Using Memetic Algorithms To Improve Portfolio Performance In Static And Dynamic Trading Scenarios

Claus Aranha
Department of Electrical Engineering
The University of Tokyo
Tokyo, Japan
caranha@iba.t.u-tokyo.ac.jp

Hitoshi Iba
Department of Electrical Engineering
The University of Tokyo
Tokyo, Japan
iba@iba.t.u-tokyo.ac.jp

ABSTRACT

The Portfolio Optimization problem consists of the selection of a group of assets to a long-term fund in order to minimize the risk and maximize the return of the investment. This is a multi-objective (risk, return) resource allocation problem, where the aim is to correctly assign weights to the set of available assets, which determines the amount of capital to be invested in each asset.

In this work, we introduce a Memetic Algorithm for portfolio optimization. Our system is based on a tree-structured genome representation which selects assets from the market and establish relationships between them, and a local hill climbing function which uses the information available from the tree-structure to calculate the weights of the selected assets.

We use simulations based on historical data to test our system and compare it to previous approaches. In these experiments, our system shows that it is able to adapt to aggressive changes in the market, like the crash of 2008, with reduced trading cost.

Categories and Subject Descriptors

J.1 [**Administrative Data Processing**]: Financial; I.1.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search

General Terms

Algorithms, Experimentation

Keywords

Memetic Algorithms, Portfolio, Operational Research,

1. INTRODUCTION

Investment Portfolios are used by financial institutions in the management of long term funds, like savings accounts, retirement funds, etc. The idea of an investment portfolio

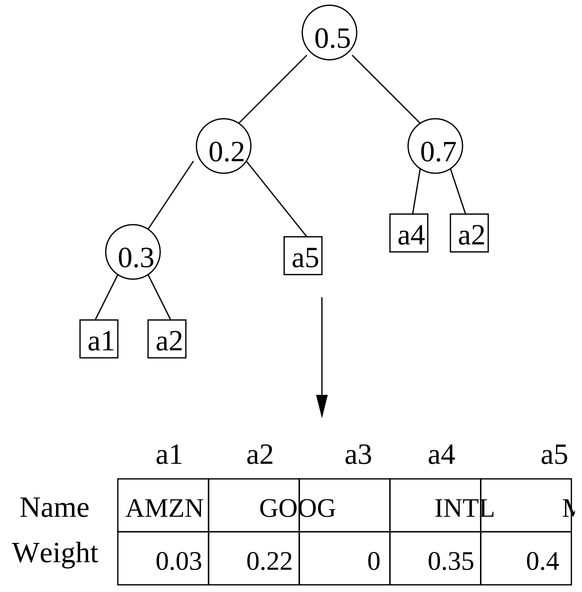

Figure 1: A tree genome and its corresponding portfolio. The values in the intermediate nodes indicate the weight of the left sub tree. The complement of that value is the weight of the right sub tree. The final weight of each asset (a_x) is given by the sum of the weights of all occurrences of that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

is that if it is unwise to keep money still, for it loses value over time, it is also too risky to invest everything into a small number of assets, for it exposes the capital to sudden changes in the market.

The Markowitz Portfolio Model [9] describes how to minimize the risk of a financial portfolio, 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 investment, for a given target return.

While the Markowitz Model for portfolio optimization concerns itself with the optimal portfolio for a snapshot of the market (a single set of return and risk values), in real life we are also concerned with fluctuations in the market over time and how to dynamically adjust the Portfolio in response to those fluctuations. We call these adjustments *Rebalancing* the portfolio; the main challenge when rebalancing a portfolio is to change its position enough to keep desired levels of risk and return, while avoiding the trading costs associated with any changes in the position.

Portfolio Optimization is a problem that has been popular with the genetic algorithm community in recent years. The reason for this is that when the problem is stated with real life constraints, like a large number of assets, trading lots and weight restrictions, it becomes too hard to solve with deterministic methods. Evolutionary Algorithm approaches have shown good results in this situation.

However, so far most works have concentrated on the solution of single-scenario cases of the portfolio optimization problem. In this work, we explore the use of Memetic Algorithms for the long-term management of Portfolios.

