COS314 Artificial Intelligence – Project II Report

Report complementing project on discrete optimization (automated stock trader)

Student Names: Kevin Mc Inerney (u15402569), Noreen Lenihan (u15402437)
Course Instructor: Dr. Katherine Malan

Table of Contents

Kevin Mc Inerney & Noreen Lenihan

COS314 Artificial Intelligence – Project II Report

Report complementing project on discrete optimization (automated stock trader)

Abstract

The purpose of the present report is to outline the key implementation steps and decisions of applying artificial intelligence optimization techniques to build an automated trading system. Precisely, the authors exploited a well-know local search algorithm – basic hill-climber – and a global search algorithm (genetic algorithm) to mechanize trading rules to produce a profitable autonomous agent in variable market conditions. We present analyses that contrast the quality of the solutions returned by the hill-climber and genetic algorithms. Results and discussions are further presented to extricate the various aspects of the trading system that proved advantageous and those that warrant further examination. Finally, the authors advance their putative 'best trading strategy' that has consistently produced the most reliable and profitable results in diverse market conditions, and thus, is deemed generalizable.

Introduction

Genetic algorithms (GA; Holland, 1975) are randomized search procedures that work on a 'population' of individuals that represent solutions. The population evolves over time as a result of operators that purport to emulate natural selection such as selection of the fittest individuals, mutation, and crossover or mating. The main steps involved in the present genetic program can be described as follows (Potvin, Soriano & Vallee, 2004):

Step 1: *Initialization* – In the initial step, we initialize our current population of traders with random trading strategy and evaluate their fitness based on their strategies.

Step 2: *Selection* – Next, we select *k* number of individuals from the population, depending on the selection strategy chosen (rank, Roulette wheel, tournament, random), to be parents for new offspring.

Step 3: *Modification* – The chosen parents from the selection process now mate or hybridize to reproduce. Mutation may also be applied to any individual in the population (not necessarily offspring) at this stage.

Step 4: *Evaluation* – Finally, we obtain the fitness of each new individual in the population, whereby fitness is defined as profit excluding initial wallet amount.

The above steps are repeated until the maximum number of generations has been reached (stopping condition) in our case although there exists other stopping conditions (e.g. stagnation).

Contrastingly, the hill-climbing algorithm is a single solution-based meta-heuristic, that is, unlike GAs which work on a host of solutions at the same time, it works on a single solution at a time (Malan, personal communication, 2015). Hill-climbing is a local optimization technique compared to global optimization (GA). There are many variants of this algorithm for example simple hill-climber (first best neighbour) and steepest ascent hill-climber (best neighbour) (Brownlee, 2011). The routine for a typical hill-climber can be described as follows:

Step 1: Generate an initial solution within constraints of problem

Step 2: From this solution, generate a set of new candidate solutions (neighbours)

Step 3: Select a solution from this collection of solutions that is closer to the goal state the current best solution.

Step 4: Repeat Step 2 with new solution until some stopping condition is reached

Step 5: Output best solution

Hill-climbers are susceptible to local maxima, ridges and plateaus, a discussion of which is beyond the scope of the current report.

In the sections below, we present the findings of both algorithms, and some reflections on the results for each.

Implementation Details

It is instructive to note some key implementation decisions in the present project.

- As our training data used $US Dollars as its currency, the authors decided to convert the initial wallet (R100,000 as specified) and all transaction costs to this currency.
- It was unanimously agreed by the authors that the automated trader should be able to manage multiple stocks simultaneously. Towards this goal, Renko charts for each individual stock were constructed and compiled into a 'Master Table' – a TreeMap data structure – whose key was 'Date' (see Figure 1). This strategy allows the system to place stock entries with the same Renko pattern (for example, 5 solid bricks) on the same date together. As all these patterns must conform to the same buy/hold/sell decision when an appropriate brick pattern is detected, it was necessary to contemplate which stock should be bought/held/sold on a particular date when many stocks followed the same pattern on the same date. To illustrate, if both AGL and GFI stocks had a Renko pattern corresponding to a 'buy' decision on 23/5/2003, one of these stocks must be prioritized and bought, to accord with the project specification which issues that "every time a trader makes a buy decision, it must buy the maximum number of shares possible with the money on hand (including fees…)". Originally, a priority queue was implemented to deal with this – stocks with similar decisions on a particular date were evaluated to detect the stock with the highest volatility. We used the current size of a particular stock's Renko chart at time of entry into the map as an index for its volatility such that a stock with a larger Renko chart size denoted a higher volatility, as there was, on average, more instability in its stock prices over time. When it was not possible to delimit the stock that espoused higher volatility, the algorithm then chose the stock that would leave the smallest remainder in the trader's wallet. We supposed that a higher volatility would connote further buys/sells in the longer term, which was favourable, and so conferred a higher priority on this stock. Despite the promise of this strategy in terms of profitability, it was later re-evaluated and discarded because its prioritizing of certain stocks worked against the principle of random sampling which is necessary for generalization to occur. Similarly, the very notion of whether higher volatility actually leads to higher profitability is beyond the ken of the authors. This type of prioritization may be beneficial in optimizing a strategy's performance once it has been chosen, however, when in the process of trying to obtain a strategy, it is deemed non-beneficial.

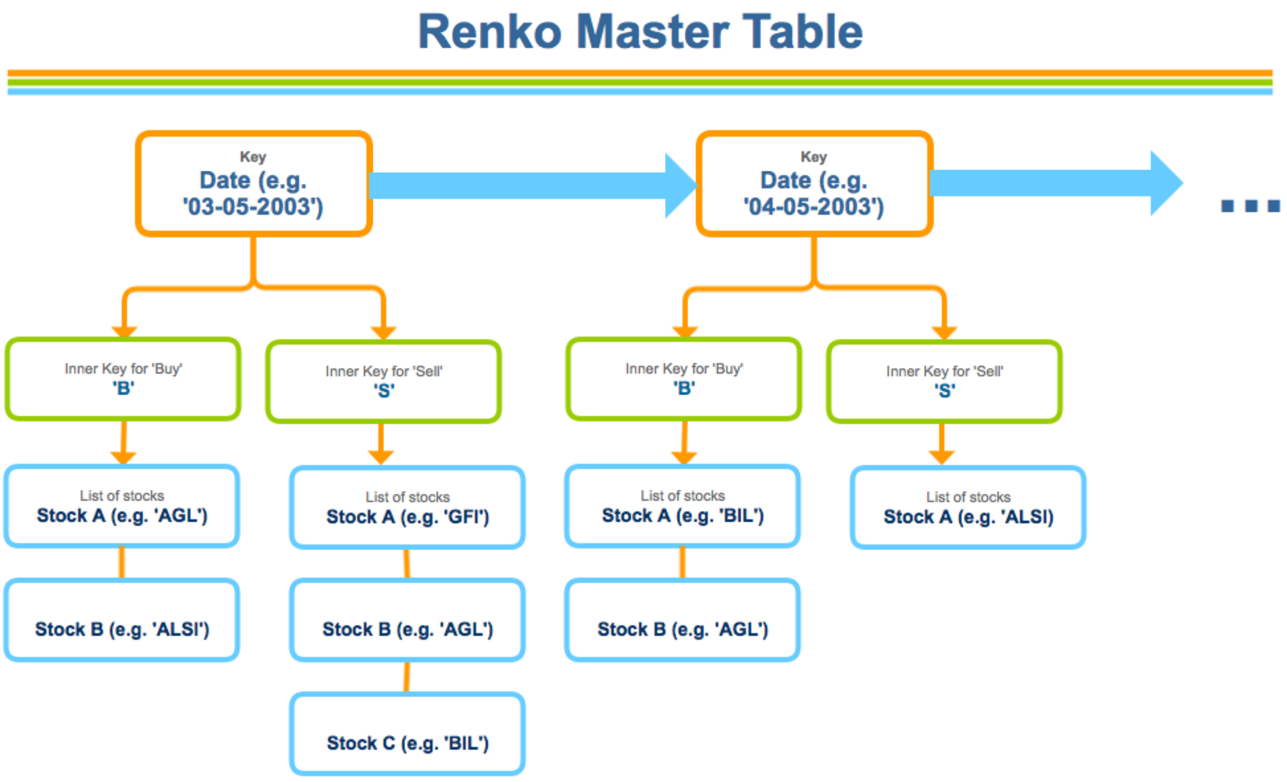

Figure 1. Tree Map data structure representing amalgamation of Renko charts from multiple stocks

