人工知能学会研究会資料
SIG-FIN-003-10

遺伝的アルゴリズムを用いたポートフォリオ最適化の最新発達
Advances in Portfolio Optimization by Genetic Algorithms

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: Porfolio Optimization is an important problem for financial engineering. It consists of finding out the best investment weights for a large group of assets, so that the Expected Return of those assets is maximized and the specific risk of the portfolio is minimized. This problem can be modeled as a Parameter Optimization problem, and Genetic Algorithms have shown better results every year. In this paper we review recently proposed techniques to optimize Financial Portfolio using Historical Price values, compare them, and draw up proposals about how to improve these results even further.

1 Introduction

The Portfolio Optimization problem consists of dividing an amount of capital between multiple assets in order to maximize the return, and minimize the risk of the investment.

Investment Portfolios are used by financial institutions in the management of long term investments, like savings accounts, retirement funds, etc. When real life large data sets and constraints are added, though, this becomes a tough problem that cannot be solved by numerical methods.

Because of this, the use of computational heuristics like neural networks and evolutionary algorithms has been an active topic of research. In particular, Genetic Algorithms is one of the most popular approaches recently. This popularity is partly because it is very easy to represent a Portfolio as a real valued array, and use that array as the genome in the Genetic Algorithm.

In the recent years, we have seen an amazing number of ideas and projects regarding the use of Evolutionary Algorithms for the Portfolio Optimization problem. In this paper we will take a closer look at the most significant works published in the previous two years, drawing some comparisons and seeing where they complement each other, and where extra work is still needed.

The goal of this work is to allow the portfolio researcher to identify new paths in need of study, and for the industry expert to get up to speed with the state-of-the-art techniques for this important financial problem.

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 [8].

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 [10]. Markowitz's Portfolio Model could be solved by numerical methods, like Quadratic Programming [15].

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 unsolvable by numerical methods. This is what motivates the use of Search heuristics like Evolutionary Computation to solve Portfolio Optimization problems in real world conditions.

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 [15]: 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:

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

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

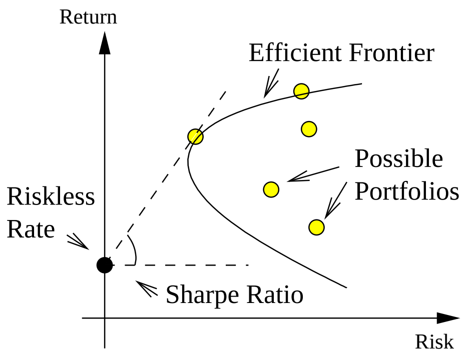


Figure 1: Risk-return projection of candidate portfolios. The search space is bounded by the Efficient Frontier. Sharpe ratio is the angle of the line between a portfolio and the risk-free rate.

Where 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:

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

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[14] and Shu[12].

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 1.

2.2 Constraints

The Portfolio Optimization problem, as modelled by Markowitz above, can be solved by deterministic numerical methods with a small number of assets. However, in real life applications, there are a number of constraints that must be applied to a feasible solution. When all the constraints are added to the Portfolio Optimization problem, the search space becomes concave and discontinuous, making the problem much more difficult, and reinforcing the importance of heuristic methods to solve it.

Among the constraints applicable to the portfolio Optimization problem we find cardinality constraints, lots, trading costs and trading volume.

Cardinality constraints are policy held by the investors which determine the maximum and minimum number of assets that a portfolio must be composed off. Portfolios with a number of assets above or below these limits are not valid.

Lots is a more interesting constraint. While the weights in the portfolio representation in Markowitz model can be any real number, in practice stocks are sold in bundles with a definite size, called 'lots'. When assembling a portfolio, the indivisibility of lots means that the exact value dictated by the optimized weight cannot be reached, and some sort of compromise must be found.

The trading costs constraint reflects the fees that you have to pay when buying or selling a security. They are more important in dynamic scenarios, where the existance of trading costs means that when deciding the optimal portfolio, the system must think not only in the current situation of the market, but also the position previously held. A new portfolio that differs too much from the previous one will incur great trading costs.

