A Tree-based GA Representation For The Portfolio Optimization Problem

Claus C. Aranha
Institute of Electrical Engineering, The University of Tokyo
Tokyo, Japan
caranha@iba.k.u-tokyo.ac.jp

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

ABSTRACT

Recently, a number of works have been done on how to use Genetic Algorithms to solve the Portfolio Optimization problem, which is an instance of the Resource Allocation problem class. Almost all these works use a similar genomic representation of the portfolio: An array, either real, where each element represents the weight of an asset in the portfolio, or binary, where each element represents the presence or absence of an asset in the portfolio.

In this work, we explore a novel representation for this problem. We use a tree structure to represent a portfolio for the Genetic Algorithm. Intermediate nodes represent the weights, and the leaves represent the assets. We argue that while the Array representation has no internal structure, the Tree approach allows for the preservation of building blocks, and accelerates the evolution of a good solution. The initial experimental results support our opinions regarding this new genome representation. We believe that this approach can be used for other instances of Resource Allocation problems.

Categories and Subject Descriptors

I.1.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search

General Terms

Algorithms

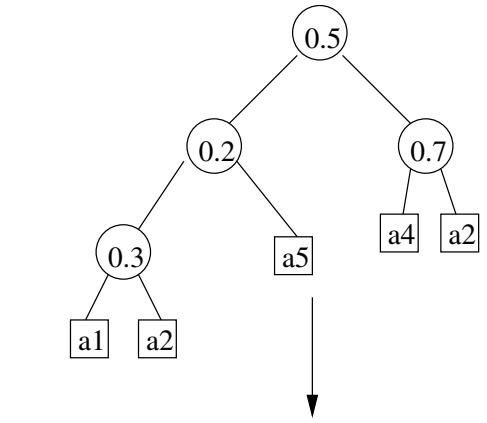
Keywords

Finance, Representation, Genetic Algorithm, Optimization

1. INTRODUCTION

In this work, we propose a new representation of candidate solutions for GA in the Portfolio Optimization problem. The Portfolio Optimization problem consists of the division of a fixed amount of Capital among a variety of assets in order to maximize the Estimated Return, and Minimize the Risk.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.



	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 subtree. The complement of that value is the weight of the right subtree. 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.

This problem belongs to the class of Resource Allocation Problem, where the Resource is the investment capital, the jobs are the assets, and the two utility functions mentioned are functions of the assets belonging to the portfolio and their weights.

In the last few years, many works have been published about the use of Genetic Algorithms for the Portfolio Optimization problem. These works concentrate either on how to adapt current Portfolio optimization techniques to real-world constraints [12, 7], or on how to deal with the multi-objective nature of the problem [2, 15, 12].

The majority of works that apply Genetic Algorithms to Portfolio Optimization seems to consider that the only way to represent a Portfolio is by using an array-based structure. There are different ways that this array is used: The most common one is to use a real valued array to hold the weights of all the assets belonging to the portfolio [8, 6, 3]. Another possible method is to use an array of binary values that state

whether a given asset belongs to the portfolio or not. A few works [2, 12] have used a hybrid approach, with one binary array and one real valued array.

We propose a radically different approach for genome representation. The portfolio is represented by a tree, inspired by Genetic Programming, where each terminal node holds one of the available assets, while each non-terminal node holds the weights of the two subtrees that branch from it (figure 1).

In this representation, each sub tree (including a subtree composed of only a terminal node) is also a complete portfolio tree, and can be evaluated by the same fitness function. This characteristic allows us to use the utility values of the sub trees to implement a crossover operator that takes into consideration the quality of a subtree when selecting the cutting point.

We show that a GA using this tree representation is able to find simpler candidate solutions than a GA using the traditional array representation. These simpler solutions possess similar utility values as their more complex counterparts, so among other advantages, they incur a smaller transaction cost in a multi scenario case.

This paper is organized as follows: In section 2, we give a more detailed description of the Portfolio Optimization problem. In section 3, we describe the Array based Genome Representation. In section 4 we present and discuss the Tree Representation. In section 5 we show experiments comparing the performance between these two representations, and then we discuss the results of these experiments.

2. PROBLEM DESCRIPTION

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 problem was first modeled by Markowitz [9]. This basic model, can be solved by Quadratic programming [16]. However, when adding real world constraints to the problem (eg. large number of assets, restrictions to the values of weights, trading costs, etc), the search space becomes too large, non-convex, and thus the problem becomes unsolvable by these methods. This is what motivates the use of GA to solve Portfolio Optimization problems in practice.

