Evolutionary Refinement of Trading Algorithms for Dividend Stocks

Robert E. Marmelstein, Bryan P. Balch, Scott R. Campion, Michael J. Foss, Mary G. Devito
Department of Computer Science,
East Stroudsburg University
East Stroudsburg, Pennsylvania, USA

Abstract - This paper describes a Stock Strategist application which can mine, test, validate and refine complex equity trading strategies based on historical financial data. We address the capabilities and operation of the program and detail the currently implemented strategy to trade stocks that pay high dividend yields. Additionally we will present preliminary results which show the ability of the system to automatically refine a specific trading strategy using genetic algorithms. The methodology of the test will be discussed and the results of two variant genetic algorithm approaches will be compared and contrasted.

KEY WORDS Data Mining, Simulation, Algorithmic Trading, Genetic Algorithms.

I. INTRODUCTION

This paper describes a Stock Strategist application that is designed to evaluate and refine specific stock trading strategies. There is definitely a very large body of academic work on the application of data mining techniques to discover and validate patterns in the stock market [1]. For this project, we decided to pursue a strategy focused on trading strategies for stocks that pay substantial dividends due to their predictable behavior both before and after the Ex-Dividend (Ex-Div) date. Further, we restricted our stock pool to those stocks which historically have a high dividend yield (6% or more).

A dividend is a payout of profits to owners (shareholders) of a company. The dividend is some percentage of the stock price, typically anywhere from 1% to 12% annually. Most companies that pay dividends do so quarterly. Unlike like certificates of deposits or bonds (where earned interest is proportional to holding time), companies will pay the entire dividend to whoever holds the stock on the Ex-Div date. As a result, the price of a stock tends to rise in the lead up to a dividend payment; often the price will rise by significantly more than the amount of the dividend payment itself. After the Ex-Div date, the stock drops sharply, sometimes by much more than the dividend amount. In these cases, the stock price may recover quickly from the typical Ex-Div drop.

Figure 1 illustrates this pattern for the stock of Vector Group Limited (symbol: VGR). There are two basic strategies for trading the dividend event. The first is riding the wave on the way up and selling at, or near, the predicted peak. The other is buying right after the Ex-Div drop in anticipation of a quick price recovery.

II. SYSTEM DESCRIPTION

The following paragraphs describe how the Stock Strategist program operates for a given strategy. Strategies are implemented as a composite set of C# classes that conform to interface specifications that facilitate extensibility, as well as, support for multiple strategies and comparative analysis. A strategy can be compiled, linked, and executed if it is written according to an Interface specification.

A. Preliminaries

Before running the Stock Strategist, the user must specify the following information:

- **Stock Pool** – The set of stock symbols of interest to the current strategy. Only stocks in the pool can be purchased for the asset portfolio.
- **Start/End Dates** – The data range over which a simulation will be run. Typically, the end date is the current date.
- **Starting Capital Balance** – The initial capital available to purchase portfolio assets.
- **Transaction size** – The minimum size of a buy transaction. Larger sizes result in less diversity and more risk, while smaller sizes diversify more at the expense of using capital for less attractive positions.
- **Commission amount** – This is the brokerage fee associated with each buy and sell transaction.

**B. Modes**

The Stock Strategist has four basic modes: Data Acquisition, Opportunity Identification, Backtesting, and Strategy Refinement. The Data Acquisition mode updates the database for those stocks in the stock pool. Data is incrementally extracted from several financial websites. The Opportunity Identification mode identifies opportunities for purchase within some time window from a specified date. Only stocks within the stock pool are eligible for this evaluation. The Backtesting mode evaluates the performance of a given strategy over a [past] data range. Lastly, the Strategy Refinement mode applies a search technique to improve the performance of a given strategy. This is typically done by optimizing the rule weights for scoring a given candidate stock. Note that these modes are not mutually exclusive. For example, the Strategy Refinement mode makes use of the Backtesting mode which, in turn, makes use of functionality in the Opportunity Identification mode.

**C. Backtesting and Decision Cycle**

As stated above, the Backtesting mode evaluates how a given strategy would have worked over a selected time period in the past. To accomplish this, a decision cycle is applied to the portfolio on a weekly basis from the specified start date through the end date. The purpose of this cycle is to make buy/sell decisions based on the strategy. The success of the strategy is gauged by the amount of portfolio net profit achieved over the test period after subtracting out transaction fees.