A closely related concept to trading costs is the limitations on trading volume. This constraints states that only up to a certain volume of a security can be bought or sold in a single trade. This adds a time limit to the cost limit of the previous constraint.

3 Recent Approaches to Portfolio Optimization

As described in the introduction, a superficial read of Markowitz's definition of the Portfolio Optimization problem will suggest a simple implementation for the model in Genetic Algorithm, where the genes correspond to the real valued weights. In fact, the first efforts to apply evolutionary computation to this problem followed this line.

However, recent advances have shown much more sophisticated gene representations and dedicated operands. The modern approaches come mainly in three categories.

The first are the "dual array" models, which solve the problems with the naive implementation of a weight array by adding an index array in the gene representation. The second category are the "Stock Selection" works, which start from the premise that the weights themselves are not as important as the choice of securities to compose the portfolio. Based on this premise, their strategy focuses almost exclusively in ranking the available stocks.

The last kind of approach are those who don't fit in the two main groups above. Usually they use some sort of evolutive network implementation to achieve fine control of the stocks that participate in the portfolio.

3.1 Genetic Programming Network

A Genetic Programming Network is an structure composed by *Judgement Nodes* and *Action Nodes*, which are linked in a neural network fashion. The GPN works by starting at some pre-determined starting location, and then running through the judgement and action nodes.

Chen et al. [3] used the concept of a GPN to generate an automated trader which used technical data to take portfolio forming decisions. In this GPN, when the operator passed by a judgement node, it would acquire extra information about the market state. When it passed by an action node, it would perform a buy/sell action based on the information acquired so far.

Chen also modified the original GPN with "control nodes", which guarantee that most GPN nodes are visited by the operator.

Chen's approach is quite unique among the other works studied here, in that it does not rely in an static "study, predict, result" cycle - instead, information gathering and acting are done simultaneously.

3.2 Technical Index Evolution

Some recent works have tried an indirect approach to the problem of rankign assets. Horta et al. [5] use genetic algorithms to optimize parameters which are utilized in the calculation of the technical indexes.

These parameter and indexes, then, are used for assigning rank values to each security within the market, and the best ranked securities are inserted into the portfolio. The GA also contains the relative weights of the different indexes when calculating the rank value.

In their current work, which of the technical indexes will be used are defined from the beginning. Future work from the current position would be to use of Genetic Programming to assemble the combinations with greater ease.

Another researcher who is using Technical Index Evolutio is Lumanpauw [9]. Lumanpauw developed a rather complex system divided into two stages. The first stage utilizes fuzzy neural networks to calculate the estimated returns of the assets, and this information is used to rank the available assets. In association to the fuzzy neural network, Lumanpauw uses a local search to optimize the weights of the chosen assets.

3.3 Vector based MOGA

Instead of trying to develop a new and advanced algorithm to approach the problem, Skolpadungket instead concentrating in attending to the needs of the real world [13]. In his paper, Skolpadunkget describes in good detail how he adapted the array approach to attent to all the above described constraints.

Basically, his method constitutes of two arrays, the first one a binary array that indicates if a certain security is part of the portfolio or not. And the second array indicates the evolved weights for the security.

To conform with the problem constraints, Skolpadungkat developed a series of "repair functions", which modifie the individuals in the population, so that all of them obey the limits set up by the traders.

Chiam and Mamum also published a work in Vector Based MOGA for portfolio optimization [4]. In their work, to avoid having to change the candidates so much for the "fixing function". The new genome representation for them has an index array with a fixed size, in which for each weight, a number representing the index of the security associated with the signal. Their tested their algorithm with a large stock simulation.

3.4 Factor Model

Factor Model is the name given by Clack and colleagues to their approach to Portfolio Optimization. Basically, it consists of the evolution of a formula to calculate the rank of an asset using Genetic Programming. In this solution, the leaves of the GP are structural information about the company behind the stock, and many techncal indexes.

While they have achieved great results using the above method, Petel and Clack [11] have tried to use the ALPS framework to improve the diversity of the individuals generated by their factor Model. ALPS is a population policy where individuals in a population may only crossover with other individuals with a similar age. They managed to show that not only diversity was preserved, but also the quality of the result improved.