2.1 Markowitz Model

We 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 in the market (bonds, securities, currency, etc). These weights must obey two basic restrictions:

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

$$0 \leq w_i \leq 1 \tag{2}$$

In other words, the sum of all weights must be 1, and the weights must be positive. Since the problem with possible negative weights can be reduced to the version with only positive weights [16], we assume that the weights will be positive for simplicity.

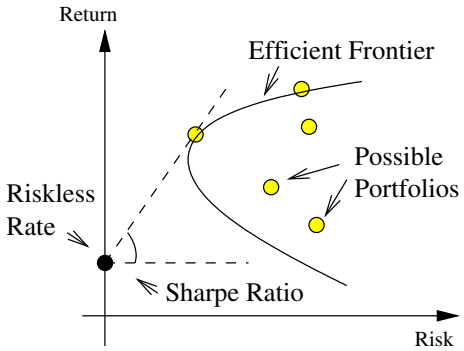


Figure 2: Risk-return projection of candidate portfolios. The search space is bounded by the Efficient Frontier. Sharpe ratio is the angle of the line between a portfolio and the risk-free rate.

Each asset has an expected return value, expressed by R_i. The expected return value for the portfolio is given by the sum of the expected return values for the assets that are part of that portfolio, as follows:

$$R_P = \sum_{i=0}^{N} R_i w_i \tag{3}$$

Also, each asset has a risk measure, σ_i. In the Markowitz model, the risk of an asset is defined as the variance of that asset's returns over time, and the risk of the Portfolio is defined as the covariance between its assets, as follows:

$$\sigma_P = \sum_{i=0}^{N} \sum_{j=0}^{N} \sigma_{ij} w_i w_j \tag{4}$$

Where $\sigma_{ij}, i \neq j$ is the covariance between i and j, and σ_{ii} is the variance 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[13] and Shu[11].

2.2 Fitness Measures

There are a number of fitness measures that can be used when applying GA to the Portfolio Optimization problem.

A popular fitness measure is the Sharpe Ratio [15, 13, 6]. It is defined as:

$$Sr = \frac{R_P - R_{riskless}}{\sigma_P} \tag{5}$$

Where $R_{riskless}$ is the risk-free rate, an asset with 0 risk and a fixed, low return rate (for example, government bonds of stable nations). The Sharpe Ratio expresses the trade-off between risk and return for a Portfolio given a fixed rate of return. A higher Sr value indicates a better Portfolio.

Another approach is to evaluate the Return and the Risk separately, and rank the candidate portfolios using some form of Multi-Objective Genetic Algorithm. Coello surveyed a number of works which use this approach [14]. Recently, however, it has been stated that MOGA may not perform as

well for this problem as using the Sharpe Ratio directly [15]. Whether or not this holds true is still an open problem.

2.3 Real World Constraints

The Markowitz Model, as described above, can be solved by optimization techniques such as Quadratic Programming [16]. 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 considered, and affect greatly the outcome of the optimization process. In our previous work [2], we have addressed the problem of how to reduce the difference between portfolios of consecutive scenarios to reduce trading cost. Our current proposal also addresses this concern, by removing from the candidate solutions assets that do not contribute to the final result, but increase the trading cost of the portfolio.

3. ARRAY REPRESENTATION

Most works on Portfolio Optimization using Evolutionary computing represent the Portfolio in the evolutionary system through the use of arrays [15, 5, 12].

We find three basic ways to represent a portfolio as an array for a genetic algorithm in the literature. In the first, each element i of the array is a real valued variable that represents the weight of the asset a_i in the portfolio. In the second, each element i is a Boolean variable that represents whether the asset a_i belongs to the portfolio or not. The third method is a combination of both previous methods, where we have two arrays, a real valued array, and a Boolean array.

In all three representations described above, the order of the assets in the array is arbitrary, usually alphanumeric by the name of the asset (see figure 3).

3.1 Operators

The crossover operator used is the Linear crossover, where for each element in the array there is an equal chance of copying the information from the first parent, the second parent, or mixing information from both parents.

Because the assets are aligned in the array in an arbitrary order, there is no difference between linear crossover and other methods that depend on partitioning the genome in certain ways, like N-point crossover. This is because the order of the assets can be re-arranged without any change in meaning to fit a particular partitioning.