Memetic Algorithms are an hybrid heuristic of a genetic algorithm and local optimization. In general, the genetic algorithm improves the solution in large strokes, while the local optimization fine tunes the solutions generated by the GA. We have chosen to extend the Tree-based Genetic Algorithm, described by Aranha in [2], with a hill climbing algorithm. We call the new system MTGA (Memetic Tree based Algorithm). The GA generates a tree structure (Figure 1) that includes information about the relationship of the assets in the market, while the hill climbing algorithm fine tunes the weights between those assets.

We propose that the hybridization of the MTGA leads to an increased performance of portfolio generation, when compared with pure GA solutions, and that the local optimization step can be used to perform rebalancing in Dynamic Market scenarios. To validate these proposals, we used simulated experiments based on historical data of large markets in the last three years.

Our results are very encouraging, and point towards the validity of using local optimization for improving Genetic Algorithms with real valued genome representations, and of using Memetic Algorithms to provide solutions that are able to adapt to dynamic environments.

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 [5].

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 Estimated Return of the portfolio, to be maximized, and the Risk, to be minimized. By reducing its risk, the investment becomes less susceptible to drastic changes in the market.

The model for the Portfolio Optimization problem was formally proposed by Markowitz [9]. Markowitz's Portfolio Model could be solved by deterministic methods, like quadratic programming [16]. 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 be used to solve the problem.

2.1 The Markowitz Model

Let us define a portfolio P as a set of N real valued weights $(w_0, w_1, ...w_N)$ which correspond to the N available assets

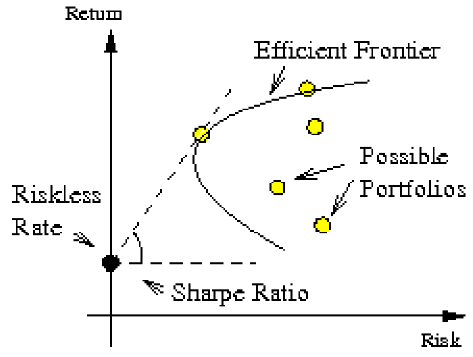

Figure 2: Risk-return projection of candidate portfolios. The search space is bounded by the Efficient Frontier. Sharpe ratio is the angle of the line between a portfolio and the risk-free rate.

in the market. These weights must obey two basic restrictions [16]: 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*. The estimated return 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. In this work, the estimated return is calculated as the moving average of the past 12 months for each asset, but other estimation methods can be used instead.

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

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

Where $\sigma_{ij}, i \neq j$ is the covariance between i and j, and $\sigma_{ii} = \sigma_i^2$ is the deviation of the estimated return of asset i. 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[12] and Shu[10].

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 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, in face of a dynamically changing market. 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:

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

$$C = \sum C_a \tag{5}$$

Where C_a is the cost associated with the trading of one asset, and T_a is the amount of the asset actually traded. 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 essential to add a new goal to the Dynamic Portfolio Optimization problem: to minimize the transaction cost, also represented as the distance between two portfolios.

3. RELATED RESEARCH

The portfolio Optimization Problem has enjoyed a lot of popularity in recent years. While multiple authors have tried variations of the main approaches in this field, we can group the main lines of work in two different ideas: Weight Arrays and Trading strategies, which are differentiated mainly by whether different weights for each asset are calculated or not, and whether all assets are evaluated at once, or separately.

In particular, we have not yet observed the use of hybrid heuristics, like Memetic algorithms, to bridge the gap between these two approaches among the current literature. We expect that our current work can help fulfill that role.

3.1 Weight Arrays

The main line of research in Portfolio Optimization using Evolutionary computing is the use of some sort of array as the genome for an individual, with the goal of using the GA operators to optimize the weight values in the array.

In the array representation an individual is composed of an array A with as many elements as there are assets in the market. Each element a_i is a real value, that defines the exact weight of that particular asset in the portfolio. A few representative examples of this approach would be [7, 4].

However, it is hard for the normal Genetic Algorithms operators (crossover, mutation) to fine tune the values of real valued genomes without specialized crossover operators [13]. Intuitively, regular crossover operators usually take the value of either of the parents, or their average, as the new value for a chromosome. This covers all possible values for binary valued chromosomes, but not so for real valued ones.

