Intelligent Computation Applied to Mutual Fund Investment:
A Genetic Algorithms Approach

Nuno Lopes
Instituto Superior Técnico
Lisboa, Portugal
nuno.m.lopes@ist.utl.pt

Abstract—This dissertation presents a study of the portfolio management problem applied to mutual fund investments. An approach based on Genetic Algorithms is followed and three different heuristics are proposed to forecast the rise or fall of a mutual fund. These heuristics are based in the concepts of daily variations, relevant maximums and minimums and simple moving averages. A prototype developed in MATLAB is used for the testing and comparing of the heuristics. All three heuristics have outperformed the Buy-and-Hold strategy in the tests performed.

I. INTRODUCTION

The arrival of the World Wide Web provided a platform for the birth, years later, of a new paradigm of investment: e-investment. No longer limited by the constraints of accessing daily market information, which can now be browsed online in mere seconds, and with the possibility of managing their investments online provided by a great number of homebanking services, the number of small private investors has increased substantially in the recent years.

Mutual Funds are currently one of the most sought after financial products. According to the Investment Company Institute [1], in the last quarter of 2007, the total value of Mutual Fund assets worldwide has reached $26.20 trillion, with a net cash flow of $383 billion in that quarter alone.

As the number of investors in these products increases every day, so does the number of investors with less than sufficient information to make logical and supported investment decisions. For an initiate in the financial markets, the first months, or even years, are a daunting experience. Although there are thousands of books and online information about financial markets that anyone can benefit from, including specific information for new Mutual Fund investors [4], it is extremely hard for a beginner to apply the abstract concepts to the practical cases. With very little experience, often the learning process for a new investor is based on trial and error, with an emphasis on error. The help of an application that provides advice on actual mutual funds would be very valuable for any new investor. Even if such an application cannot possibly be correct 100% of the times, as long as the advice provided is helpful most of the times, it is a definite progress for those taking the first steps in this area.

In the opposite side of the investor spectrum, the expert investors are more confident in their assessments of the market and consequent choices. However, studies have shown that intuitive reasoning is often flawed [2] and most of these investors make their decisions based on subjective preferences and biased assumptions rather than logical conclusions [3]. Although an expert would not blindly follow the advice of a machine unless it was proven to be correct, an application that can process the huge amounts of numerical information related to a group of mutual funds, make a logical evaluation of that data and provide advice based on it, can always be used as an additional tool for any expert investor, even if only to increase or decrease his amount of certainty in his own conclusions.

II. STATE-OF-THE-ART

The Fund Manager system [5], from Beiley Software company, was first launched in 1993 and has since then increased in functionality in order to provide a very comprehensive tool for portfolio management. It can be used to track stocks, mutual funds, options, bonds and cash accounts.

The Fund Manager system is distributed in three versions with increasing functionality: Personal, Professional and Advisor. The Personal version is more focused to individual investors and includes the standard features of portfolio creation and updating, Internet price retrieval, and chart
reporting, including trendlines. The Professional version is aimed for professional traders, with higher needs of technical analysis and includes all the functionalities of the Personal version plus analysis of portfolio statistics, including Sharpe ratio, Beta and Correlation, and security analysis, including simple Moving Averages, exponential Moving Averages and Bollinger Bands. Finally, the Advisor version focus on providing support to trading advisors and includes functionalities of client management and reporting, as well as interfacing with some brokers.

Overall, Fund Manager is a good portfolio management application regarding portfolio evaluation and only lacks the price forecasting analysis component to be a complete solution.

The Optimal Trader System [24] was developed in 2006, with the goal of applying signal processing to the analysis and forecasting of securities.

The operating method of Optimal Trader is to select and optimize five different models for each security and one top model based on artificial neural networks, and then make a trading decision based on the merging of the decisions made by each model. The types of models it uses for technical analysis are the following: Adaptive Moving Averages, Momentum, Regression Analysis, Relative Strength Index (RSI), Stochastic RSI, Stochastic, Stochastic Inverted, Parabolic Stop and Reverse, Combined Analysis Model, Trailing Stop-Loss Indicator.

Optimal Trader conceptually divides the indicators into two types: lagging indicators that assume the security price will continue to follow current trends and leading indicators that aim to predict when current trends will be broken.