The mutation operator consists of perturbing each element in the Array. Real valued elements will be perturbed within a normal distribution, and Boolean elements will be flipped. Each element in the array has an independent chance of being changed by the mutation.

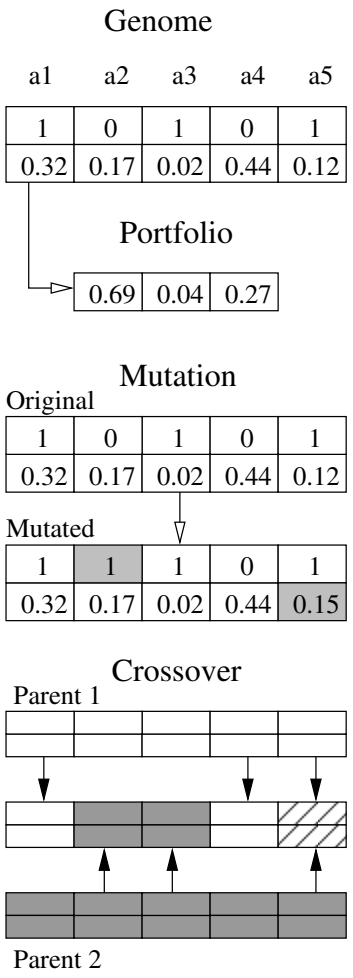


Figure 3: An array representation of a portfolio, the mutation and linear crossover operators

4. TREE REPRESENTATION

We propose the use of a Tree-based representation as an alternative to the Array based representation. Tree-based genome representations are often seen in Genetic Programming. However, they also have been used for GA based optimization problems. For example, Iba et al. has used tree structures to evolve optimal weights for a polynomial function [10].

Tree-based representations are useful when the the possible solutions to the problem being solved can be broken into sub-solutions which are also valid solutions to the problem. If this condition is satisfied, the sub-trees that compose an individual are also possible solutions, and can be evaluated.

For Financial Portfolios, a subset of assets of a portfolio is also a valid portfolio, after normalizing their weights. So it is possible to develop a tree representation which is composed of nested sub-portfolios.

The advantage of such representation is that the fitness function is applicable not only for the root node (whole individual), but for each sub tree as well. It becomes possible to use this information to effect better crossover and mutation operators.

4.1 Implementation

We currently implement the tree genome as a binary tree. Intuitively speaking, the root node of the tree points towards the total amount of capital. At each node, a real value between 0 and 1 determines how much of the capital that reached that node goes to the left sub tree and right sub tree. At the terminal nodes, all the capital that reached that node goes to the asset designed by the value of the terminal node.

Tree Structure

The non-terminal nodes represent the weights between its two sub trees. Each non-terminal node in the tree contains 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 that node has weight $1 - w$. Figure 1 shows this representation.

To extract the portfolio from the genome, we calculate the weight of each terminal node by multiplying the weights of all nodes that need to be visited to reach that terminal from the root of the tree (w if the sub tree branches to the left, $1 - w$ if to the right). Then the weights of terminal nodes that point to the same asset are added together. The assets which are not mentioned in the tree have a weight of 0 assigned to them.

Because of this structure (the sum of the weights of the two sub trees of any non terminal node is 1), extracting a portfolio by the method above will result in all weights being correctly normalized.

Also, this structure means that a portfolio containing all N available assets requires a tree with depth $\log_2 N$. For instance, for the NASDAQ100 market, which contains 100 assets, it will be needed a tree of depth 7.

Fitness Calculation

To calculate the return for each sub tree, we modify the calculation of the return of a tree N to become a recursive function where:

$$R(N) = wR(c_l) + (1 - w)R(c_r) \quad (6)$$

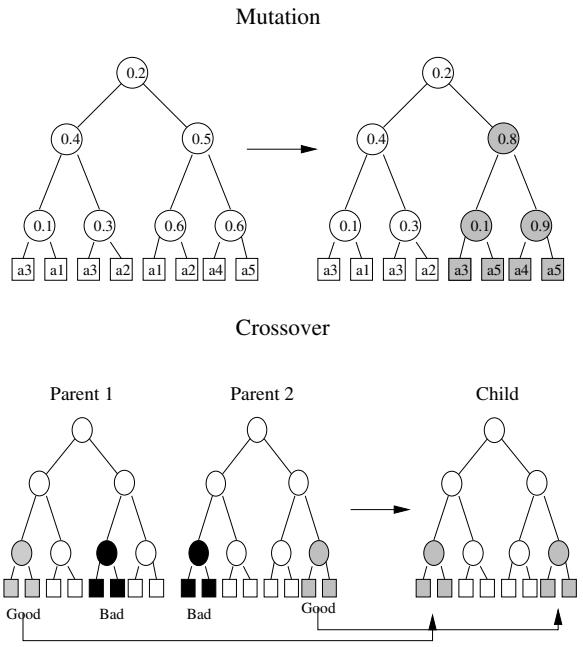


Figure 4: Crossover (BWS) and Mutation operators for the tree representation.