Another problem with this representation is that the real valued array does not include information about the covariance relationship between the different assets, which makes the search blind to a important piece of information in the utility function.

Specially, the large number of assets in real world markets makes it impractical to expect that crossover and mutation alone be able to converge the weights of an array with possibly hundreds of assets. In [8], a subset of all the assets is randomly chosen to form the weight array, which illustrates the difficulty of optimizing the values of multiple assets.

To address this issue, some works used an hybrid representation with binary and real valued arrays [1, 11]. The binary array defines whether an asset is part of the portfolio or not, while the real valued array evolves the weight for all assets.

While this approach solves partially the problem of multiple assets not related to the optimal portfolio making it to the final solution with very small weights, it does nothing to relieve the problem of fine tuning individual weights of assets, or establishing relationships between them.

3.2 Trading Strategies

The other popular approach to the Portfolio Optimization problem with evolutionary algorithms is the use of *Trading Strategies*. We define a trading strategy system as one where the focus is not in defining the weight of each asset, as in the previously discussed works. Instead, the Trading Strategies approach consists of analyzing each asset individually, and using some sort of evolved rule to decide if each particular asset should be part of the portfolio or not.

In one example of this line, Szeto creates rules based on the values of moving averages with different lengths to decide if a certain asset will or will not be included in the portfolio [6]. Wei and Clack, use more complex GP rules, which include varied indicators of each assets financial performance to generate a decision of whether including an asset or not in a portfolio with a limited number of open "slots" [14, 15].

However, after the assets are chosen, equal weights are assigned to all assets participating in the portfolio, which makes us wonder if a better result would not be achieved by optimizing the final weights of these chosen assets as well.

4. MEMETIC TREE-BASED GENETIC ALGORITHM

The MTGA is an extension to the TGA algorithm described in [2]. TGA introduced a new genetic structure to resource allocation problems. In this tree structure (fig 1), the leaves of the tree are the assets to be optimized, and the intermediate nodes express the relative local weights between these assets.

The TGA's tree representation, when compared to an array representation of weights, is able to express and learn relationships between assets, and it also allows for parts of the genome to be evaluated by the fitness function, which led to specific crossover operators.

In the MTGA, we add a local optimization step to the this system. The optimization step uses a hill climbing algorithm on the local weights (intermediate nodes) of the tree, in a recursive bottom-up fashion, so that for each node only a simple two variable optimization needs to be done. There are two main motivations for the use of Memetic Algorithms in this problem.

The first motivation is that Memetic Algorithms are believed to be more appropriate for real valued genomes than pure Genetic Algorithms [13]. This is because the standard crossover and mutation operators in GA are not well suited for finely tuning real-valued variables in the genome representation. This is specially true for the TGA, where crossover and mutation only operate on the tree structure of the genome, and can only affect weights indirectly, by manipulating the depth in the tree where an asset is found.

So we add the local optimization step to fine tune the weight parameters. The optimization becomes a two-stage process: In the first stage, the crossover and mutation operators select the assets, and establish relations between them using the tree structure. In the second stage, the local optimization directly modifies the weights of the intermediate nodes to establish the optimal weights for the assets given the tree structure of that individual.

The second motivation is to implement portfolio rebalancing, as presented in section two, using the local optimization step. This allows us to minutely change the weights of each asset belonging to a portfolio, adapting it to changes in the market, while maintaining its structural characteristics. This form of rebalancing generates portfolios that, while not optimal when considering only the risk/return objective functions, have a very low distance from the previously held portfolio, and thus very low trading costs associated with them.

4.1 Tree Representation

An individual's genotype in MTGA is represented as a binary tree.

Each non-terminal node holds the weight between its two child sub trees. This weight is a real value, w, between 0 and 1 inclusive. The left sub tree has weight w, and 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 (i.e. they have not been selected to the portfolio).

