2J2-3

A New GA representation for the Portfolio Optimization Problem

Claus C. Aranha*1 Hitoshi Iba*2

The University of Tokyo, Department of Eletrical Engineering and Information Systems

We introduce a new representation for candidate solutions in Genetic Algorithms applied to Resource Allocation problems. The common approach for this problem is an array-based representation. Our proposal is the use of a binary tree to represent individual solutions, which possess an innate structure that allows the implementation of smart genetic operators. We apply this technique to the Portfolio Optimization problem, in which a finite capital must be divided among a number of financial assets with the goals of minimizing the risk and maximizing the estimated return measures. Our experiments show that the use of tree based genomes results in solutions that are less complex, while maintaining the same utility value, than those generated by traditional techniques.

1. Introduction

Genetic Algorithms (GA) is a random search heuristic that is based on the evolutionary process of living beings. In a GA, we encode the solution to the problem that we want to solve as a "genome", generate many of those solutions, test them according to a given heuristic (the fitness function), and generate new candidates from the best ranked "parents". By repeating this process, the algorithm is able to find good solutions to a wide range of practical problems.

We propose a new way to represent candidate solutions for the Portfolio Optimization problem. This problem consists of a fixed ammount of financial capital that must be divided among a variety of assets so as to maximize the Estimated Return, and minimize the Risk. It has been shown that GA is appropriate to solve this problem, and many works have been published on this subject [Lin 05, Aranha 07, Yan 07].

While these works concentrate on choosing good fitness functions, or adding new real-life constraints to the problem model, all of them use a traditional array-based structure to represent candidate solutions as genomes in the GA. The weakness of this representation is that the array possess no structure, so it is not possible for the algorithm to learn which assets are important or not to a given scenario.

Our approach uses a tree-based representation, where the terminal nodes are the assets, and the non-terminal nodes indicate the relationship between these assets. So the Genetic Algorithm is also able to evaluate, indirectly, whether the candidate tree represents an accurate relationship between assets - in other words, the structure of the portfolio emerges in the population.

Another advantage of the tree structure is that it is possible to use the fitness function of the whole tree to evaluate any subtree. This information allows us to indentify the building blocks in an individual (parts of the tree that contribute most to the utility value), and to design crossover operators that maintain these building blocks.

Contact:
caranha@iba.k.u-tokyo.ac.jp; 03-5841-8436
iba@iba.k.u-tokyo.ac.jp; 03-5841-7424

In this works, we execute comparative experiments between the array structure and the tree structure, using historical data from the NASDAQ and NIKKEI markets. The results of these experiments show that the GA with tree-based genomes produces portfolios that are simpler than traditional array based GA.

The tree-based solutions possess fewer assets that do not contribute to the portfolio (have a very low weight). This results in a lower trading cost. Finally, we believe that these results can be applied to other, similar resource allocation problems.

2. Problem Description

The mathematical model of the Portfolio Optimization problem is due to Markowitz [Markowitz 87]. 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. The weights obey the following:

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

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

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:

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

Also, each asset has a risk measure, σ_i. 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:

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

Where $\sigma_{ij}, i \neq j$ is the covariance between i and j, and σ_{ii} is the variance of asset i.

From the definitions of the Risk rate and the Estimated Return, we can derive a Utility function to use as Fitness for

the Genetic Algorithm. The most commonly used Fitness function is the Sharpe Ratio, which is defined as:

$$Sr = \frac{R_P - R_{riskless}}{\sigma_P} \quad (5)$$

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

This basic model for the Portfolio Optimization Problem, as formulated by Markowitz, can be solved by Quadratic programming [Yuh-Dauh-Lyu 02]. However, when adding real world constraints to the problem (large number of assets, restrictions to the values of weights, trading costs, etc), the search space becomes non-convex, and the problem becomes unsolvable by these methods. This is what motivates the use of GA in this problem.

3. Proposal

Traditionally, arrays have been used to represent solution candidates in GAs. For the Portfolio problem a real-valued array, where each element corresponds to the weight of one security in the market, is used. The order of the assets is arbitrary, and linear crossover is usually applied.

However, the array representation has some limitations. It is not possible to assign any structure or relationship to assets and their weights inside an array. Since the choice of portfolio depends on the relationships between assets, some information is lost when converting a portfolio to a simple array.

In this work we propose the use of a Tree-based representation for the solution candidates of the Portfolio Optimization problem. This idea is based on previous GP works, where a weigth tree was used to develop polynomials for system analysis [Nikolaev 01].