If N is a node Where c_l and c_r are its left and right children, respectively. And if N is a leaf (asset):

$$R(N) = MA(a_N) = \frac{\sum_{t=0}^{T-1} r_{(}a_N)^{t-T}}{T} \quad (7)$$

Where MA is the moving average, which is the average of the return value of the last T periods (T is parametric). This recursive function has the same complexity than iteratively calculating the estimated return for a full portfolio, so we can have the return value for the sub trees at no extra cost.

The calculation of the risk for each sub tree, on the other hand, becomes more expensive. The variance of the sum of two sub trees is given as:

$$\sigma(c_l + c_r) = w * \sigma(c_l) + (1-w)\sigma(c_r) + 2w * (1-w)Cov(c_l, c_r) \quad (8)$$

Where the risk of the children sub trees, $\sigma(c_l)$ and $\sigma(c_r)$ is given at the terminal node level by the data set, and is the result of the above calculation at each level. So the covariance between each sub tree needs to be re-calculated, and this increases the overhead of the system. To avoid unnecessary calculations, the covariance is only re-calculated in the sub trees that were modified, during mutation and crossover.

Genetic Operators

The Mutation operator works by cutting off a tree at a random point, and replacing it a new, random sub tree from that point. In this work we choose the cutoff point for the Mutation operator randomly. Alternatively, the cutoff point may be chosen probabilistically, with a chance proportional to the inverse of the fitness of a non-terminal node in the tree.

The Crossover operator works by exchanging sub trees between two individuals. One point at the same depth is

chosen for each tree, and the sub trees that start from that point on are swapped between the two trees.

We call it Simple Crossover if the choice of the crossover points is random, or based on data not related to the fitness (like depth). We define Guided Crossover as crossover policies that use fitness information from the sub trees of an individual to choose the crossover points in each parent.

For example, a simple Guided Crossover is the Best-Worst Sub tree Crossover (BWS). Here, a random depth is chosen, and the sub trees at that depth with the best and worst fitness values are identified. The sub tree with the worst fitness of the first parent is switched with the sub tree with the best fitness of the second parent.

Both operators are illustrated in figure 4.

4.2 Motivation

There are some benefits that we expect from the use of a three based genome, according to the above implementation.

Population Convergence

By choosing the crossover point by the performance of the sub trees with the BWS operator we expect the convergence rate of the system will improve. This is because the crossover between two good individuals will more consistently produce good individuals.

We demonstrate that the BWS crossover operator often generates better individuals by exchanging a bad sub tree for a good sub tree. Assume that the new sub tree has a higher fitness value than the old sub tree. This may not hold true if the fitness difference between the two individuals is too big (the best sub tree of one individual is worse than the worst sub tree of the other individual).

In this case, the New sub tree is A, the Old sub tree is B, the 'sister' sub tree is P, and the weight between P and A/B is w. If $f(S)$ is the fitness function of a sub tree S, we have $f(P) > f(B)$, because B is the worst sub tree, and $f(A) > f(B)$, by our initial assumption.

We want to know under what conditions $f(wP + (1 - w)A) > f(wP + (1 - w)B)$. If the fitness is the Sharpe Ratio, we can use the *two fund theorem* [16], which states that a linear combination of two portfolios results in a third portfolio which is in a line between the first two. So we know that the possible portfolios with the old sub tree B and the new sub tree A are in the segments $\overline{PB}$ and $\overline{PA}$.

Because the weight w does not change with the crossover operator, all the possible portfolio combinations $B' = wP + (1 - w)B$ and $A' = wP + (1 - w)A$ are parallel to the sub trees A and B. If $f(A) > f(B)$, then $f(B') > f(A')$ only when the intersection points $\overline{AB} \cap \overline{OP}$ and $\overline{A'B'} \cap \overline{OP}$ are located in different quadrants (see figure 5). So, replacing a sub tree with low fitness for a sub tree with high fitness will in most cases result in an individual with higher fitness.