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

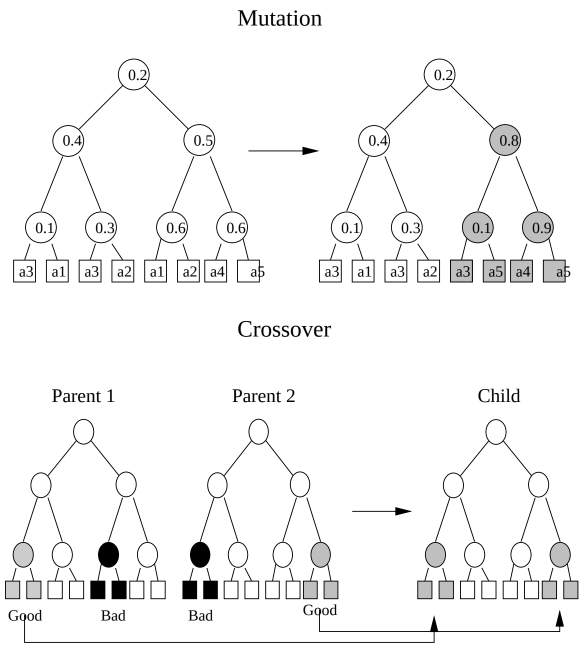

Figure 3: Crossover (BWS) and Mutation operators for the tree representation.

4.2 Evolutionary Operators

Because of its tree representation, the basic crossover and mutation operators of MTGA are similar to those of a GP. Mutation occurs by cutting off the tree at one point, and randomly generating a new sub tree from that point. Crossover occurs by switching sub trees of two individuals at a randomly chosen point. In our implementation, the cut-off point is selected by first randomly choosing the target depth, and then following a randomly selected path from the root to that depth.

Besides these basic operators, MTGA also includes the *Best Worst Sub tree* (BWS) crossover operator. In this operator, the sub tree with the highest fitness from one parent is exchanged with the sub tree with worst fitness of the other parent (see Figure 3). This operator uses the recursive fitness characteristic of the tree based representation to improve the exploitation characteristics of the genetic search [2].

In our implementation, the probability that the BWS crossover will be executed instead of a regular crossover during the breeding step of the generic algorithm is controlled by a parameter. In the experiments reported, this parameter is set as 0.6. The influence of the value of the BWS parameter in the final result is not examined in this work.

4.3 Local Search Step

The MTGA improves over the TGA by adding a local optimization step. This step aims to finely tune the intermediate weights in the tree representation. It complements the structural search done by the crossover and mutation operators.

A common question in Memetic Algorithms is how often should local learning be applied [3]. If you apply it to all individuals in the population, the entire algorithm may become too costly. We have opted for using a parameter that determines the probability of one individual being selected for local optimization. For the experiments discussed in this paper, this probability was set at 0.6.

At the start of every generation, a number of individuals from the population is chosen with the above probability. Before the fitness evaluation step, we execute the local optimization on these selected individuals.

For each selected individual, the local search operator performs a recursive hill climbing optimization. Algorithm 1 is executed, starting from the root node. The algorithm calls itself, descending to the deepest level of the tree where the risk and return value of the two-asset portfolio of that node is calculated. This calculation is performed by algorithm 2. After all intermediate nodes in the bottommost level are calculated in this way, the resulting values are used to calculate the two asset portfolio for the nodes one level above, repeating this process recursively until the program returns to the root node.

Algorithm 1 Recursive Tree Optimization(tree node)

```
if Child Nodes are not leaves or locally optimized then
    Recursive Tree Optimization(left child)
    Recursive Tree Optimization(right child)
end if
$weight$ = Local Search(this node)
Calculate Risk and Return ($weight$)
return  New Risk and Return Values
```

Algorithm 2 Local Search(tree node)

```
Require: Child nodes are leaves or locally optimized.
Ensure: Current node is locally optimized
while ($meme\_speed > meme\_thresh$) AND
($0 < weight < 1$) do
    $old\_fitness = fitness$
    $weight = weight + meme\_speed$
    if $weight > 1$ then
        $weight = 1$
    end if
    if $weight < 0$ then
        $weight = 0$
    end if
    calculate_fitness($weight$)
    if $fitness < old\_fitness$ then
        $meme\_speed = meme\_speed * meme\_accel * -1$
    end if
end while
```

In algorithm 2, *meme_speed, meme_accel and meme_thresh* are parameters. *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_thresh* 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.