Hassan and Clack [6, 7], on the other hand, concentrated on keeping diversity on the clusters that form when a population in a multi objective problem spreads over the Pareto line.

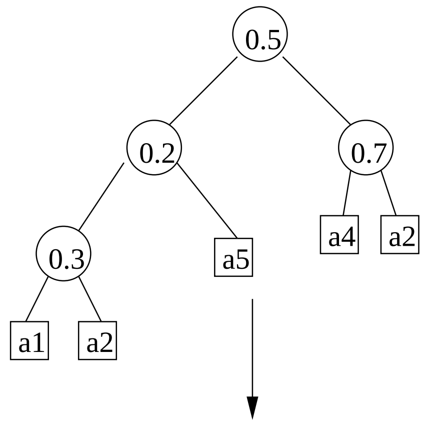
Hassan was mainly concerned with the fact that, while in the training dataset the individuals in the population would crowd the Pareto set, during the testing stage these individuals would migrate through the pareto front, gathering in only a few locations. To solve this, she instituted a new policy where individuals in a Pareto cluster could only mate with other individuals in the same cluster.

The results were that individuals that fell in a certain region during the training period, did not move around so much with relation to the other individuals when going to the test period.

3.5 Memetic Tree-based GA

The MTGA was proposed by Aranha and Iba in [2] in response to shortcomings in the array based methods. 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



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

Figure 2: 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.

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.

3.5.1 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 2 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.

3.5.2 Evolutionary Operators

The tree representation for an individual's genetic material in the MTGA requires the redesign of the basic evolutionary operators (crossover and mutation), but it also allow the development of new operators that use the unique characteristics of the tree representation.

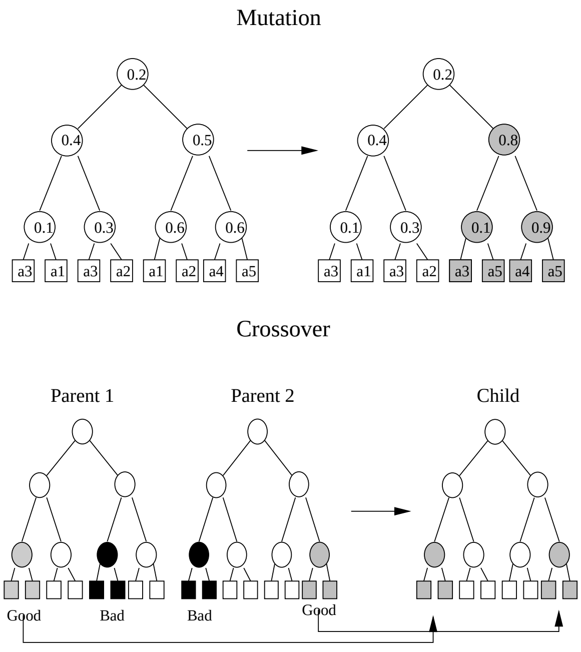


Figure 3: Crossover (BWS) and Mutation operators for the tree representation.

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.

If the crossover point is chosen at random, the operator is called *Simple Crossover*. Like in the mutation operator, in the simple crossover a depth is chosen with linear probability, and a path is randomly followed from the tree until the target depth.

3.5.3 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 3.5.3.