Genome Structure

Another advantage of this implementation is that the evolution of the population will result in the development of an internal structure of the genome.

The crossover operator, over the course of generations, moves good sub trees of different sizes to the best individuals. If the sub trees are attached to other good sub trees, the block as a whole has a higher chance of being transmitted to an individual of higher fitness. So the evolutionary pressure acts not only on the individual level, but in the sub tree

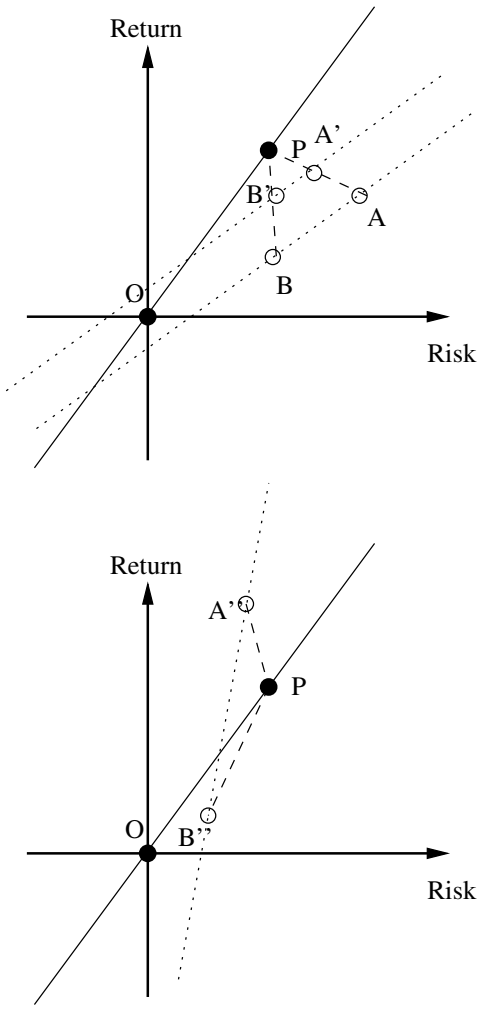


Figure 5: Conditions where it is and it is not possible to lower an individual's fitness by replacing a low fitness sub tree with a high fitness one. In the upper image, the tree PA may have a higher or lower fitness than PB depending on the weight, so we can see the parallel $\overline{AB}$ crossing $\overline{PO}$ at different quadrants. In the lower image, it is not possible to choose a weight w so that the portfolio will LOSE fitness by replacing a worse sub tree.

Table 1: Experiment Parameters

Name	Value
Generation	200
Population	400
Tournament K	10
Elite	5
Mutation Rate	0.02
Crossover Rate	0.8
Riskless Rate	0.02
Significant Asset Threshold	0.03

level as well. Some sub trees will eventually be seen more often in the successful individuals - we call these emergent sub trees the *building blocks* of the population.

It is not possible to observe such building blocks under the array representation, due to its arbitrary ordering. Also, it may be possible to extract information about the market and its assets by studying these building blocks (although this is not addressed in this work).

Vestigial Assets

One of the main differences that was observed during the experiments between portfolios evolved by array representation, and those evolved by tree representation, is that the later had a much lower average number of assets, for the same final result. When counting only the assets with weight above a certain threshold, the difference was much smaller.

What this tells us is that evolution using the above described tree representation tends to eliminate assets that do not contribute to the end result much more aggressively than the array based method. This is because while the BWS operator actively eliminates low fitness sub trees and their assets from individuals. In the array representation, on the other hand, if an asset with a minimal weight do not contribute either negatively or positively to the fitness, it is very hard to remove it from the individual. This requires either a random mutation, or a crossover with another individual that don't have that particular asset.

There are two main benefits of eliminating these minimal weight assets that do not contribute to the portfolio: 1) the resulting portfolio becomes clearer and more understandable, and 2) the Vestigial Assets do incur transaction costs in multi-scenario optimizations, and since they don't contribute to the result, these transaction costs are unneeded.

5. EXPERIMENTAL RESULTS

We performed a series of experiments to compare the performance of the tree based representation with the array based representation. The parameters used in the experiment follow in table 1. Of those parameters, the only one which is not familiar to users of EC is the last one, Significant Asset Threshold. That parameter is used to determine if a particular asset in a portfolio is significant or not. In the current experiment, we consider as relevant to the portfolio any asset that composes more than 3 percent of the total value of the portfolio.