A side-effect of the local optimization is the generation of *Introns* in an individual's genome. An intron is a part of the genotype that do not affect the phenotype. In this representation, an intron is a sub tree that does not influence the final portfolio, because its parent node has a local weight equal to 0 or 1.

While introns are known to increase processing time and complexity of results in GP, they also allow for more variety in individuals generated from crossover, and, in the particular case of the MTGA, can be useful if reactivated during rebalancing (see next subsection). In this work, we do not prune introns from the genome.

4.4 Rebalancing

Rebalancing means making small adjustments and corrections on a portfolio over time, in order to adapt it to changes in the market conditions.

We can use the local optimization step described above to achieve this. Just as the local optimization adjusts the weights of an individual after crossover and mutation have taken place during normal evolution, we expect that it can in the same way correct the weights of the final individual after changes in the market values.

In practice, at first a portfolio is created by running the MTGA, and choosing the best individual of the final population. The tree structure (genotype) of this individual is kept. Then, after a certain amount of time (defined by the trader's policy) has passed, new values for the estimated return and correlation between the assets is calculated. Based on these new values, the system executes algorithms 1 and 2 on the tree structure of the original portfolio, generating the rebalanced portfolio.

By executing the rebalancing strategy in this manner, the portfolio completes the asset selection in the initial time period, and from then on it uses only the assets already selected. So, if it 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

In order to verify the performance of MTGA, we execute two simulation experiments which explore different aspects of the problem.

In the first experiment, we evolve a portfolio using the MTGA on different markets and scenarios (time periods), comparing the performance of this portfolio with other evolutionary approaches and the market index. The goal of this experiment is to establish to what degree Memetic Algorithms are better at fine tuning the values of portfolio weights, and what influence this have at the portfolio's performance.

In the second experiment, we apply the MTGA to a dynamic trading scenario. The MTGA generates the portfolio at the start of a 12 month period, and the portfolio is rebalanced monthly using the local optimization method previously described. We compare this with another trading policy where the portfolio is re-generated from scratch, and with the market index. Their monthly return and difference amount are compared. The goal of this experiment is to establish whether the local optimization step of the Memetic algorithm can be used to rebalance portfolios as proposed.

5.1 Datasets

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

The NASDAQ data set contains 100 assets, and the S&P has 500 assets to chose from and compose a portfolio.

For each data set, the log return of the monthly closing value is used as the monthly return value of each stock. Values up to December 2008 are in the data sets, which were obtained from freely available on-line sources. Of special interest is the period starting from the later half of 2008, when the economic crash generated a large amount of instability in the asset values.

5.2 Parameters

We used the same parameters for all runs of evolutionary algorithms mentioned here. The number of generations was 500, with 200 individuals per generation. The crossover rate was 0.8 and the mutation rate was 0.03. The tree depth was set at 8, with a maximum of 128 terminals in a full tree. The riskless asset's return was set at 3% (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_thresh. Other than meme_thresh, which changes the precision of the search, changing these values does not seem to affect the quality of the local search.

5.3 Portfolio Generation experiment

In the first experiment, we generate portfolios for various scenarios in the two data sets and compare the results between the proposed method, the previous GA based method, and the benchmark index value. The goal is to analyze whether using the local optimization step to fine tune weights allow us to reliably produce better portfolios.

For each run of the MTGA and the TGA, we repeated the experiment 20 times with different random seeds. The results displayed here are the average of these 20 runs.

A summary of the results of this experiment can be seen on table 1. The Sharpe Ratio values of the Benchmark index, the TGA and the MTGA are listed, along with the TTest value between the TGA and the MTGA.

As can be seen in the table, both the MTGA and the TGA are able to beat the benchmark index, with the MTGA achieving consistently better scores than the TGA.

A closer analysis of the results show that both methods achieve the high Sharpe ratios by minimizing the risk, rather than increasing the return. In fact, while the return stays at an average of 0.4 to 0.6 for both methods, the very low risk values are responsible for the high increase in Sharpe ratio.