Although research about applying intelligent computation to the trading of mutual fund assets is scarce, there have been several similar works considering the stock market instead of mutual funds. These works have explored genetic algorithms [6-9], genetic programming [10], genetic network programming [11-12][22], neural networks [13-14], fuzzy logic [15-18], natural language processing [19] or other techniques [20] in an attempt to create the ultimate moneymaking strategy for the stock market.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Inputs</th>
<th>Parameters</th>
<th>Outputs</th>
<th>Techniques</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>R. Jiang et all</td>
<td>Price History</td>
<td>Moving Averages</td>
<td>Buy / Sell</td>
<td>GA</td>
<td>Better than random-walk</td>
</tr>
<tr>
<td>W. Lin et all</td>
<td>Price History, Stock / Fund details²</td>
<td>Sharpe index, EPS</td>
<td>Portfolio</td>
<td>GA</td>
<td></td>
</tr>
<tr>
<td>H. Iba et all</td>
<td>Price History</td>
<td>-</td>
<td>Buy / Sell</td>
<td>GP</td>
<td></td>
</tr>
<tr>
<td>S. Mori et all</td>
<td>Price History, Stock / Fund details³, Financial Indicators</td>
<td>Moving Average</td>
<td>Next day prediction</td>
<td>GNP</td>
<td></td>
</tr>
<tr>
<td>Kwon et all</td>
<td>Price History</td>
<td>Moving Averages, RSI, Stochastics</td>
<td>Buy / Sell</td>
<td>NN, GA</td>
<td>Better than buy-and-hold</td>
</tr>
<tr>
<td>J. Mandiuk et al</td>
<td>Price History</td>
<td>Several Oscillators</td>
<td>Buy / Sell</td>
<td>NN, GA</td>
<td>14% above buy-and-hold</td>
</tr>
<tr>
<td>R. Simutis</td>
<td>Price History, Expert Opinions, Market Index</td>
<td>EPS</td>
<td>Buy / Sell</td>
<td>FL, GA¹</td>
<td></td>
</tr>
<tr>
<td>L. Talluru</td>
<td>Stock / Fund details², Investor preferences</td>
<td>-</td>
<td>Buy / Sell</td>
<td>FL</td>
<td></td>
</tr>
<tr>
<td>H. Wang</td>
<td>Stock / Fund details², Financial Indicators, Investor preferences</td>
<td>-</td>
<td>Portfolio</td>
<td>FL, GA¹</td>
<td></td>
</tr>
<tr>
<td>W. Yang</td>
<td>Price History, Stock / Fund details³, Financial Indicators</td>
<td>Clustering</td>
<td>Next day prediction</td>
<td>FL</td>
<td>87% hit ratio</td>
</tr>
<tr>
<td>V. Sagar</td>
<td>Expert Opinions</td>
<td>-</td>
<td>Next day prediction</td>
<td>NN</td>
<td></td>
</tr>
<tr>
<td>R. Lee</td>
<td>Price History</td>
<td>-</td>
<td>Next day prediction</td>
<td>NN</td>
<td></td>
</tr>
<tr>
<td>S. Mabu et all</td>
<td>Price History</td>
<td>IMX</td>
<td>Buy / Sell</td>
<td>GNP</td>
<td>Better than buy-and-hold in 12/20 stocks</td>
</tr>
</tbody>
</table>

¹ In this case the Genetic Algorithms are only used to boost the temporal performance of the other techniques
² Details include trading expenses, information on fund’s / company’s management team and other information not related to price of participation units / stocks.
In general, each of these intelligent computation techniques is capable of solving the problem with adequate modeling, but each of them faces different limitations. Neural Networks for example, although being the technique that has gathered a bigger number of supporters for this problem, has the limitation of not being able to withdraw conclusions about how its decisions are made, and therefore cannot create generic rules.

A summary of the characteristics of these applications can be found in Table 1. Since each of these authors tested their applications with different data, parameters and environments, it is impossible to establish an objective comparison between their results. Therefore, the option was to simply analyse what conclusions each of the authors drew from the performance of their applications in the testing period.