- To prevent prioritizing a stock based on its stability, which would not likely generalize well to other data, the next general consensus was to randomly pick one stock to buy/sell/hold when there is more than one stock present in the Master Table on a given date with a given predetermined decision ('B' or 'S'). The authors conceded that this would lead to more unbiased and generalizable outcomes.
- To ensure that the trader does not make trivial purchases, whose transaction costs obscure the stock prices themselves, we enforced a limit such that the minimum stocks that the trader could purchase is 4. Below this limit, it obfuscated the chances of the trader to make worthwhile stock purchases later on. We called this method 'blocking'.
- To evaluate the trader's fitness appropriately, we implemented a 'SellOffAssets()' method in our Trader class to clear the trader's stock portfolio at the end of the trading period, to best gauge its return.
- In our efforts to discover the most profitable trading strategy, we utilized the popular Sharpe Ratio by obtaining the mean profit of a trader divided by its standard deviation. Originally devised by William F. Sharpe, this widely-accepted index is "a way to examine the performance of an investment by adjusting for its risk" (Wikipedia, 2015).

In general, a calculated Sharpe ratio value of 1 is appreciable, 2 is very good, and 3 is excellent. We used to metric to determine our ‘best trader’. In choosing a strategy, we strived for a balance between profitability and risk as measured by the Sharpe ratio.

Fitness Evaluation

The fitness of an individual trader is defined as the profit earned by the trader at the end of the trading period. In our implementation, it is necessary to divide the returned figure by 100 to obtain the amount in $US Dollars (all calculations deal with cents due to the dramatic performance difference when using integers versus doubles in these types of problems).

Hill-Climber

In this section, we provide an overview of the Hill-Climber algorithm in the current project, and discuss the results obtained using this algorithm for stocks in period 2003-2008 (sourced externally from Yahoo Finance).

Overview

The current project makes use of a basic hill-climber to find the most profitable trader.

Neighbourhood Function

Our hill-climber finds a trader’s neighbours by taking in the string representation of the current trader’s strategy (e.g. ‘BSHHHSBS…’) and changing the letters from left to right until it yields a more profitable strategy. Importantly, when changing a letter representing a buy or sell or hold, for example the letter ‘S’, we implemented a random/probabilistic method of choosing either of the alternative letters, in this case, ‘H’ or ‘B’. The authors felt this was more statistically sound than, for example, systematically choosing ‘H’ every time it encountered an ‘S’.

Results

To animate the progression of the hill-climbing algorithm process, Figure 2a depicts the growth in performance of ten different hill-climbers on the same stocks. It is apparent from the graph that hill-climbers are susceptible to local maxima.

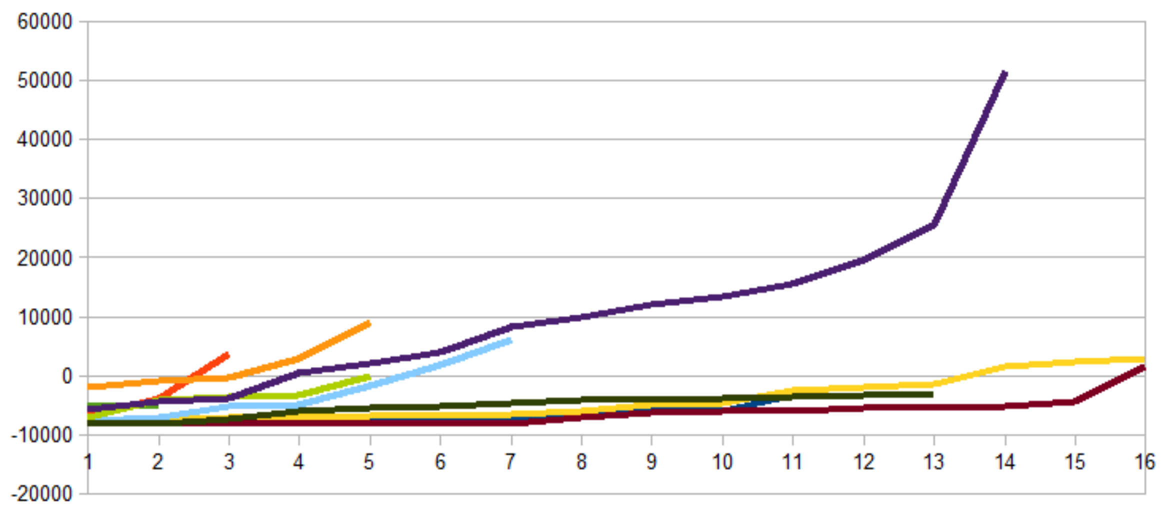

Figure 2a. Progression of hill-climbing algorithm for ten difference instances

We next ran 50 instances of Hill-Climber – using 13 stocks - for a more nuanced examination of the minimum, average, and maximum profits it yields (see Figure 2b). The maximum profit obtained from this test was $33,391.66, the minimum profit yielded was -$8,179.12, and the average profit over all 50 instances was $6,945.20.