On the downside, the MTGA on average consumed computation time one order of magnitude above the TGA. Table 2 shows the average times for the two data sets. Since we are using a simple policy to decide which individuals undergo local optimization, it is expected that a cleverer policy might mitigate this somewhat.

Dataset	TGA	MTGA
NASDAQ	2 min.	10 min.
SP500	5 min.	50 min.

Table 2: Time expended for the Portfolio Generation Experiment (average)

5.4 Dynamic Trading experiment

In the second experiment, we try to generate and maintain a portfolio for a 12 month period. During this period, the portfolio is rebalanced monthly. We compare the rebalancing done by the local optimization step of the MTGA, with a policy of generating a new portfolio from scratch using the MTGA every month. The initial portfolio is generated using the MTGA in the same way described in the previous experiment.

We evaluate both approaches in two ways: first we compare their results with the actual returns from the market index for the period. This will show how stable the portfolio was. Then we compare the differences of the portfolios at each month in the one year period, to evaluate how much trading cost each portfolio would have garnered.

We present the results of three scenarios: NASDAQ 2007, NASDAQ 2008 and S&P 2007. The S&P in 2007 is a reasonably stable index, while the NASDAQ 2007 has a more pronounced variation. The NASDAQ 2008 scenario includes the September market crash (preceded by a lesser crash in March).

Figure 5.4 shows the return values for the portfolios rebalanced with local optimization and with re-evolution. Two things become clear at first glance. The first is that the return values for both portfolios are reasonably constant, even during the unstable period of 2008. The second is that the difference in terms of returns of the two rebalancing methods is rather small.

Investigating the data more closely we see the information on table 3. It shows that, although the return results were quite similar, the distance results where quite different. As expected, the Portfolio which was rebalanced by running the MTGA from scratch had much larger differences in the positions held each month than the portfolio which was rebalanced by the local optimization algorithm.

In particular, the result for the period of 2008 was particularly positive. In spite of the economic tension, reflected in the strong changes in return for the index, the MTGA managed to assemble a stable fund for the period.

6. DISCUSSION

While previous researches have shown the effectiveness of Genetic Algorithms applied for Portfolio optimization problems, our experiments have shown that, by extending the GA with a local optimization step, we were able to consistently improve the performance of the algorithm to generate Sharpe-Optimal portfolios.

This is because the local optimization step is able to finely tune the weights of the assets which were selected by the Genetic Algorithm. In the smaller NASDAQ100 market, the advantage of the Memetic Algorithm was consistent. For the larger S&P500 market, the results were more varied, but the MTGA stayed ahead.

	NASDAQ100, 2006					S&P500, 2007			
Date	Index	MTGA	TGA	TTest	Date	Index	MTGA	TGA	TTest
Jul	-1.85	85.43	76.42	0.42	Jul	-3.28	1416.94	456.74	0.09
Aug	0.44	28.5	22.88	0.09	Ago	-0.74	853.5	192.82	0
Sept	0.42	26.52	22.54	0	Sept	0.22	664.51	190.99	0.01
Oct	0.41	23.52	20.66	0.01	Oct	-0.64	527.49	176.56	0
Nov	0.08	33.57	27.37	0	Nov	-3.25	955.51	156.88	0
Dec	-1.25	10.98	10.00	0.11	Dec	-1.39	310.86	92.49	0

Table 1: Average Sharpe Ratio values for the Portfolio generation experiment.