III. PROBLEM MODELLING

The management of a portfolio of mutual funds can be mapped to a search problem, which can be tackled by any search algorithm. This mapping of the problem is the real challenge presented. A well-defined search problem has 4 components: initial state; actions; goal test and path cost. After the problem is defined, an heuristic for the problem needs to be found to guide the search.

First of all, the problem needs to be classified either as an optimization, semi-optimization or constraint-satisfaction. As in most problems, it can be classified in more than one way, with different results depending on the choice. By choosing the optimization path, the problem can be stated as follows: find the portfolio of mutual funds with the highest expected return (for a set level of risk) in the time period considered. On the other hand, stating the problem as constraint-satisfaction can be as: find a portfolio of mutual funds with expected return higher (for a set level of risk) than the return provided by the reference index in the time period considered. While in the first case there is the need to search all the available mutual funds looking for the optimal solution (or until we are sure none of the unexplored solutions is better than the one found), in the latter the search algorithm can stop as soon as it reaches the first portfolio that satisfies the condition. A semi-optimization problem requires some reference for the value of the optimal solution in order to determine if a solution found is close enough to the optimal solution. Since in this problem we cannot estimate the value of the optimal solution before searching the entire state space, it cannot be stated as a semi-optimization problem. The choice between optimization and constraint-satisfaction will be made depending on the temporal and spatial complexities of the search, which will in turn depend on the amount of mutual funds considered. An optimization statement would be ideal, but if the number of mutual funds makes the search space so large that it cannot be fully explored in “real-time”, then the problem needs to be relaxed to a constraint-satisfaction statement.

After the goal condition has been determined, the following step is to decide what to consider as a state. Since our goal formulation depends on the evaluation of a portfolio, our states will have to allow the description of the portfolio and the mutual funds composing it. Considering an example where we have 6 mutual funds (A, B, C, D, E, F) one of the possible states can be \{IN(A), IN(D), IN(E), IN(F)\}. Assuming the investment is equally distributed through all the mutual funds in the portfolio, this is state description is sufficient to allow an evaluation of the current state. However, it does not contain the information of which mutual funds are not in the current portfolio, which is necessary information for the search algorithm to generate other states from this one. So, there is the need to add that information to the state description, resulting in a state description like \{IN(A), IN(D), IN(E), IN(F), OUT(B), OUT(C)\}. Once again, this description still assumes that the investment is equally distributed through all the mutual funds in the portfolio and all of the available money is being invested. If these conditions need to be lifted later in the design, then the state description would become \{IN(A, value_A), IN(D, value_D), IN(E, value_E), IN(F, value_F), OUT(B), OUT(C), AVAILABLE(value_available)\}. This representation is significantly more complex and computationally heavy to process, so for the remaining of this work the simpler description will be used, as it is sufficient for a successful solution of the problem. According to this description of state, we can define our initial state as our current portfolio. If we are building a new portfolio, the initial state is an empty portfolio, which in the example above would result in \{OUT(A), OUT(B), OUT(C), OUT(D), OUT(E), OUT(F)\}. If we are not building a new portfolio but instead revising and modifying one, the initial state represents the portfolio we are given.

The size of the search space will depend on the number of mutual funds we want to consider and also the number of mutual funds we want to have in the portfolio.

The next step is the description of the available actions. Considering the state description chosen, the actions necessary are simply those of buying and selling shares of a mutual fund. Formally, we
can represent these actions by a set of preconditions that need to be met before the action can be taken and a set of effects that results from taking the action. In this problem, we have simply the actions represented in Figure 1.

<table>
<thead>
<tr>
<th>BUY(x)</th>
<th>SELL(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRECONDITIONS: {OUT(x)}</td>
<td>PRECONDITIONS: {IN(x)}</td>
</tr>
<tr>
<td>EFFECTS: {IN(x)}</td>
<td>EFFECTS: {OUT(x)}</td>
</tr>
</tbody>
</table>

Figure 1 - Actions in the search problem

This notation allows the derivation of all the actions that result of instantiating "x" with any mutual fund. Note that according to the state description chosen, there is no need to specify how many shares of the mutual fund will be bought or sold, since that amount is only determined after the solution to the problem is found, depending on how many mutual funds are in the portfolio.