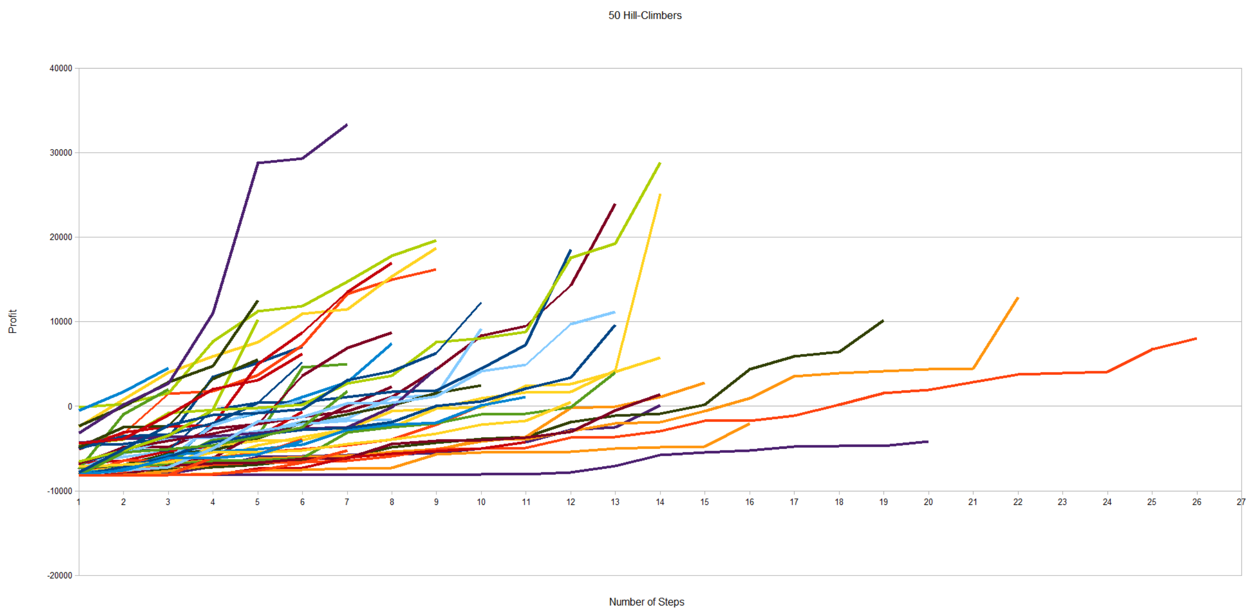

Figure 2b. 50 Hill-Climbers performance on 13 stocks

We first trained our hill-climber on four stocks from the 2003-2008 period (see Fig. 3). Sorting the resultant strategies based on the increasing profit yielded the blue line in the graph below. Next, we extracted these strategies, in sorted order, and evaluated the performance resulting from these strategies on the training data (red line) – which shows a close relationship to the blue line as anticipated. Finally, we tested these strategies on our test data, consisting of 4 different stocks from the same period. As depicted by the green line in Figure 3, it is clear that there is little correspondence to the training data. At first glance, it may appear that the green line shows a steady increase. However, this trend is difficult to accept due to the large standard deviation. Upon examination of the test data, it was found that one particular stock was unusually profitable and this may have contributed to the large standard deviation.

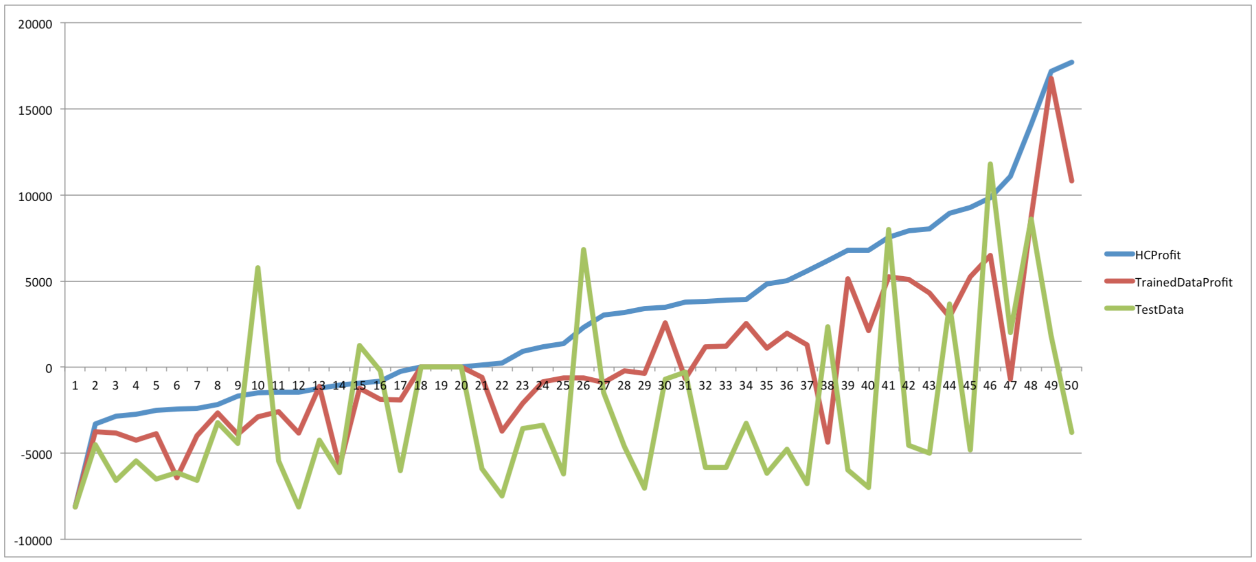

Figure 3. Hill-climbing algorithm demonstrating profit attained over 50 runs

In an attempt to obtain a clearer picture of the relationship between the test data and our chosen strategies, we redid the analysis using different randomly sampled test stocks (see Figure 4). Although the training data clearly follows the blue line formula, when these strategies were utilized on the unseen data, there is little correlation. Such results indicate a clear lack of generalization to our test data.

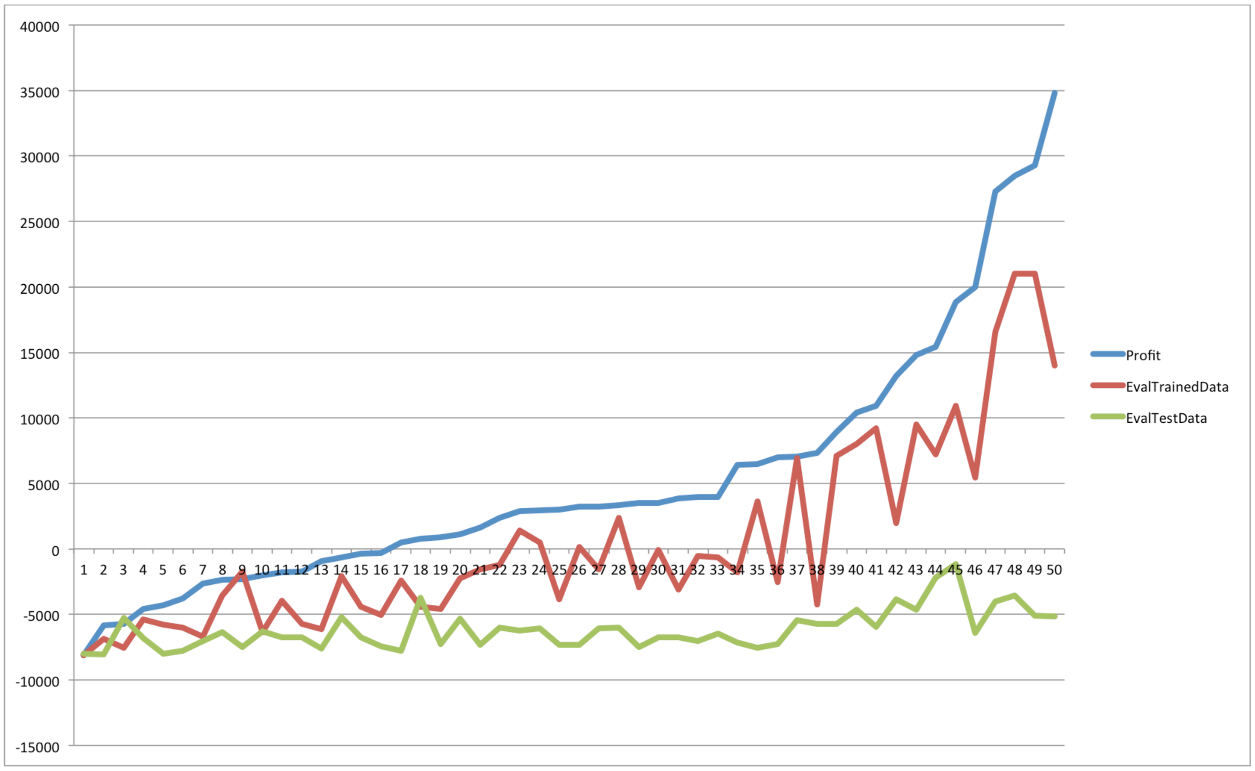

Figure 4. Hill-climbing algorithm on another set of test data over 50 executions

To overcome the apparent lack of generalization on our test data, we then trained our algorithm on a larger set of data – namely, 13 stocks from the same period. We hypothesized that this effort would lead to greater generalization on our test data, which consisted of 4 randomly sampled stocks. As demonstrated in Figure 5, the results did not conform to our assumption, and little generalization was again visible.

