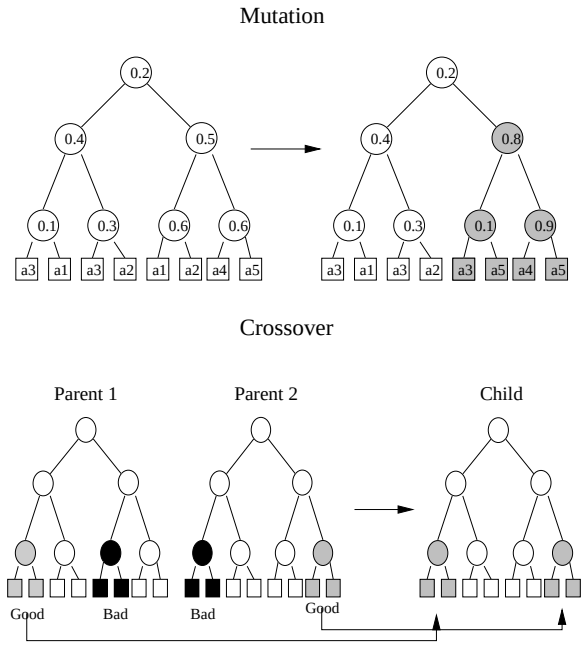


Figure 3: Crossover (BWS) and Mutation operators for the tree representation.

terminals with repeated assets, usually the actual number of assets in a tree is much smaller than this.

4.2 Evolutionary Operators

The *mutation* operator works by cutting off the tree at a point, and replacing the cut-off sub tree with a randomly generated sub tree. In this work, the cut-off point is selected by first randomly choosing the target depth (with a linear probability), then following a random path from the root node until the desired depth is achieved. This selection method favors cut-off points near the leaves, which results in less aggressive mutations (see Figure 3).

The *crossover* operator works by exchanging sub trees between two individuals. One crossover point is chosen for each tree, and the sub trees that start from that point on are swapped between the two trees. In this operator, the sub tree with the highest fitness from the first parent is exchanged with the sub tree with worst fitness in the second parent. This operator usually improves the fitness of the individual receiving the better sub tree [2]. This means that the BWS can be used to emphasize a policy of exploitation in the search (See Figure 3).

4.3 Local Search

The local search operator executes a simple hill climbing optimization on each node of an individual. It starts on the deepest non-terminal nodes, and then works its way back towards the root.

For each visited node, the return and risk value for the left and right children are obtained, and used to calculate the utility function as if the node was a two-asset portfolio. The pseudo-code for the hill climbing function can be seen in Figure 4.3.

```
while (meme_speed > meme_tresh AND 0 < weight < 1)
do
   old_fitness = fitness;
   weight = weight + meme_speed;

   if (weight > 1)
      weight = 1;
   if (weight < 0)
      weight = 0;

   calculate_fitness(weight);

   if (fitness < old_fitness)
      meme_speed = meme_speed * meme_accel * -1;
done
```

Figure 4: Algorithm for local search

Where the parameter *meme_speed* is the value by which the weight changes every iteration, *meme_accel* must be < 1.0, and is the value by which *meme_speed* changes every time the weight cross the optima point. And *meme_tresh* is the minimum value of *meme_speed* which signalizes the end of the search. The search also ends if the weight reaches 1.0 or 0.0 (when the optima is not in the weight range 1..0).