A single scenario was used for each run, and no particular constraints were taken into consideration. The fitness function used was the Sharpe Ratio.

These reasoning behind these choices is that, in this experiment, we are more interested in observing the difference between the evolutive behavior of the GA with different representations, than trying to improve their performance in relation to the index (which is the goal of many other works about this problem).

For each scenario (1 month), the population was trained on the 12 month period previous to it, without further knowledge of their target environment. Each scenario was repeated 30 times with different random seeds - the results showed here are averages of these 30 runs. We ran a total of 72 scenarios, 36 from the NASDAQ100 data set (100 assets total), and 36 from the NIKKEI data set (225 assets total).

The results for this experiment were remarkably similar, independently of the scenario played. The Tree-based population started with a large number of assets (near 100% of the total available assets), but between 30 to 70 generations the number of assets falls down to almost the number of significant assets. The array-based population, on the other hand, starts at around 50% of the total available assets; quickly rises to 80-90% of the assets, and then slowly drops and stabilizes at 30-40% of the assets.

This pattern is illustrated on figure 6, taken from the NASDAQ Jan-2006 scenario. The exact same pattern can be seen in the other scenarios, with minor variations on the steepness of the curves, but not on the general plateaus.

The differences between the two methods are highlighted in table 2, which lists some representative results. The utility field is the return of the best portfolio in the population for that scenario. Those values are compared with the "Index" value, which represents an ideal portfolio according to Markowitz Model. The difference in utility between the tree representation and the array representation is small enough to say that they are equivalent in this regard.

The final number of significant assets is roughly the same for both methods in the NASDAQ market, and close in the Nikkei Market. On the other hand, the final number of total assets is much higher for the Array-based representation, which denotes its inability to remove from the portfolio genes that are no longer contributing to the solution.

6. CONCLUSION

We have proposed a new representation for the GA implementation of the Portfolio Optimization problem. In this approach, a tree based genome is used instead of the array based genome.

The tree based representation has advantages to the traditional array based representation, in that it allows the emergence of a structure by means of the BWS crossover operator. This operator uses fitness values of the sub trees of an individual to make an informed choice about the crossover point. Our experiments show that portfolios evolved using the tree representation have a lower number of significant assets, and a much lower total number of assets when compared to array based genomes, with the same utility value.

This result entails a number of practical benefits, such as reduced trading costs, ease to adapt results to trade constraints, and ability to understand the resulting portfolios.

Still, this is an early work on a novel approach to the Portfolio Optimization problem. In this work we analyzed some of the difference in results between the approaches (total number of assets, number of significant assets, final fitness value, convergence speed). We do not claim, however, to have exhausted the study of the differences and similarities between the two approaches.

Table 2: Representative Results

Scenario Name and Date	Utility			Average Assets		Significant Assets	
	Index	GA-Array	GA-Tree	GA-Array	GA-Tree	GA-Array	GA-Tree
NASDAQ 2004/Jan	0.023	0.018	0.017	*57.74*	*19.09*	9.66	9.65
NASDAQ 2005/Jan	0.077	0.035	0.057	*40.3*	*12.08*	9.93	7.91
NASDAQ 2006/Jan	0.021	0.049	0.074	*40.15*	*10.03*	11.81	7.79
NIKKEI 2004/Jan	0.004	0.002	0.003	*115.34*	*25.12*	17.5	15.83
NIKKEI 2005/Jan	-0.003	-0.012	-0.001	*143.01*	*32.10*	19.42	16.11
NIKKEI 2006/Jan	0.014	0.19	0.13	*85.31*	*23.78*	35.6	14.08