The missing element, path cost, brings another choice. The path cost represents the added cost of walking one step forward in the search space. In this problem, that corresponds to adding a new mutual fund to the portfolio or removing a mutual fund from the portfolio. Although it is possible to use the path cost to represent the buying and selling fees and therefore setting the path cost as a small value, it should be noted that adding to the solution a mutual fund that was already in the initial state or removing a portfolio which was not in the initial state has no cost, which would lead to some exceptions that would require additional mechanisms of comparisons past states. As a result, for this work we will consider that the amounts invested are sufficiently large for the trading fees to be ignored, and the path cost will therefore be 0.

Genetic Algorithms are a search technique inspired by the theories of human evolution. It is based on the concept of creating a population of individuals, where each individual represents a possible solution for the problem in analysis. Through a series of genetic processes, such as mutation, selection and crossover, each new generation of individuals represents (on average) a better solution to the problem than the individuals of the generation before. Studies indicate that Genetic Algorithms perform better in problems where parts of each individual (adjacent fields) represent meaningful components of a solution.

Unlike Neural Networks, with a Genetic Algorithms it is possible to draw information about how decisions are reached. This is an important aspect to consider since an ideal solution is capable of generating wisdom from the results obtained, discovering generic rules that can be applied in a wide range of situations.

As in any search algorithm, the efficiency of a Genetic Algorithm is mostly determined by how good is the heuristic used. Heuristics [23] are important in any search problem, but when dealing with a belief problem like this one, they are even more important. In a traditional search problem, an heuristic is merely an estimate of the cost of reaching a solution through a designated state. After a solution has been found, the actual cost of reaching that solution is determined and the values determined by the heuristic in the search for that solution are no longer relevant. In this case, however, the value of the solution (the effective return and not the expected return) will only be known after a solution has been found and executed. As such, the heuristic will have to not only estimate the value of a solution, but it will determine the beliefs of the application concerning the solution found, which will be the basis for any decisions made.

In the case of Genetic Algorithms, the heuristic is used to calculate the fitness value of each member. The fitness value is of utmost importance since the individuals with greater fitness value are the ones who are most likely to be selected to “parent” the next generation, so in order for the algorithm to reach a good solution, the fitness value must be a good indicator of how good is the solution represented by each individual.

An heuristic function for the problem of portfolio management must have two essential components. The first component evaluates the correlations between the mutual funds contained in the portfolio and effectively measures how well the portfolio diminishes risk. This component has been studied in detail by portfolio theory and several good measures have been found, so this part of the problem will be considered solved.

The second necessary component of the heuristic measures the expected profit for the portfolio in the future. Forecasting profit for a portfolio can be done by forecasting the profit for each mutual fund in the portfolio and calculating the sum of the estimates with the adequate weights, depending on the percentage of each fund in the portfolio.

As mentioned before, it is also necessary that the heuristics are computationally light so they don’t add even more complexity to the already computationally heavy search algorithm for portfolio generation.

The choice of a good heuristic is not a trivial task, and it will limit the success of the whole application. A bad heuristic in this problem will
cause poor evaluation of solutions, which will in turn lead to bad decisions being taken. The biggest question, however, is whether or not an adequate heuristic for this problem actually exists, since the problem requires the prediction of future events. Either there is proof that the future events are determined (or at least correlated to a high degree) with past events that we can analyse, or the future events are totally random. In the latter case, it is not possible to find an heuristic for this problem, and the problem itself is deemed unsolvable.

One possible method of improving the reliability of the forecasting is to use several heuristics at the same time and make a trading decision based on a voting system. Although this approach introduces an additional computational load, the order of complexity of computing several heuristics is the same as that of computing the heuristic in the set with higher order, so it is preferable to join several heuristics with the same order than to choose a single, more complex heuristic that has a higher order of complexity.

IV. ARCHITECTURE

A portfolio management application can be mapped to a 3-tier architecture, like the one shown in Figure 2. The interface layer handles the communication with the users, including both receiving the inputs for the tasks and returning the information as graphic reports. This layer is independent of the bottom layers and, as such, the graphic reporting system should ignore any implementation details and work with well-defined interfaces with the Trading Layer.