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*

```
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

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).

The main use of this operator is to improve the weights of the individuals during the evolutionary run. A second utility of this operator, though, is to rebalance the portfolio between scenarios. When we use a portfolio through a long period of time (multiple scenarios), it is necessary to adjust the portfolio to changes in the market [1].

By repeating the use of local-search operator after a change of market scenario, we can adapt the weights between the assets to the new conditions of the market. Because the the assets in the portfolio does not change, we achieve lower costs than if we started the evolutionary system anew for each scenario.

4 The Rebalancing Problem

An active field of study among researchers in the Financial Engineering field is the problem of Portfolio Rebalancing.

Most of the works cited and described here only actuate on *static* portfolios. The static portfolio problem is only concerned with the optimal portfolio at the moment. However, when making investments in the real market, we are more worried with *dynamic* portfolios.

In Dynamic Portfolios, the trader already has a position in his hand, and he wants to change to a more advantageous position. However, if he changes his position too much, the trading costs will be too high. So the movements from the first portfolio to the next must either be small, or be spread over time.

Also, after the investor has his new optimal portfolio, the system must be able to detect that the market has changed beyond the predicted amount, and thus the portfolio needs to be re-balanced again.

4.1 Pareto Front Management

Hassan and Clack's management of the Pareto Front and the cluster of solutions in certain regions of the pareto front is an example of one of the current researches that are being developed on this matter.

By stabilizing the movement of a individual during the training and test data, so that the individual does not change clusters between the two occasions, the system increases the ability of the solutions to degrade gracefully, which increases the reliability of the system in situations of market stress.

4.2 Dynamic Memetic Optimization

The MTGA previously discussed includes a local search operator that can be used to change the internal weights on the structure based in small changes in the market. While the MTGA does not possess the same "graceful degradation" propeties as the previous system, experimental results show that by using the local search operator to update the weights monthly results in a very stable return of investment.

5 Conclusion

Research on the Portfolio Optimization problem, as modelled by Markowitz and even with a number of real life constraints, has been developed a lot in recent years, and some amount of cross-polination between the research efforts can already be observed.

While the methods to determine which are the best securities inside a market, and given those securities, what are the optimal weights to secure a high return and low risk, are well known and developed, there are new questions and research directions that can be found when you step back and look at all the research effort that has been done as a group.

The first such direction has already been identified before. It is the research of good methods for the solution of the rebalancing problem. The rebalancing problem requires a higher level of intelligence from the systems, where the past and future positions are taken into consideration, and a feeling for when it is the right time to update your portfolio is needed. This is an area that is receiving some early attention from the community.

The second research direction that can be seen when observing all the work that has been done so far, is the integration between the weight optimizers and the factor model systems. It is quite obvious that the biggest divide in the works in this field is in whether one should concentrate on choosing the best stocks, or finely tuning the weight between those stocks. However, this is a false dicothomy - both problems can be tackled at the same time, by mixing the specific parts from algorithms from both sides of the divide.

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*, pp. 873–880. ACM Press, July 2008.

[3] Y. Chen, S. Mabu, K. Shimada, and K. Hirasawa. Construction of portfolio optimization system using genetic network programming with control nodes. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pp. 1693–1694, 2009.

[4] SC Chiam, A Al Mamun, and YL Low. A realistic approach to evolutionary multiobjective portfolio optimization. *IEEE Congress on Evolutionary Computation, 2007.*, pp. 204–211, 2007.

[5] António Gorgulho, Rui Neves, and Nuno Horta. Using gas to balance technical indicators on stock picking for financial portfolio composition. In *Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference - GECCO '09*, p. 2041, New York, New York, USA, 2009. ACM Press.

[6] Ghada Hassan and Christopher D. Clack. Multiobjective robustness for portfolio optimization in volatile environments. In *GECCO'08*, pp. 1507–1514, Atlanta, Georgia, 2008.

[7] Ghada Hassan and Christopher D. Clack. Robustness of multiple objective gp stock-picking in unstable financial markets. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO '09*, p. 1513, New York, New York, USA, 2009. ACM Press.

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

[9] E. Lumanpauw, M. Pasquier, and C. Quek. Mnfs-fpm: A novel memetic neuro-fuzzy system based financial portfolio management. In *IEEE Congress on Evolutionary Computation, 2007. CEC 2007*, pp. 2554–2561, 2007.

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

[11] S. Patel and C. D. Clack. Alps evaluation in financial portfolio optimisation. *2007 IEEE Congress on Evolutionary Computation*, pp. 813–819, September 2007.

[12] 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.

[13] P. Skolpadungket, K. Dahal, and N. Harnpornchai. Portfolio optimization using multi-obj ective genetic algorithms. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 516–523, 2007.

[14] 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.

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