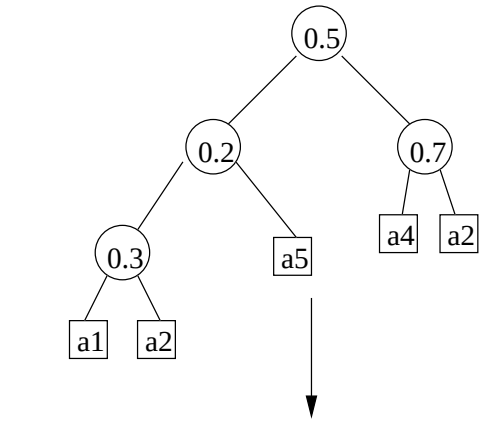
The advantage of such representation is that the fitness function is applicable not only for the whole individual (main tree), but for partitions of the solution as well (sub trees). This allow us to create informed crossover and mutation operators, and to extract structure information from the individuals.

3.1 Tree Structure

A candidate solution is implemented as a binary tree. Each terminal node correspond to an asset in the problem (one asset may appear in more than one node). Each non-terminal node contains a real value between 0 and 1, w, which represent the relative weights between its left child, and right child. The left child has weight w, and the right child has weight $w - 1$ (see figure 1).

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 (w if left, $1 - w$ if right). This can be computed efficiently by dynamic programming.

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.



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

図 1: A tree genome and its corresponding portfolio. The terminal nodes represent assets in the portfolio, and the non-terminal nodes represent the relative weight between the assets.

3.2 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:

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

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 leaf 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 only in the sub trees that were modified, during mutation and crossover.

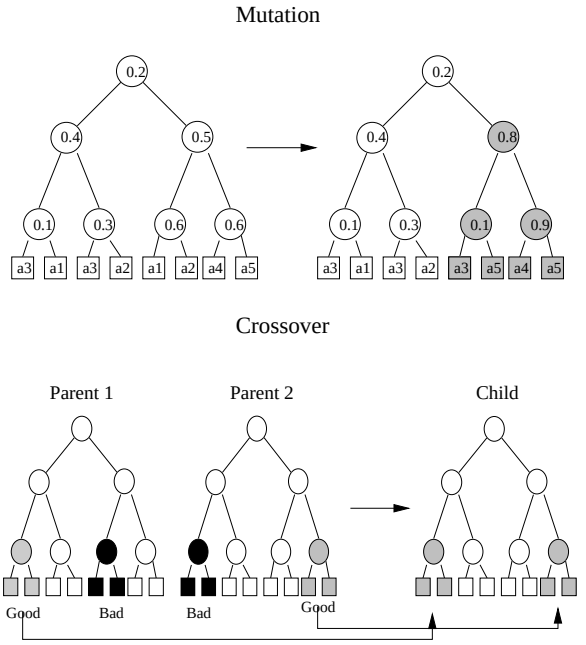


図 2: Crossover (BWS) and Mutation operators for the tree representation.

3.3 Genetic Operators

In a Genetic Algorithm, genetic operators are functions used to recombine the candidate solutions with the highest fitness values in order to generate new candidate solutions. There are usually two kinds of operators, the Mutation operator, which modifies a single candidate, and the Crossover operator, which recombines two candidates.

The Mutation operator works by cutting a tree at a random node, and generating a new sub tree from that node. The cutoff node is chosen randomly. The role of the mutation operator is to introduce new genes into the population, like new terminals or new sub-tree structures. It corresponds to the "explorarion" policy of a learning algorithm.

The Crossover operator works by exchanging sub trees between two individuals. One point at a given depth is chosen for each tree, and the sub trees that start from those nodes are swapped. It corresponds to the "exploitation" policy of a learning algorithm

We call the operator *Simple Crossover* if the choice of the crossover nodes is random, or based on data not related to the fitness (like depth). We also define the *Guided Crossover*, which uses fitness information from the sub trees of an individual to choose the crossover nodes 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 2.