The Trading Layer is where the processing takes place, both the mutual fund evaluations and the portfolio generations. Once again, this layer must be independent of the Data layer below it so the implementation of the Trading Layer cannot have any knowledge of how the database is organized.

Finally the Data layer supports the Trading layer by taking care of the database management. In a portfolio management application, this task also includes updating the database periodically by accessing external sources through web services.

The prototype developed for this project focuses mainly in the component of heuristic evaluation and mutual fund forecasting. This is part of the Mutual Fund Evaluation Module, which is in turn integrated in the Trading Module of the global application.

It should be noted, however, that although the Mutual Fund Evaluation Module is part of the Trading Module and must be present in each instance of the application to allow the user to specify parameters for the evaluation, this module works independently of the search algorithm and the mutual fund evaluation can be done in pre-processing time relative to the portfolio generation.

From this fact, we can conclude that a copy of this module can be present in the remote server where the main database is stored and periodically compute the evaluation of each mutual fund with a default set of heuristics and parameters. The results of these evaluations would then be stored in the main database and could be downloaded by users who do not wish to customize parameters for this evaluation and / or have slower computers, thereby reducing the time needed to perform the global search (mutual fund evaluation plus portfolio generation).

The primary concern while developing the COIN prototype was fast development, in order to start testing heuristics and obtaining results to determine whether this problem is solvable as soon as possible. MATLAB was considered as a development environment due to the extreme simplicity of developing interfaces with the GUIDE feature. The alternative would be to choose a programming language that allowed for faster processing, namely C but, since the goal was to develop computationally light heuristics, MATLAB ended up being the final choice, according to the rationale of faster development.
This prototype was not designed to be an application for end-users but merely to serve as a testing tool to try out different choices of strategies and parameters and compare the results. Figure 3 shows the interface of the COIN prototype.

V. OPTIMIZATION KERNEL

In order to solve this problem, there was the clear necessity of finding adequate heuristics. As seen in chapter 3.3, the heuristics must be computationally light and, in order to join them together through a voting system, they must analyze different aspects.

In face of these requirements, heuristics based on complex graphic analysis methods like Bollinger Bands or chart pattern recognition have been dismissed due to the belief of being too computationally heavy.

In order to make sure the heuristics developed would analyze different aspects, the idea was to have one heuristic with no memorization at all, one that weights current information with past information and one that works mostly with past information. This rationale led to the three heuristics proposed and that are detailed next.

The first heuristic proposed, and the simplest of the three, is based in the principle of efficient market hypothesis. This principle states that the market variations reflect all the information currently know about the financial product in question. In practical terms, this means that if the value of a mutual fund increased since yesterday, this indicates that there is positive information about the mutual fund circulating and, more important, that unless there is new negative information about the mutual fund today, then the fund value is expected to increase again tomorrow.

To implement this concept, the Heuristic 1 creates a model defining a Rise threshold and a Fall threshold. Whenever the price variation in the last day meets or surpasses one of these thresholds, it indicates that next day's value will continue to rise or fall, respectively.

The chromosome for this heuristic is represented by a sequence of two floats. In each reproduction there are two individuals randomly selected from the population (father and mother) and the new individual inherits the Rise threshold from his father and the Fall threshold from his mother.

When an individual suffers a mutation, it has a 50% probability of changing its Rise threshold to a new random value inside the parameter limits and a 50% probability of changing its Fall threshold to a new random value.

The second heuristic proposed introduces the concept of relevant maximums and relevant minimums. Since Heuristic 1 only takes into account last day's value and today's value and ignores every other information, Heuristic 2 uses a similar model but introduces some memory into the...
strategy. Memorising every past value and considering every past value on each step would result in a daunting computational weight, so this heuristic only memorizes the most recent relevant maximum and the most recent relevant minimum.

The “relevant” concept here is meant to ignore very small variations in the NAV that should not be considered. The heuristic therefore ignores any variation lower than the thresholds defined in the model.

“Relevant maximums” and “relevant minimums” are defined incrementally since the start of the fund’s price history. The starting value of the fund is considered both as a RMA and RMI.

A RMA is defined as a maximum with an X increment since the last RMI before it, where X is the Rise threshold.

A RMI is defined as a minimum with a Y decrement since the last RMA before it, where Y is the Fall threshold.