4.4 Rebalancing

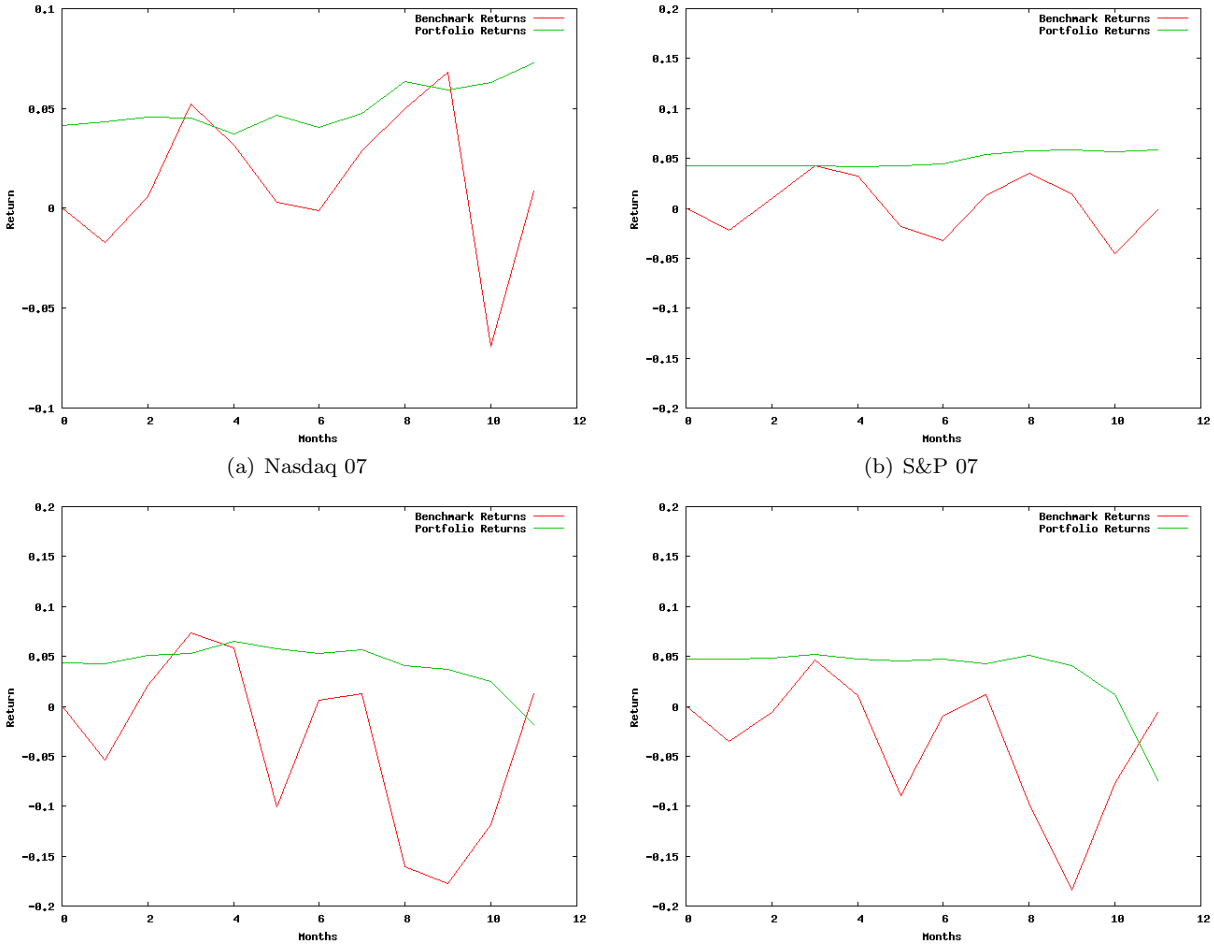
In this work we introduce the use of the local search step with the tree structure to rebalance the portfolio according to changes in the Market environment.

First the optimal portfolio is evolved for the initial time period, following the steps described above. After that, for each subsequent time period new values for the estimated return and correlation between the assets is calculated. Based on these values, the system executes the local optimization step on the optimal portfolio's trees, in order to calculate new best weights based for the previous assets.

By realizing the Rebalancing strategy in this manner, the portfolio realizes the asset selection in the first time period, and then uses only the assets that were selected in the subsequent time periods. If we can guarantee a good initial asset selection, the rebalancing policy will be able to react to market changes to keep the return levels.

5 Experiments

We use a simulation of the market behavior to analyse the results of the portfolio strategy generated by MTGA on a dynamic market. Our goal is to determine if the Memetic Strategy is able to generate a stable, low risk return during the whole period. In this section we describe the experiment and its results.



(a) Nasdaq 07

(b) S&P 07

(c) Nasdaq 08

(d) S&P 08

Figure 5: Experiment result for the Rebalancing strategy in the four datasets, compared to the benchmark portfolio

5.1 Datasets

We use two datasets in our simulations. The NASDAQ dataset contains assets from the NASDAQ100 index, which is composed mainly of technology related industry. The SP500 dataset contains assets from the S&P 500 index, which has a more varied composition, with assets from industry of many different fields.

For each dataset, we have two 12-month data periods, one for the year of 2007 and one for 2008. The year 2007 data represents a more stable period of market movements, and serves as a benchmark for normal behavior.

In the late part of the year 2008 there was the big crash, which is reflected in the 2008 dataset. The goal of this dataset is to verify if the policy generated by MTGA and its rebalancing is able to weather a period of economic downturn.

5.2 Parameters

We used 500 generations and 200 individuals per generation. The crossover rate was 0.8, and the mutation rate 0.03. The tree depth was 8 (128 terminals in a full tree). The riskless asset's return was set as 0.03.