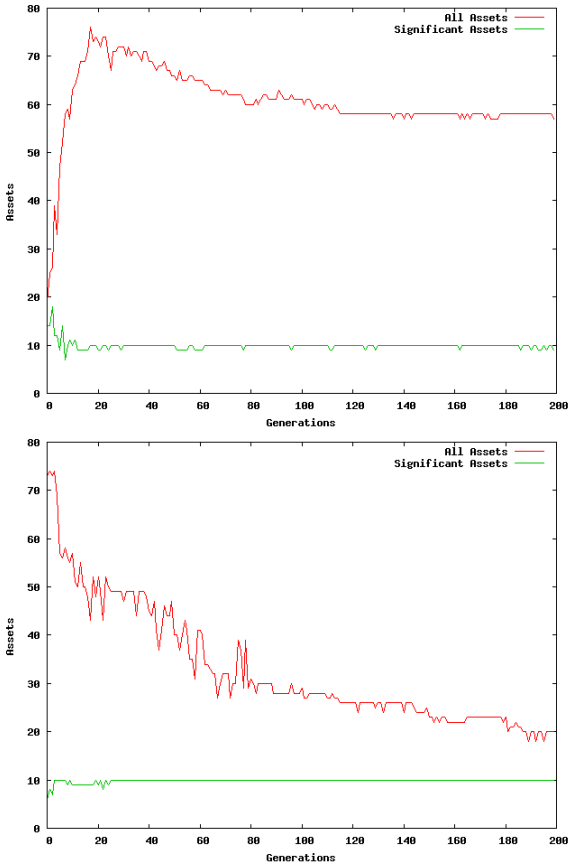


図 3: Average number of assets and significant assets when using array-based genomes (above) and tree-based genomes (below). The array genome is not able to eliminate superfluous assets.

4. Experiments

We performed a series of experiments to compare the performance of the tree based representation with the array based representation. As the main goal of the experiments is to observe the differences in the behavior of the GA with each representation, a single scenario framework was used, and no particular constraints were taken into consideration.

For each scenario (1 month) the experiment 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 and dataset. The Tree-based population started with a large number of assets (near 100% of the total available assets), but after 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 drops and stabilizes at 30-40% of the assets.

This pattern is illustrated on figure 3, taken from the NASDAQ Jan-2004 scenario. The exact same pattern can be seen in all other scenarios, with minor variations on the

表 1: Representative Results

Scenario Name	Utility			Average Ass ets		Significant Assets	
and Date	Index	GA-Array	GA-Tree	GA-Array	GA-Tree	GA-Array	GA-Tree
hline 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

steepness of the curves, but not on the general plateaus.

In practice, this means that the optimal portfolios generated by traditional GA usually contained the majority of the assets in the market, with minimal weights (less than 0.3%), and a few assets with large weights that composed the "body" of the portfolio. Because the crossover operator in an Array GA cannot eliminate an individual asset (no structure), these insignificant assets get passed down the generations, if they don't contribute or hamper the fitness function.

In the tree-based GA, on the other hand, the guided crossover operator eliminates subtrees which do not contribute to the utility value of the portfolio. The practical result is that the optimal portfolios for the tree-based GA are very similar to those of the array-based GA in regard to the main assets, but the insignificant assets are not present in these solutions.

Quantitative differences between the two methods are highlighted in table 1. Significant assets means assets in the Portfolio that have a contribution larger than 3% to the total value of the portfolio.

We can see that the number of significant assets is roughly the same for both methods. 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.

5. Discussion

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 such that it allows the emergence of structure by means of the BWS crossover operator.

Our experimental results showed that by using a Tree representation instead of an Array representation, the GA manages to generate simpler results, which contain only the assets really needed for the Portfolio.

This result is important to real-world applications, as one of the most limiting constraints in this problem is the Trading Cost [Aranha 07]. If two portfolios have the same return, the smallest of them will have a smaller trading cost when entering and leaving the position.

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). 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.

参考文献

[Aranha 07] Aranha, C. and Iba, H.: 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)

[Lin 05] Lin, D., Li, X., and Li, M.: A Genetic Algorithm for Solving Portfolio Optimization Problems with Transaction Costs and Minimum Transacton Lots, *LNCS*, No. 3612, pp. 808–811 (2005)

[Markowitz 87] Markowitz, H.: *Mean-Variance analysis in Portfolio Choice and Capital Market*, Basil Blackwell, New York (1987)

[Nikolaev 01] Nikolaev, N. Y. and Iba, H.: Regularization Approach to Inductive Genetic Programming, *IEEE Transactions on evolutionary computation*, Vol. 5, No. 4, pp. 359–375 (2001)

[Yan 07] Yan, W. and Clack, C. D.: Evolving Robust GP Solutions for Hedge Fund Stock Selection in Emerging Markets, in *GECCO 2007 - Genetic and Evolutionary Computation Conference*, London, England (2007), ACM Press

[Yuh-Dauh-Lyu 02] Yuh-Dauh-Lyu, : *Financial Engineering and Computation*, Cambridge Press (2002)