Each new RMA indicates that next day’s NAV will rise and each new RMI indicates that next day’s NAV will fall.

Since the model for this heuristic is identical to the model of Heuristic 1, the chromosome structure is also identical, as well as the Reproduction and Mutation mechanisms.

The third heuristic proposed explores the technique of simple moving averages. Four different moving averages were considered, with periods of 10, 20, 50 and 100 days. This heuristic generates a model that determines whether each of the four moving averages being higher than the NAV is a positive or negative indicator and assigns a weight to each moving average. During evaluation, each moving average is compared to the NAV and its weight is added or subtracted from a total, which is finally compared with the Rise and Fall thresholds to determine if the fund will likely rise or fall. For example, if the 10-days MA has a positive weight of 0.3 and the 10-days MA is higher than the NAV, then it will contribute with a positive 0.3 to the final decision. If the 10-days MA were lower than the NAV, it would contribute with a negative 0.3 to the final decision.

The differences between the weights effectively establish an order of importance among the moving averages, where the moving averages with higher weight have bigger influence in the final decision. Having a Rise Threshold of 1 means that the number of positive indicators minus the number of negative indicators must be equal to or greater than 1 for the heuristic to indicate a Rise decision. The Fall threshold of 2 means that the number of negative indicators minus the number of positive indicators must be equal to or greater than 2 for the heuristic to indicate a Fall decision.

The chromosome for this heuristic is represented by a sequence of six floats, with each representing a gene. In each reproduction there are two individuals randomly selected from the population (father and mother) and the new individual generated has 50% probabilities of inheriting each gene from the father and 50% probabilities of inheriting it from the mother, for a total of 36 possible combinations. A possible alternative would be the classical GA crossover, used for the portfolio search problem.

When an individual is chosen for mutation, one and only one of its genes will be changed to a new random value, giving each gene a 16.7% probability of being changed.

VI. TEST RESULTS

In order to test the heuristics, a series of tests was ran using the previously introduce COIN prototype. All the tests were ran using the same set of 11 Mutual Funds with different evolution patterns, with an average result for the Buy-and-Hold strategy of 121.09% in the 256 days testing period and 90.31 in the 32 days testing period.

To allow direct comparisons, all the tests were ran using the same parameters:
- 25% of data for testing period and 75% of data for training period (usual split for data mining is 70% training and 30% testing but changing to 75/25 avoids some unnecessary round-ups)
- 50 generations of the GA (50 can be considered a low value, but earlier tests determined the quality of the solutions with too many generations would tend to decrease due to overfitting, plus we want the heuristic optimization process to be computationally light).
- populations of 64 individuals
- 5% mutation chance for each individual.

Each batch of tests comprised 10 runs. Two sets of tests were ran, one with 256 days testing period and 768 days training period and one with 32 days testing period and 96 days training period. While the 256 days testing period provides the heuristics with a long training period and a testing period where most of the funds are rising, the 32 days testing period forces the heuristics to work with less data and in a testing period where most funds are falling.

Results from the testing of Heuristic 1 can be found in Table 2. These tests were ran allowing the Rise and Fall Thresholds to vary between -5.0 and +5.0 with one decimal digit, resulting in a number of different combinations of 10000.
Table 2 - Results of Heuristic 1: strategy based on daily variations

<table>
<thead>
<tr>
<th>Size of Testing Period</th>
<th>32</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average result:</td>
<td>95.75</td>
<td>126.18</td>
</tr>
<tr>
<td>Average Standard Deviation</td>
<td>4.18</td>
<td>11.20</td>
</tr>
<tr>
<td>Average number of days in the market:</td>
<td>18.75</td>
<td>177.76</td>
</tr>
<tr>
<td>Average number of entries:</td>
<td>2.71</td>
<td>16.05</td>
</tr>
<tr>
<td>Average number of entries with profit:</td>
<td>1.00</td>
<td>9.34</td>
</tr>
<tr>
<td>Average profit per entry (%):</td>
<td>-1.31</td>
<td>1.75</td>
</tr>
<tr>
<td>Average profit per day in the market (%):</td>
<td>-0.23</td>
<td>0.15</td>
</tr>
<tr>
<td>Average percentage of good decisions:</td>
<td>53.07</td>
<td>55.26</td>
</tr>
</tbody>
</table>

The results for this strategy were good. The average results were above the Buy-and-Hold strategy in both the testing periods, and the average number of days in the market is significantly lower (58.6% and 69.4% depending on testing period) making it less risky than the Buy-and-Hold strategy.