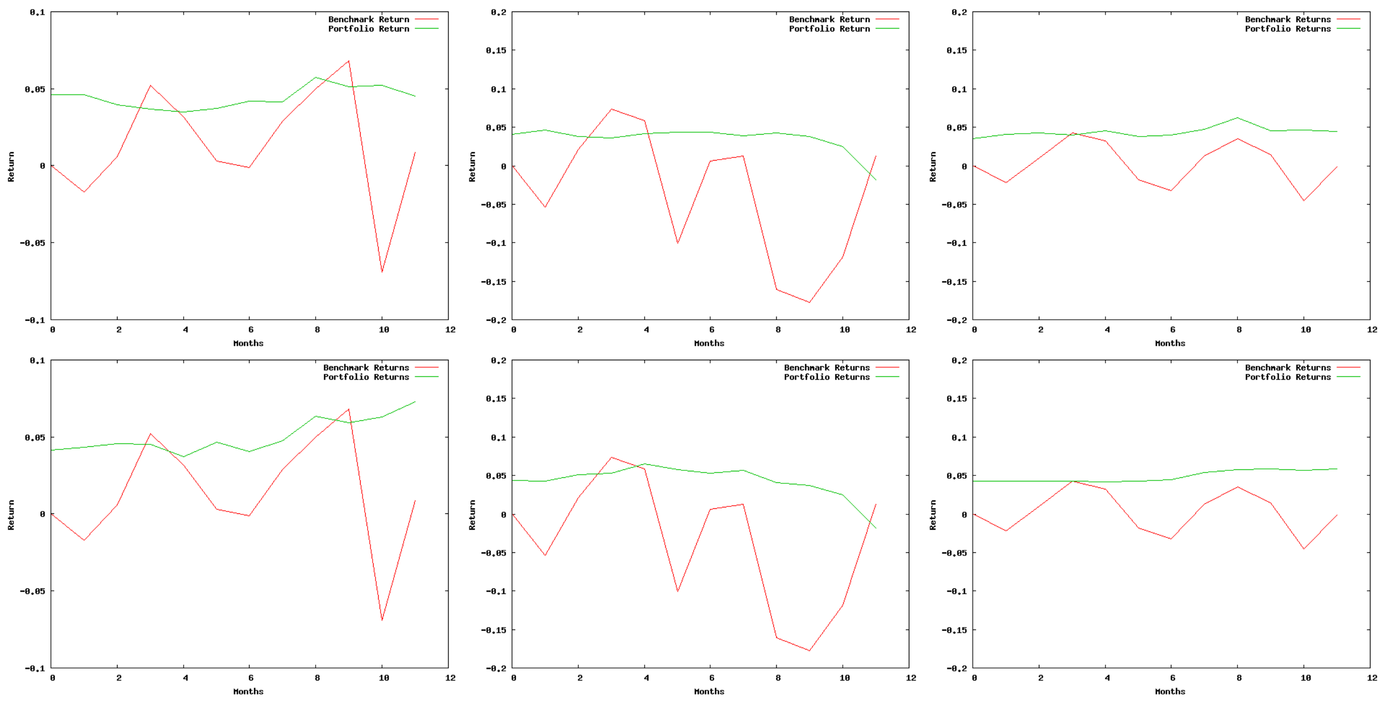

Figure 4: Result of the Rebalancing Experiment. On the top row, full re-evolution in the scenarios NASDAQ 2007, NASDAQ 2008 and S&P 2007. On the bottom row, the same scenarios as above, but the results come from using local optimization as the rebalancing technique.

Data set and Method	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sept	Oct	Nov	Dec
NASDAQ '07, Re-evolve	0	0.68	0.77	1.38	0.71	0.56	1.43	1.09	1.54	0.63	0.55	0.96
NASDAQ '07, Local Op.	0	0.32	0.34	0.98	0.27	1.16	1.13	1.16	1.16	0.18	0.25	0.06
NASDAQ '08, Re-evolve	0	1.80	1.00	0.52	1.01	0.62	0.87	0.44	0.36	1.24	1.86	0
NASDAQ '08, Local Op.	0	0.42	0.68	0.55	0.86	0.34	0.33	0.32	0.22	0.75	0.61	0
S&P '07, Re-evolve	0	1.73	1,07	1.54	1.84	1.34	1.72	1.88	1.76	1.73	1.48	1.84
S&P '07, Local Op	0	1.18	0.31	1.25	0.55	0.42	1.23	0.58	2.01	2	2	2

Table 3: Distance Between rebalanced Portfolios

However, this performance comes at a steep increase in computational cost. Implementing a more complex policy for deciding which individuals in a population will undergo local optimization is a promising idea, since the current policy is to pick an arbitrary number of individuals, chosen randomly.