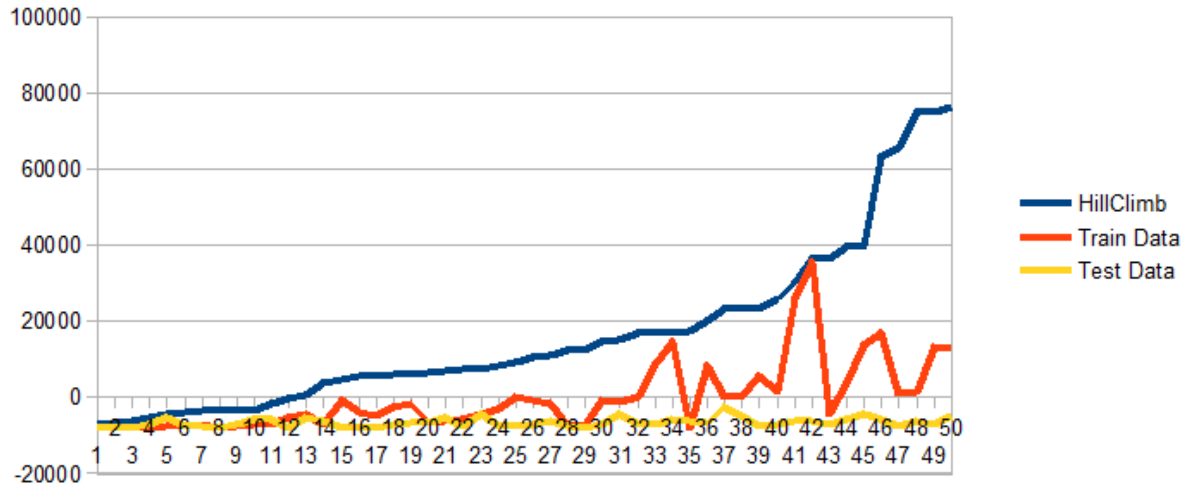

Figure 5. Testing hill-climbing algorithm trained on larger training set consisting of 13 stocks

Genetic Algorithm

Overview

We implemented a genetic algorithm in the present project to find a profitable and generalizable trading strategy. We make use of Roulette wheel, random and tournament selection strategies, crossover – one and two-point, and mutation procedures.

Results

To determine the optimal parameters for the genetic algorithm, we conducted a series of experiments investigating the combination of various settings. Due to the large domain of settings, we decided to split the possible parameter settings into five categories that we deemed to be representative of high, low, and medium selection pressure. Further, we probed various settings in which selection pressure increased from low pressure to medium pressure and low pressure to high pressure. The actual settings utilized for the various mutation, crossover, etc. operators are presented in the Table 1 below. Results for these settings can be observed in Figure 6.

	High Pressure	**Low Pressure**	**Medium Pressure**	**Low To Medium Pressure**	**Low To High Pressure**
Population size	100	100	100	100	100
Maximum generations	1000	1000	1000	1000	1000
Selection Type	Tournament	Tournament	Tournament	Tournament	Tournament
Crossover Type	One-point	Two-point	One-point	Both*	Both*
Mutation Rate	0.01	0.2	0.15	Low-med**	Low-high***
Crossover Rate	0.1	1.0	0.6	Low-med**	Low-high***
Generation Gap	1.0	0.025	0.1	Low-med**	Low-high***
Tournament Size	1.0	0.05	0.5	Low-med**	Low-high***

Table 1. Parameter settings for experiments on finding optimal GA settings

* Algorithm changes from two-point crossover to one-point crossover after 500 generations

**Low-to-medium settings changed in increments/decrements from low settings to medium settings every 100 generations

***Low-to-high settings changed in increments/decrements from low settings to high settings every 100 generations

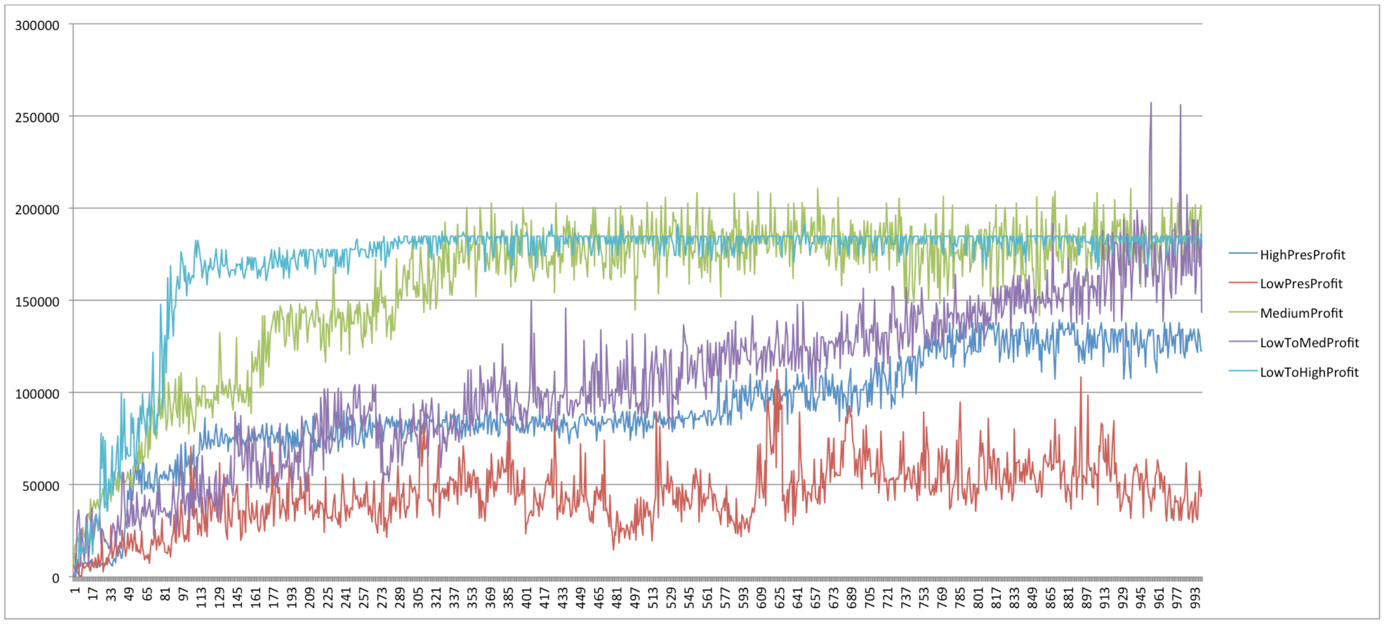

Figure 6. Experiments on different parameter settings representing high, low, and medium selection pressure

First, we tested our data on our genetic algorithm using high selection pressure settings. Over 1000 generations, there was little to no generalization on unseen data. However, when we examined the first 200 generations, there appeared to be some improvement in performance. It was supposed that over-fitting did not have an opportunity to hinder generalization in the earlier generations. This effect can be observed in Figure 7 below.