The percentage of good decisions was lower than expected, but ignoring the funds with inadequate training period causes an increase of this value to about 58%. Still, this value is too low for this heuristic to be considered good for this problem.

Table 3 - Results of Heuristic 2: strategy based on relevant maximums and minimums

<table>
<thead>
<tr>
<th>Size of Testing Period</th>
<th>32</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average result:</td>
<td>98.66</td>
<td>127.73</td>
</tr>
<tr>
<td>Average Standard Deviation</td>
<td>1.72</td>
<td>8.22</td>
</tr>
<tr>
<td>Average number of days in the market:</td>
<td>14.25</td>
<td>202.81</td>
</tr>
<tr>
<td>Average number of entries:</td>
<td>2.71</td>
<td>12.98</td>
</tr>
<tr>
<td>Average number of entries with profit:</td>
<td>0.90</td>
<td>6.49</td>
</tr>
<tr>
<td>Average profit per entry (%):</td>
<td>-0.29</td>
<td>1.82</td>
</tr>
<tr>
<td>Average profit per day in the market (%):</td>
<td>-0.09</td>
<td>0.14</td>
</tr>
<tr>
<td>Average percentage of good decisions:</td>
<td>53.98</td>
<td>54.58</td>
</tr>
</tbody>
</table>

Results from the testing of Heuristic 2 can be found in Table 3. These tests were ran allowing the rise and fall thresholds to vary between –5.0 and +5.0, with one decimal digit, resulting in a number of different combinations of about 10^9. However, the implementation of the algorithm automatically eliminates the combinations where the Rise or Fall thresholds surpass the numbers that can be reached by combining the weights of the moving averages, thus reducing the number of actual different combinations to about 10^8.

This heuristic also provided good results in the testing period. Although the average result was a little less than the results obtained with the Buy-and-Hold strategy for the 256 days test (121.09), this value was accomplished with a much lesser number of days in the market (only 37.9%), a small number of entries and a good average profit per entry, making it a viable strategy.

The 32 days test this heuristic performed very well, surpassing the Buy-and-Hold strategy.

Once again, the downfall of this heuristic is the low percentage of good decisions, which is even lower in this case.

Table 5 provides a summarized comparison between the results of each heuristic and the Buy-and-Hold strategy. All three heuristics outperformed the Buy-and-Hold strategy in the 32 days test and 79.2%) but the average gain per entry in this test (1.82%) was very good.

It should also be noted that this heuristic provided a considerably smaller average standard deviation than heuristic 1.

As happened with Heuristic 1, the negative factor in the results is the low percentage of good decisions, which is not high enough for this heuristic to be considered reliable.

Table 4 - Results of Heuristic 3: strategy based on moving averages

<table>
<thead>
<tr>
<th>Size of Testing Period</th>
<th>32</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average result:</td>
<td>96.83</td>
<td>118.84</td>
</tr>
<tr>
<td>Average Standard Deviation</td>
<td>3.30</td>
<td>8.20</td>
</tr>
<tr>
<td>Average number of days in the market:</td>
<td>11.84</td>
<td>97.05</td>
</tr>
<tr>
<td>Average number of entries:</td>
<td>1.36</td>
<td>8.14</td>
</tr>
<tr>
<td>Average number of entries with profit:</td>
<td>0.11</td>
<td>3.51</td>
</tr>
<tr>
<td>Average profit per entry (%):</td>
<td>-3.22</td>
<td>1.88</td>
</tr>
<tr>
<td>Average profit per day in the market (%):</td>
<td>-0.27</td>
<td>0.19</td>
</tr>
<tr>
<td>Average percentage of good decisions:</td>
<td>54.08</td>
<td>51.40</td>
</tr>
</tbody>
</table>

Results from the testing of Heuristic 3 can be found in Table 4. These tests were ran allowing the weight of each moving average to vary between –1.0 and +1.0 and the rise and fall thresholds to vary between –4.0 and +4.0 with one decimal digit, resulting in a number of possible different combinations of about 10^8. However, the implementation of the algorithm automatically eliminates the combinations where the Rise or Fall thresholds surpass the numbers that can be reached by combining the weights of the moving averages, thus reducing the number of actual different combinations to about 10^8.