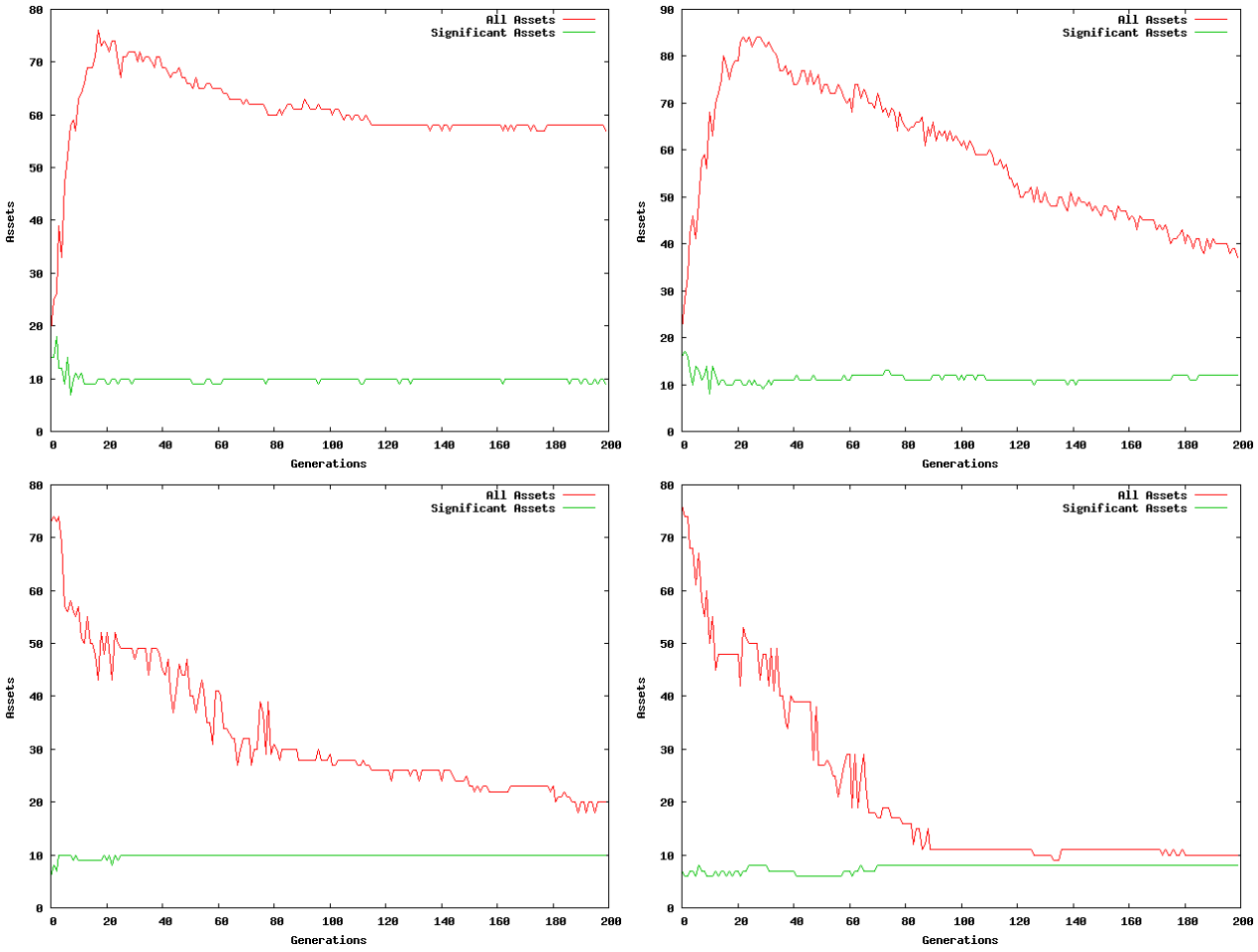


Figure 6: Average number of assets and significant assets when using array-based genomes (above) and tree-based genomes (below). The array genome is not able to eliminate superfluous assets. (The data in the chart refers to the 2004 scenario (left) and 2006 scenario (right)

There are more open questions about the comparative performance of both representations. Population diversity, performance in multi-scenario frameworks (addition of cost) and the addition of constraints are a few examples of such open questions, which we intend to address in the future.

7. REFERENCES

[1] C. Aranha. Portfolio management with cost model using multi objective genetic algorithms. Master's thesis, The University of Tokyo, 2007.

[2] 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.

[3] 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.

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

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

[6] 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.

[7] D. Lin, X. Li, and M. Li. A genetic algorithm for solving portfolio optimization problems with transaction costs and minimum transacton lots. *LNCS*, (3612):808–811, 2005.

[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] N. Y. Nikolaev and H. Iba. Regularization approach to inductive genetic programming. *IEEE Transactions on evolutionary computation*, 5(4):359–375, August 2001.

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

[12] 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.

[13] 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.

[14] M. G. C. Tapia and C. A. C. Coello. Application of multi-objective evolutionary algorihms in economics and finance: A survey. In *Proceedings of the Conference on Evolutionary Computation*, pages 532–539, 2007.

[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.