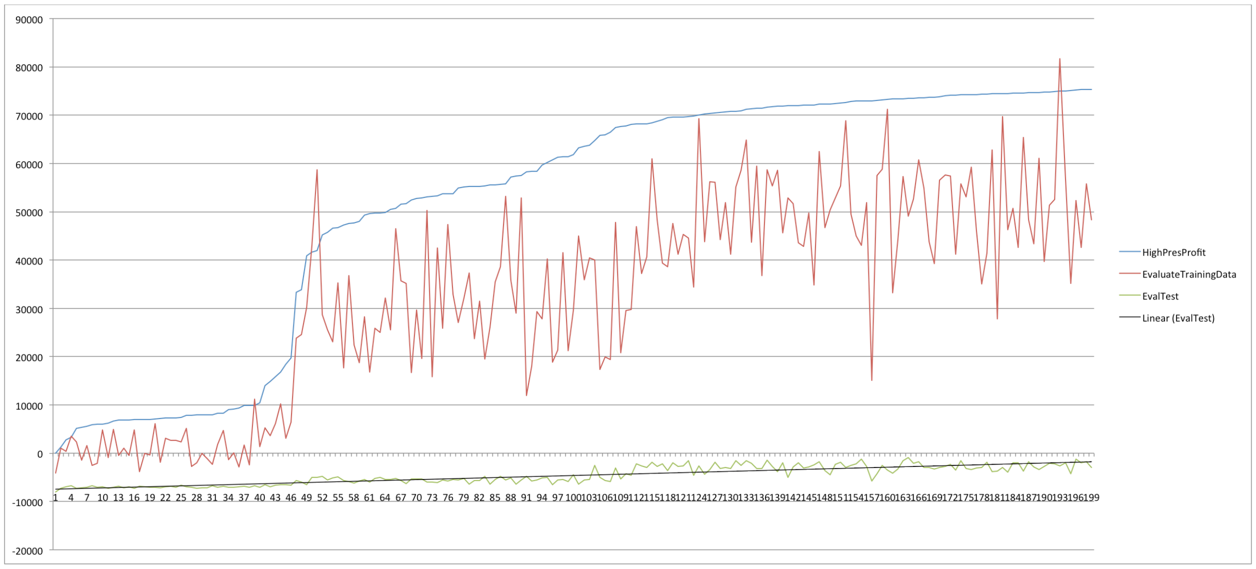

Figure 7. High selection pressure settings on unseen data over first 200 generations

Next, we tested our low selection pressure settings on unseen data. There appeared to be no generalization on unseen data.

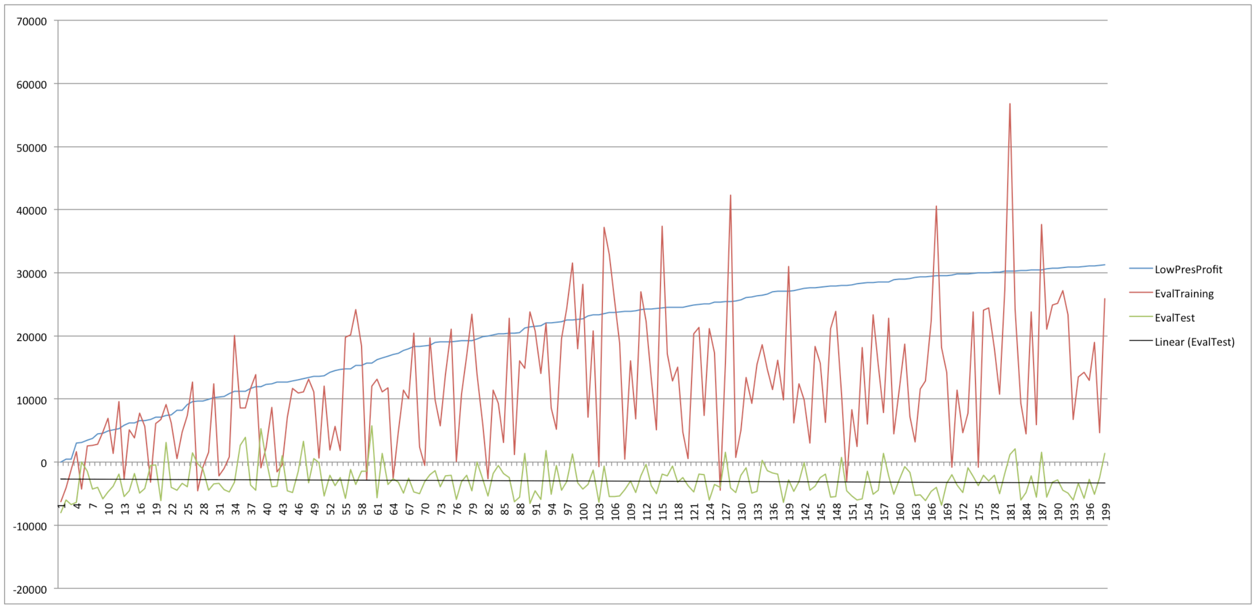

Figure 8. Low selection pressure settings on unseen data over first 200 generations

The next graph shows performance of the GA using our specified medium pressure settings over the first 200 generations. As above, there is a clear lack of generalization.

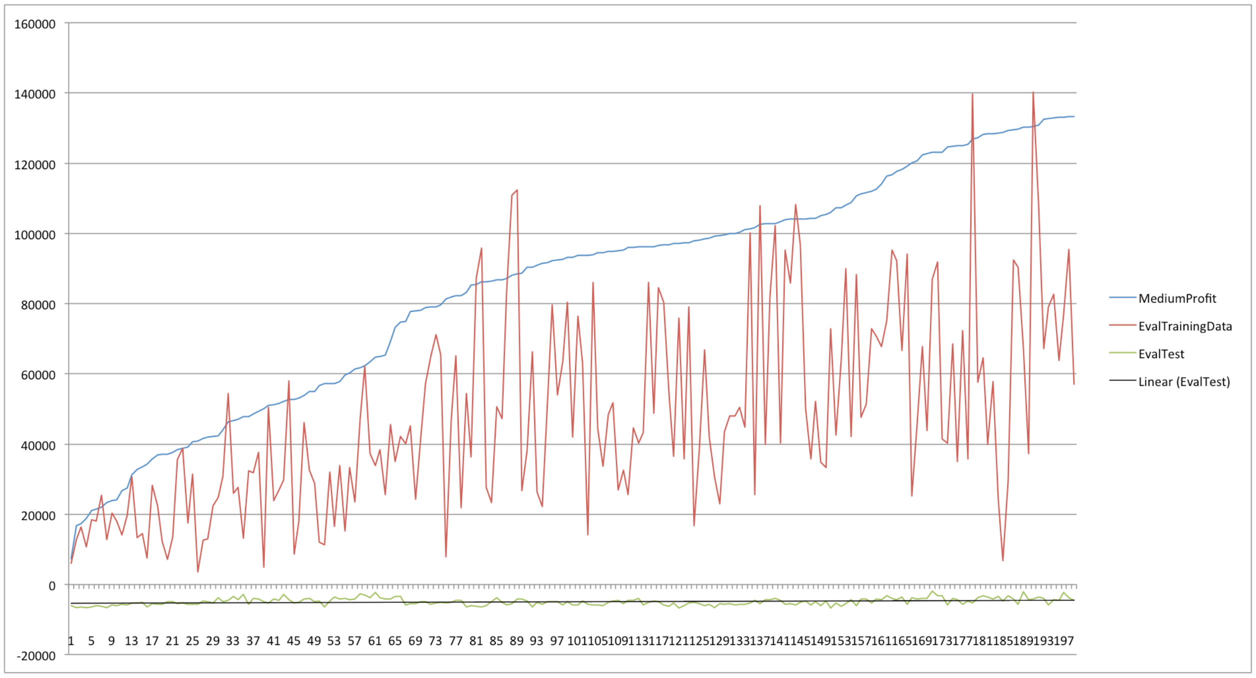

Figure 9. Performance of GA using medium selection pressure settings over first 200 generations

The next graph depicts the GA's performance using our low-to-medium parameter settings. No generalization is apparent.