This heuristic also provided good results in the testing period. Although the average result was a little less than the results obtained with the Buy-and-Hold strategy for the 256 days test (121.09), this value was accomplished with a much lesser number of days in the market (only 37.9%), a small number of entries and a good average profit per entry, making it a viable strategy.

In the 32 days test this heuristic performed very well, surpassing the Buy-and-Hold strategy.

Once again, the downfall of this heuristic is the low percentage of good decisions, which is even lower in this case.
only heuristic 3 presented lower results than the buy-and-hold strategy on the 256 days test.

### Table 5 – Comparison between the heuristics and the Buy-and-Hold strategy

<table>
<thead>
<tr>
<th>Size of Testing Period</th>
<th>32</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy-and-Hold</td>
<td>90.31</td>
<td>121.09</td>
</tr>
<tr>
<td>Heuristic 1</td>
<td>95.75</td>
<td>126.18</td>
</tr>
<tr>
<td>Heuristic 2</td>
<td>98.66</td>
<td>127.73</td>
</tr>
<tr>
<td>Heuristic 3</td>
<td>96.83</td>
<td>118.84</td>
</tr>
</tbody>
</table>

### VII. CONCLUSIONS

The results obtained with the tests show that genetic algorithms have indeed the potential to be helpful in the domain of mutual funds. The three heuristics proposed (daily variations, relevant maximums and minimums and simple moving averages) all yielded positive results which can surely be improved with more focused research on each of the areas. The low percentage of good decisions obtained with all the heuristics is definitely a serious concern. After all, it would be hard to persuade someone to invest money based on an indication with a 55% chance of being accurate. However, it should be noted that studies indicate that more than half of the investors lose money in the market, so a 55% chance of making profit can be considered relatively high.

Out of the 3 heuristics proposed, the ones based on daily variations and relevant maximums and minimums (1 and 2) were the ones that provided the best results on the set of mutual funds used for testing and showed good capabilities of generalization, performing better than the buy-and-hold strategy even in the two funds who had a very short training period (CAAMAqu with 38 days and CAAMInd with 68 days).

When looking at the average profit per day in the market, the performance of the heuristic based on moving averages is the best. Since the APPDIM obtained by the heuristic based on moving averages in the mutual funds with bigger profits is higher than the APPDIM obtained by the daily variations or relevant maximums and minimums heuristics in these funds, we can conclude that the strategy based on MA is more adequate for a portfolio management application with the goal of “cherry-picking” the best days from each fund, although such a strategy is also more risky.

This thesis has proved that the Genetic Algorithms can be a valuable asset for the management of Mutual Funds and an application based on Genetic Algorithms can help an investor to obtain bigger profits than those obtained by the traditional buy-and-hold strategy. It has also shown that there are strategies more fit to generalization and which can be applied to the majority of mutual funds and strategies specialized in a type of mutual fund or more fit to specific goals. The challenge is to determine among the infinite number of possible strategies, inputs and parameters that can be considered by such an application, combinations that provide good trade-offs between risk and expected profit.

To improve the proposed heuristics or develop new ones, there are many possibilities of additional information to consider, in particular global economy indicators, like the most important financial indexes (like NASDAQ and Dow Jones) and reference interest rates, since the evolution of world economy is always reflected to some degree in all financial products, such as mutual funds.

Alternative heuristics can be developed based on other popular financial indicators, like Bollinger Bands or chart pattern recognition. It is important to note, though, that each heuristic must be computationally light to perform its duty or it will become useless when scaling the application to work with a large number of mutual funds.

Finally, a more focused study on generalizing models must be made, so the knowledge gained from the Genetic Algorithms optimization for each mutual fund can be converted into wisdom, through the definition of generic models that can be applied to multiple mutual funds without the need of being first optimized to each mutual fund.

### REFERENCES