For the MTGA system, we used a 0.6 chance of executing the local search step for each individual. The chance of executing the guided crossover was 0.6 per crossover. The sensitivity of the system for these parameters is not explored in this work.

The parameter for the local optimization step are: 0.1 for meme_speed, 0.333 for meme_delta, and 0.003 for meme_tresh. Other than meme_tresh, which changes the precision of the search, changing these values does not seem to affect the quality of the local search.

5.3 Results

In figure 4.4 below we list the results of our experiments with the four datasets. The results shown are the return values of both policies for each month in

the one year time period.

As we can see in the figure, the portfolio rebalanced by the local optimization shows a very stead return value, while the benchmark portfolio for the same period displays very strong raises and drops. This show that the evolutionary algorithm was able to find a low risk group of assets even when using only data for the first month.

In the two datasets for 2007, the portfolio returns are more stable, slowly rising. For the 2008 datasets, we see that the portfolio return shows a drop in the last the months of the year - this is because crash in the stock market has changed the market condition to a very different one when compared with the start of the period. Still, the fall is a gradual one, and it is possible to see when a complete re-optimization of the portfolio becomes necessary.

6 Conclusion

We have tested the use of Memetic Algorithms and Local optimization for the Rebalancing of portfolios in a Dynamic Market context. Rebalancing is an important step in real life application of Portfolio optimization, because it allows the portfolio to adapt to changes in the market, like bull periods or sudden crashes.

The genetic algorithm is specially well equipped to deal with rebalancing of portfolios. This is because of its evolutive characteristics. In our experiments, we have observed that using genetic algorithms to rebalance the portfolio generated in the start of a 12-month period results in a very stable investment strategy, even during crash periods.

References

[1] Claus Aranha and Hitoshi Iba. Modelling cost into a genetic algorithm-based portfolio optimization system by seeding and objective sharing. In *Proc. of the Conference on Evolutionary Computation*, pp. 196–203, 2007.

[2] Claus Aranha and Hitoshi Iba. A tree-based ga representation for the portfolio optimization problem. In *GECCO - Genetic and Evolutionary Computation Conference*. ACM Press, July 2008. accepted.

[3] Ronald Hochreiter. An evolutionary computation approach to scenario-based risk-return portfolio optimization for general risk measures. In M. Giacobini et al., editor, *EvoWorkshops 2007*, No. 4448 in LNCS, pp. 199–207. Springer-Verlag, 2007.

[4] Toshihide Ibaraki and Naoki Katoh. *Resource Allocation Problems - Algorithmic Approaches*. The MIT Press, 1988.

[5] Piotr Lipinski, Katarzyna Winczura, and Joanna Wojcik. Building risk-optimal portfolio using evolutionary strategies. In M. Giacobini et al., editor, *EvoWorkshops 2007*, No. 4448 in LNCS, pp. 208–217. Springer-Verlag, 2007.

[6] H. Markowitz. *Mean-Variance analysis in Portfolio Choice and Capital Market*. Basil Blackwell, New York, 1987.

[7] Shu ping Chen, Chong Li, Sheng-Hong Li, and Xiong wei Wu. Portfolio optimization with transaction costs. *Acta Mathematicae Applicatae Sinica*, Vol. 18, No. 2, pp. 231–248, 2002.

[8] Felix Streichert, Holger Ulmer, and Andreas Zell. Evolutionary algorithms and the cardinality constrained portfolio optimization problem. In D. Ahr, R. Fahrion, M. Oswald, and G. Reinelt, editors, *Operations Research Proceedings*. Springer, September 2003.

[9] Harish Subramanian, Subramanian Ramamoorthy, Peter Stone, and Benjamin J. Kuipers. Designing safe, profitable automated stock trading agents using evolutionary algorithms. In *GECCO 2006 - Genetic and Evolutionary Computation Conference*, pp. 1777–1784, Seattle, Washington, July 2006. ACM Press.

[10] James Cunha Werner and Terence C. Fogarti. Genetic control applied to asset managements. In J.A. Foster et Al., editor, *EuroGP*, LNCS, pp. 192–201, 2002.

[11] Wei Yan and Christopher D. Clack. Evolving robust gp solutions for hedge fund stock selection in emerging markets. In *GECCO 2007 - Genetic and Evolutionary Computation Conference*, London, England, July 2007. ACM Press.

[12] Yuh-Dauh-Lyu. *Financial Engineering and Computation*. Cambridge Press, 2002.