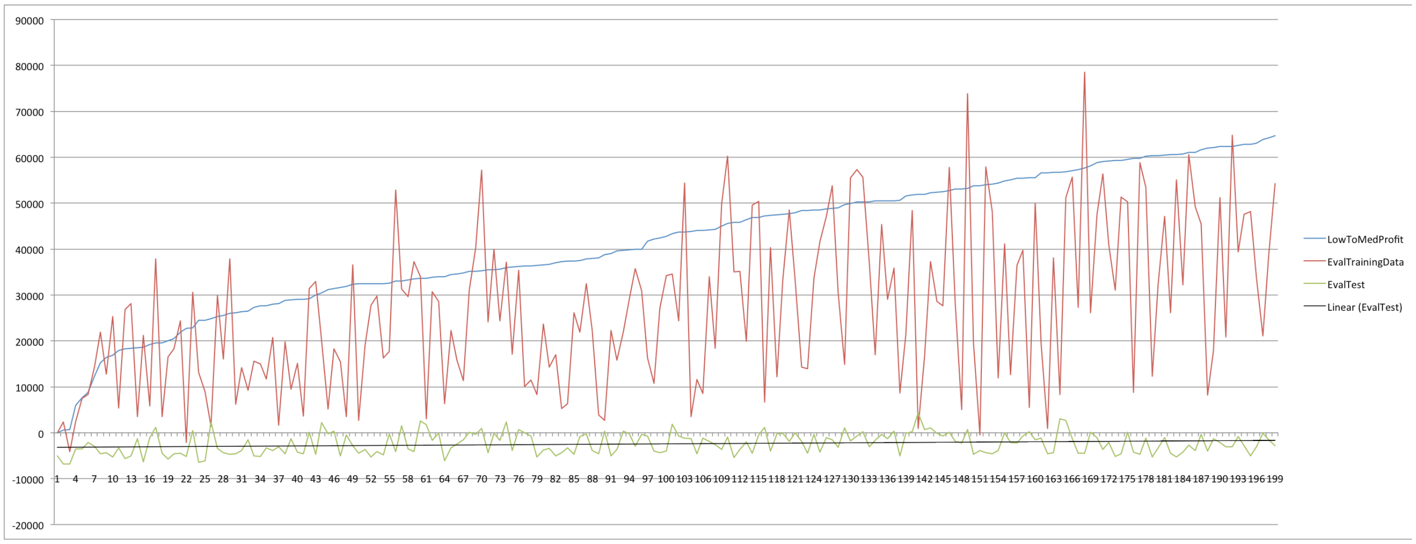

Figure 10. Performance of the GA using low-to-medium parameter settings over 200 generations

Finally, our graph depicting the GA using low-to-high selection pressure settings can be observed in Figure 11. No generalization is in evidence.

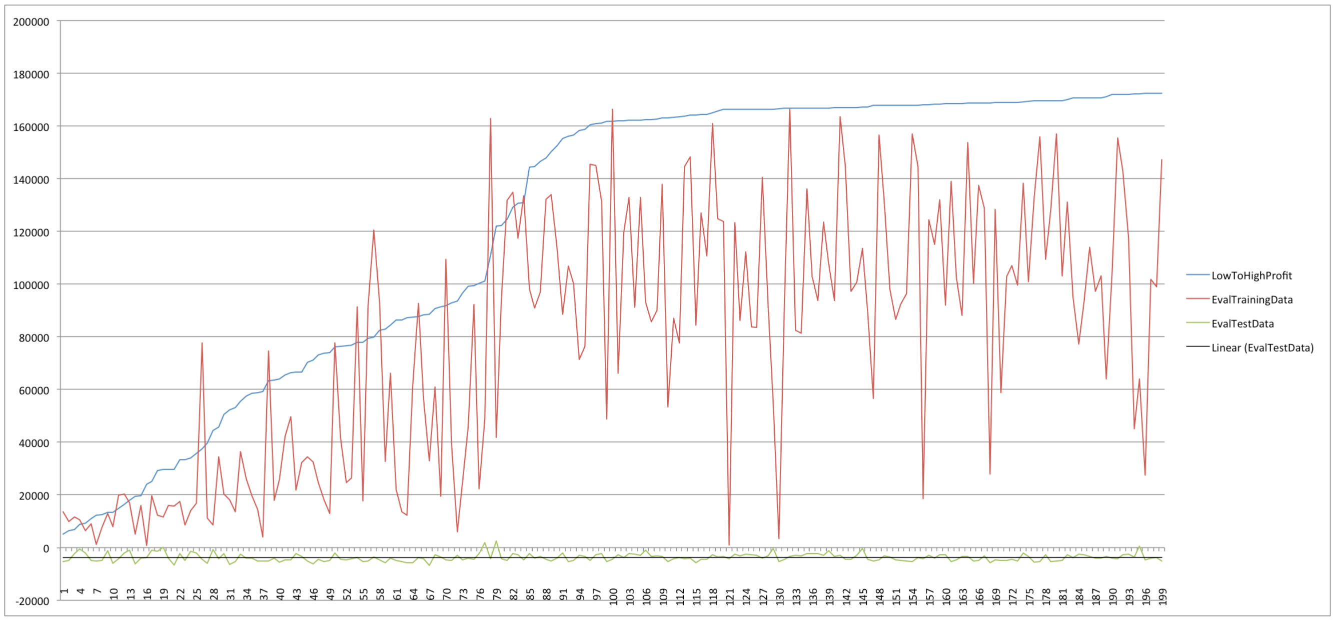

Figure 11. Performance of GA on unseen data using low-to-high selection pressure settings for first 200 generations

To increase generalization, we trained the GA on a larger dataset – 13 stocks. This change lead to a modest increase in generalization, particularly using our medium selection pressure settings (see Figure 12).

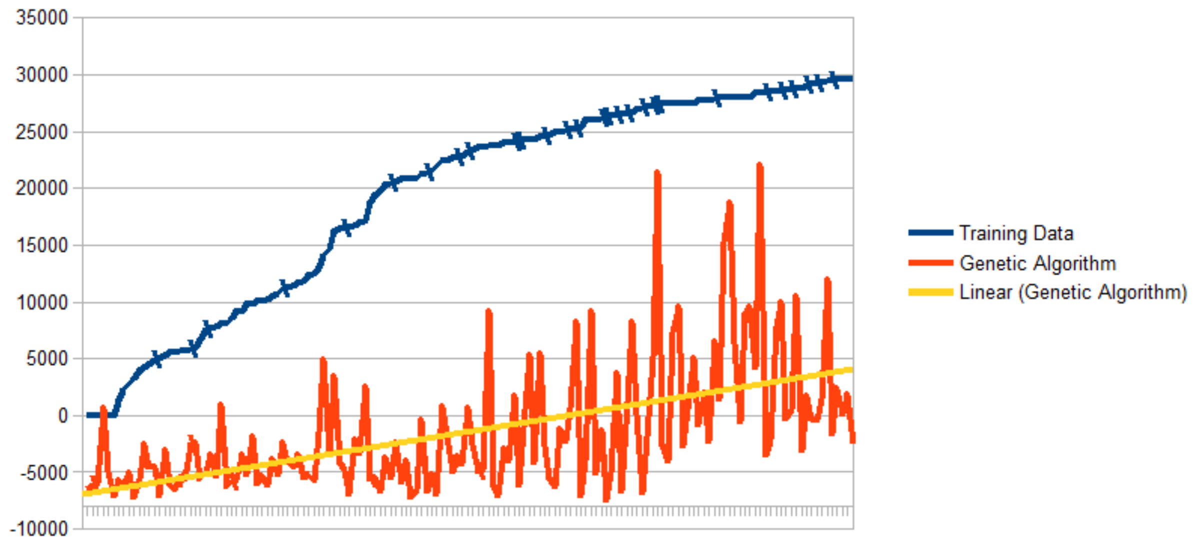

Figure 12. GA performance using a larger training dataset resulting in improved performance using medium selection pressure settings

Finally, we investigated whether there was a risk versus reward trade-off. In other words, do stocks with larger standard deviations lead to larger profits? This hypothesis was affirmed by testing all of our distinct profitable strategies obtained from all GA and Hill-Climber runs across using 4 randomly chosen test stocks (see Fig. 13). Each point on the following graphs represents a single strategy which was tested 20 times and from which we extracted a mean profit and standard deviation.