Finally, on the dynamic trading scenario, it was observed that by applying the principles of adaptation to changes in the environment, the MTGA managed to keep a stable, low risk return, even during the period of Market instability in 2008. We observed that using the local optimization step allowed us to perform rebalancing with lower transaction costs, by taking advantage of the assets chosen by the full algorithm in the beginning of the period.

7. CONCLUSION

In this work we hybridized the Tree based Genetic Algorithm with a local search. The resulting MTGA was applied to the problems of Portfolio Selection, Optimization and Rebalancing.

Our results reinforce the idea that Memetic Algorithms are generally more suited to solve problems with real-valued representation than pure Genetic Algorithms. Also, the local optimization step of the hybrid can be used to perform small adaptations to changes in a dynamic environment.

Using these properties, we created a system which is able to select and optimize a large number of financial assets into a fund with stable return values, even under a situation of market volatility. In practice, this line of research, in particular rebalancing, is important for companies which manage large financial funds and must achieve a set level of return either in bull or bear markets. However, the MTGA could be used for other resource allocation problems which can be described as optimizing a large number of real valued weights. One example of such a problem is the evaluation of game board positions.

We observed that as the number of assets grows, the variation of the final results achieved grew as well. This is either because there are a large number of peaks in the search space as the number of assets increase, or because at very small variance values, a small change in the risk rate influences the final Sharpe result greatly. Investigating this matter further seems a promising venue.

Another approach which deserves more investigation is the interaction of the "intron" nodes with the local optimization. Does leaving inactive nodes inside a genome provides relevant information for the local optimization to do its job? Or is more gained by removing these nodes?

8. REFERENCES

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

[2] C. Aranha and H. Iba. A tree-based ga representation for the portfolio optimization problem. In *GECCO - Genetic and Evolutionary Computation Conference*, pages 873–880. ACM Press, July 2008.

[3] W. E. Hart. *Adaptive global optimization with local search*. PhD thesis, University of California at San Diego, La Jolla, CA, USA, 1994.

[4] R. Hochreiter. An evolutionary computation approach to scenario-based risk-return portfolio optimization for general risk measures. In M. G. et al., editor, *EvoWorkshops 2007*, number 4448 in LNCS, pages 199–207. Springer-Verlag, 2007.

[5] T. Ibaraki and N. Katoh. *Resource Allocation Problems - Algorithmic Approaches*. The MIT Press, 1988.

[6] R. Jiang and K. Y. Szeto. Discovering investment strategies in portfolio management: A genetic algorithm approach. In *Proceedings of the 9th International Conference on Neural Information Processing*, volume 3, pages 1206–1210, 2002.

[7] C.-M. Lin and M. Gen. An effective decision-based genetic algorithm approach to multiobjective portfolio optimization problem. *Applied Mathematical Sciences*, 1(5):201–210, 2007.

[8] P. Lipinski, K. Winczura, and J. Wojcik. Building risk-optimal portfolio using evolutionary strategies. In M. G. et al., editor, *EvoWorkshops 2007*, number 4448 in LNCS, pages 208–217. Springer-Verlag, 2007.

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

[10] S. ping Chen, C. Li, S.-H. Li, and X. wei Wu. Portfolio optimization with transaction costs. *Acta Mathematicae Applicatae Sinica*, 18(2):231–248, 2002.

[11] F. Streichert, H. Ulmer, and A. 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.

[12] H. Subramanian, S. Ramamoorthy, P. Stone, and B. J. Kuipers. Designing safe, profitable automated stock trading agents using evolutionary algorithms. In *GECCO 2006 - Genetic and Evolutionary Computation Conference*, pages 1777–1784, Seattle, Washington, July 2006. ACM Press.

[13] B. Ullah, R. Sarker, D. Cornforth, and C. Lokan. An agent-based memetic algorithm (ama) for solving constrained optimization problems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 999–1006, Singapore, September 2007.

[14] W. Yan and C. D. Clack. Behavioural gp diversity for dynamic environments: an application in hedge fund investment. In *GECCO 2006 - Genetic and Evolutionary Computation Conference*, pages 1817–1824, Seattle, Washington, July 2006. ACM Press.

[15] W. Yan and C. 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.

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