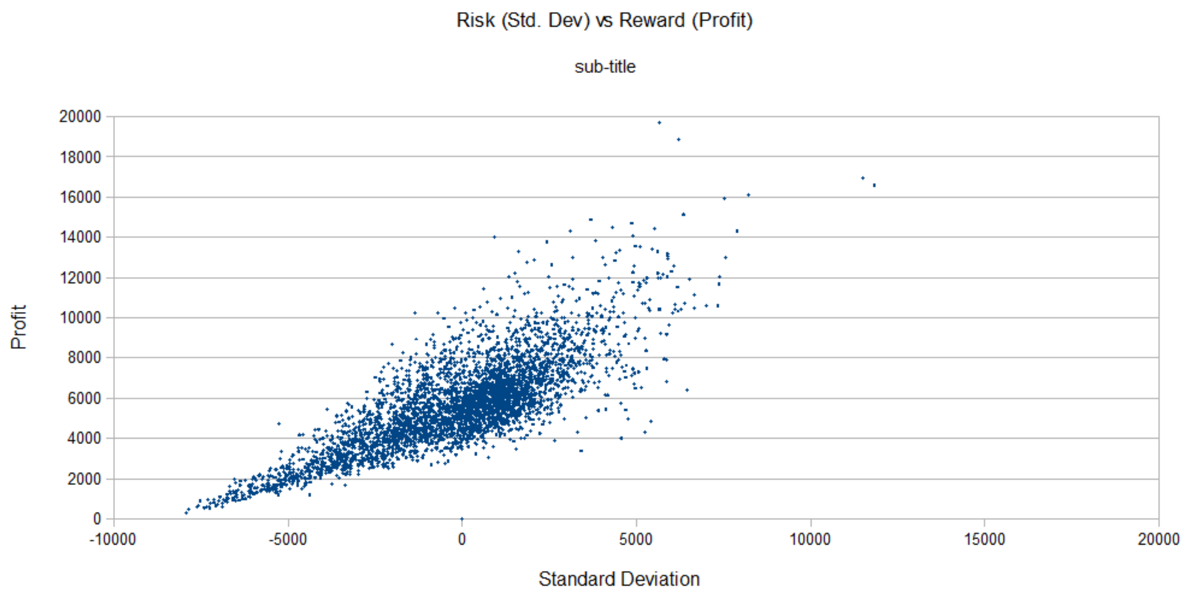

Figure 13. Standard deviations positively correlate with profit

We next calculated the Sharpe ratio for these stocks (see Figure 14). As expected, most of the time, a higher ratio corresponded to a higher profit. We chose our 'best trader' from visually inspecting this graph – we chose a strategy that was considered the most reasonable trade-off between risk and reward.

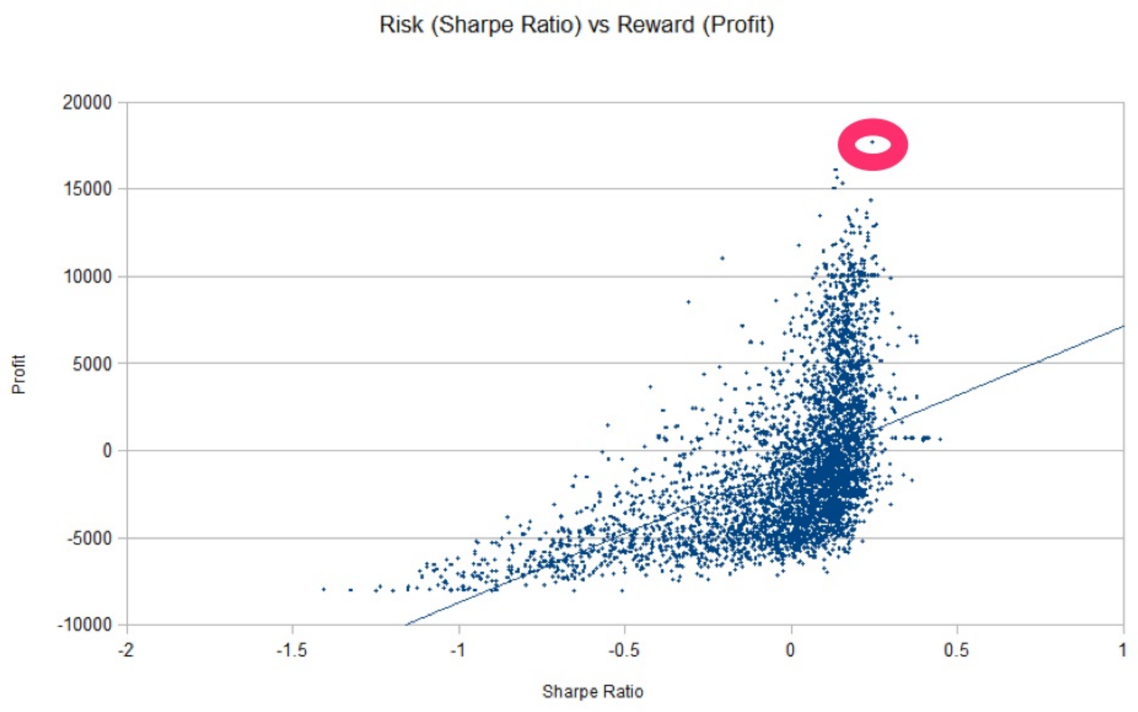

Figure 14. Sharpe ratios on all profitable strategies for test stocks
Note: The dot encircled by the pink ring is the strategy selected by the authors for their 'best trader'

Having selected our 'best trader' from the above graph, we backtracked through our data to discover which method (Hill-Climber or GA) produced this strategy. It was found that this strategy emanated from the low-to-medium selection pressure GA that had been trained on 13 stocks (see Figure 15).

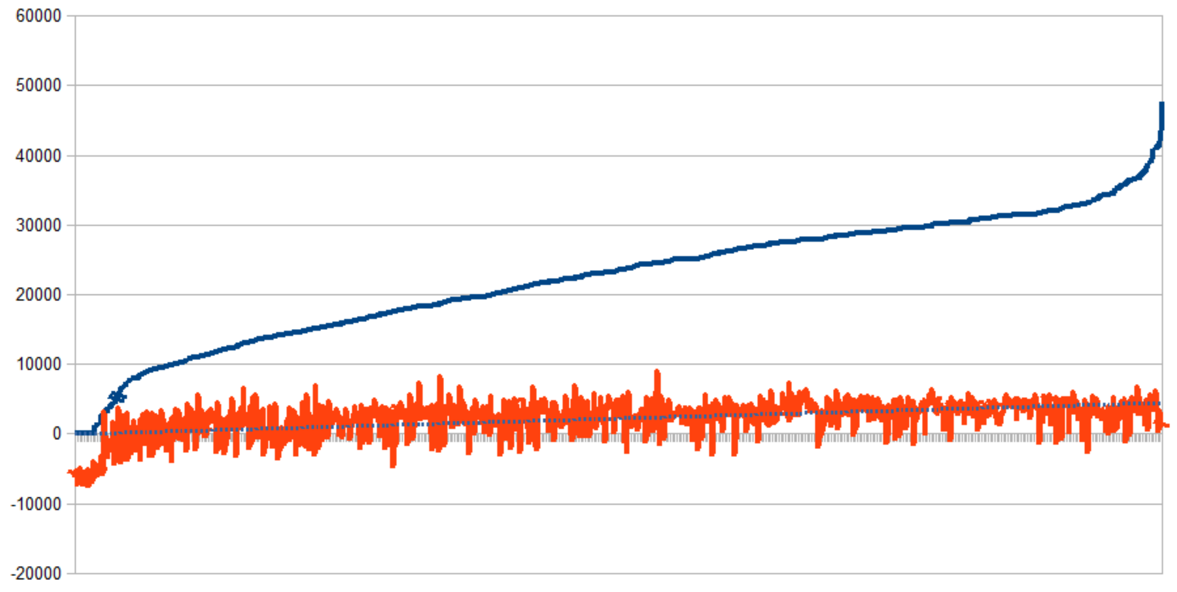

Figure 15. GA with low-to-medium pressure on 13 training stocks

We next performed a correlation coefficient to confirm that generalization did occur, and demonstrated a Pearson coefficient of 0.574 which typically indicates a positive medium-to-strong relationship between profits (fitness) and their strategies (see Figure 16).

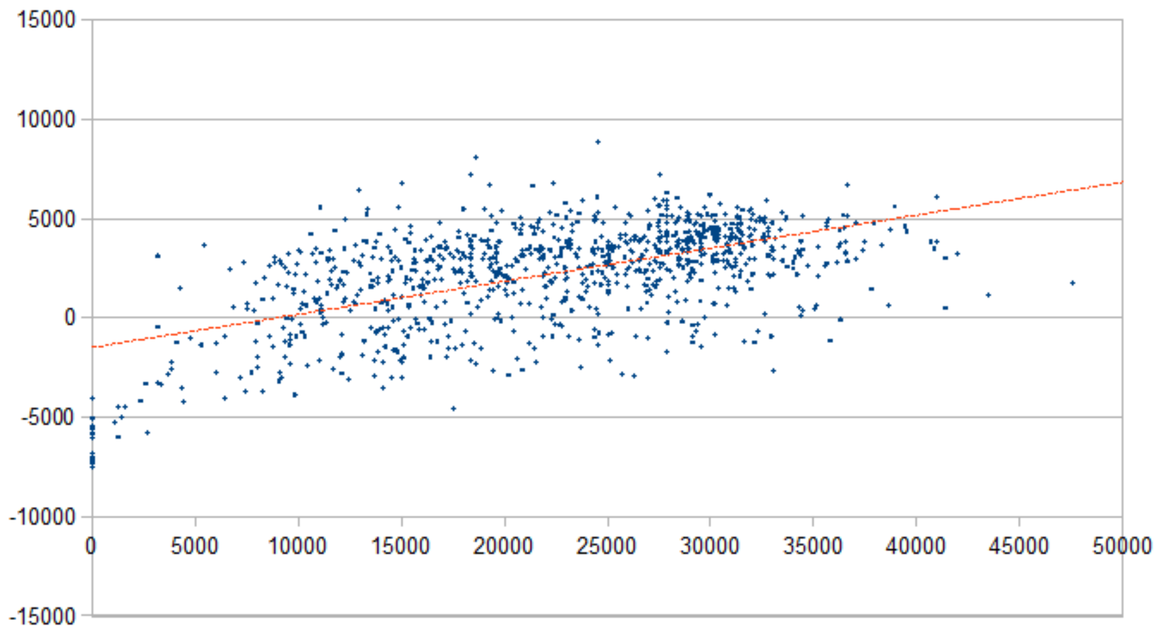

Figure 16. Scatterplot showing positive relationship between test data profit (x-axis) and training data profit (y-axis) for GA with low-to-medium pressure parameters trained on 13 stocks. Each dot corresponds to a trading strategy.

Finally, to better compare the strategies returned from the Hill-Climber and those evolved using the GA, we obtained the maximum, minimum, and average profits earned by the traders in each of five selection pressure tests. We present this data in Table 2 below.

Table 2. Performance of Genetic Algorithms with different selection pressures

	High	**Low**	**Medium**	**LowToMed**	**LowToHigh**
Maximum Profit	$840,721.95	$587,700.39	$518,320.94	$1,281,100.84	$1,281,100.84
Minimum Profit	$0	$1,070.94	-$1301.01	$5,211.05	$348.75
Average Profit	$475,761.15	$114,879.50	$302,383.14	$884,320.90	$1,121,210.09

In the next section, we discuss the implications of our results on our understanding of genetic algorithm and hill-climbing methods for implementing automated trading systems.

General Discussion

In the present project, we attempted to develop a robust, profitable automated trading system. We demonstrated that although the algorithms used were extremely powerful, particularly the genetic algorithm, they were also susceptible to over-fitting, and require great skill and expertise to master. While both algorithms performed well on training data, they both underperformed significantly on test data, which indicated a lack of generalization. The genetic algorithm produced substantially better solutions (higher profits) than the Hill-Climbing solutions, but we must demur on inferring bold conclusions, as both algorithms were not highly generalizable. We tried several strategies to remedy this over-fitting – training our algorithms on more (up to 13) training stocks, choosing to randomly buy one stock out of many when it was possible to buy many stocks on a particular day (see Figure 1), focusing on the earlier generations in our analyses when over-fitting would not have been a substantive issue, sourcing external stock data as well as the data provided to improve performance on unseen data, and randomly choosing our test data and training data from a pool of stocks. However, there were only modest-to-negligible performance differences. Utilizing these strategies to improve generalization did provide some interesting insights, however, and did ultimately lead to us finding our 'best trader'. Indeed, our 'best trader' strategy came from our genetic algorithm, using the low-to-medium parameter settings, and having been trained on 13 stocks. The No Free Lunch (NFL) theorems, as posited by Wolpert and Macready (1997), may go some way toward an explanation for the difficulty of finding the optimal set of parameters for the GA – since there may well exist no one general set of parameters for all problems (set of stocks) – and this may have further contributed to our poor performance regarding generalization. Future directions for these authors to augment their understanding of such algorithms may then include using more scientifically rigourous methods to find optimal GA parameters, such as using a meta-GA on top of a GA, and investigating adaptive GAs that adapt mutation and crossover probabilities (Srinivas & Patnaik, 1994), as well as population sizes online (i.e. during the GA's execution). These are just a few of the fruitful avenues for research and experimentation of optimization techniques in the field, which the present authors should endeavor to unpack.

Conclusion

The genetic algorithm, with low-to-medium selection pressure, demonstrated reasonable generalization in this project. Considering that the authors used only one training feature (closing price) and strategies based on just 5 consecutive days, this bodes well for developing more comprehensive training procedures. Increasing generalization is a hotly debated topic in the evolutionary programming community, but this simple project demonstrates that it is certainly achievable.

Bibliography

Brownlee, J. (2011). *Clever algorithms: Nature-inspired Programming Recipes*. Jason Brownlee Copyright.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: The University of Michigan Press.

Potvin, J., Soriano, P., & Vallee, M. (2004). Generating trading rules on the stock markets with genetic programming. *Computer & Operations Research, 31*, 1033-1047.

Srinivas, M., & Patnaik, L. M. (1994). Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics, 24*, 656-667.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation, 1*, 67-82.