Per Figure 2, the Stock Strategist looks back one week to determine what holdings in the current portfolio should have been sold based on calculated price targets. If these price targets are not met, the stock is ultimately “sold” the day prior to the Ex-Div using the opening price for that day. The gain (or loss) from these sales are added to the current capital balance. Potential buying opportunities for the next week are then evaluated, scored, and then sorted by score in descending order. Using the available capital balance, stock purchase orders are initiated (per the specified transaction size) by descending order of score until the available balance is below the minimum transaction. Any remaining balance is then used to increase the transaction size of the order with the highest scored stock. This approach ensures that the available capital is always fully utilized and limits exposure to individual stocks by having a diverse set of positions. The
detailed steps in the weekly decision cycle are shown below in Figure 3.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Identify candidates for purchase – These are based on known or predicted trigger events in the strategy. Example of trigger events include: dividend declarations, earnings reports and stock splits.</td>
</tr>
<tr>
<td>2.</td>
<td>Score candidates for purchase – The specific scoring criteria is strategy dependent.</td>
</tr>
<tr>
<td>3.</td>
<td>Identify current assets for sale – These are strategy dependent but can be based on stop-loss or date triggers, percentage gains, and relative value compared to candidate buy positions.</td>
</tr>
<tr>
<td>4.</td>
<td>Unload existing assets – Assets are sold once their sell trigger conditions are met. Again, these conditions are strategy dependent.</td>
</tr>
<tr>
<td>5.</td>
<td>Buy new assets (based on candidate score).</td>
</tr>
<tr>
<td>6.</td>
<td>Compute portfolio value.</td>
</tr>
<tr>
<td>7.</td>
<td>Advance simulation time – Advance by one week. Exit loop if past specified end date.</td>
</tr>
<tr>
<td>8.</td>
<td>Remove expired [buy-related] events.</td>
</tr>
</tbody>
</table>

Figure 3. Decision Cycle Steps

III. SYSTEM COMPONENTS

The following paragraphs describe the major components of the Stock Strategist system.

A. Automated Data Acquisition

This component automatically extracts (“scrapes”) stock data from a number of investment-oriented websites. From a web programming standpoint, extraction of this data requires a two-step process. Like many contemporary websites, the sites above utilize AJAX to produce dynamic content. As such, attempting to extract information requires reviewing the communication data flow that results in the final displayed page content. This can be achieved using developer browser extensions. When a desired data element of the displayed page is associated with a web resource, the URL of the resource is noted. In most cases, the URL has to be constructed dynamically from an intermediate stock code embedded in the target website’s HTML. Once the URL has been constructed and data retrieved, a flexible HTML parser is used to transform the potentially malformed HTML into XHTML which allows the usage of XPATH queries to surgically extract the desired data. Post extraction, the data is then validated and the updates are saved in the database. To avoid unnecessary website bandwidth and processing time, the data acquisition process only performs incremental updates. Following a successful update, the stock is updated with a timestamp for the property that was retrieved, such as quote information or quarterly data. This allows the data acquisition process to operate as a system service or start-up routine for the application to maintain an up to date view of the market.

B. Common Framework & Database Layer

This component contains the classes for each major type of data object (e.g., real-time quote, stock, daily quote, quarterly data, etc.). It also provides the code for associated database stores and queries of these objects. Interactions with the database are performed through a common database provider, which currently targets MySQL. Other databases could be used with minimal code changes such as SQLite.

C. Asset (Portfolio) Manager

The Asset Manager component contains all functionality for maintaining the list of currently held assets. Each of the assets must be one of the stocks listed in the stock pool. During a given decision cycle, the Asset Manager also tracks the various buy and sell decisions made. At the end of each decision cycle, the entire value of the portfolio (cash and currently held assets) is computed. The Asset Manager has a number of wrappers for the identification and scoring of buy or sell candidates. These wrappers rely on method calls to the Strategy component class to accomplish their primary functions.

D. Analytics

This component computes technical indicators (features) and related statistics based on the historical stock performance. While these are independent of a given strategy, they are generated for input to the Strategy component. Analytics output fall into two general categories: features snapshot or statistical distribution. The snapshot is a feature’s value at a given point in time. With regard to the statistical distribution, the mean or standard deviation of the distribution is reported. For example, the Price to Earnings (PE) ratio of a stock may be the current PE ratio (snapshot) or the mean PE ratio (of a Normal distribution) over a specified time period.

Another key feature of the Analytics component is to predict stock-related events based on their past occurrence. For example, a strategy may require prediction of a date range when the declaration of a dividend will occur. Since dividends are paid quarterly, it becomes necessary to first cluster these events into quarters (using the K-Means method [2]) to determine the likely time window for this event in any future quarter. The capability to predict future events is critical to the Strategy component and distinguishes the Stock Strategist from most commercially available trading platforms.

Once these events are predicted, the Strategy component can set up triggers based on these predictions. In most cases, an event that is (first) predicted is later confirmed by the data. These transitions are discovered as a result of the automated data acquisition process. At this point, the software changes the status of the event from
predicted to known. When this happens, the event may be treated differently by the logic in the Strategy component.

E. Strategy

The Strategy component implements the logic behind the buy/sell decisions of the application. This component determines trigger conditions (for buying or selling) and scores candidate stocks for purchase. In general, Strategy component decisions are based on a series of distinct rules native to each strategy. The output of these rules are categorized, weighted, and summed to compute a single score for decision making purposes. The buy signal computation is shown below in Equation 1.

\[
Buy\ Signal\ (stk) = \sum_{i=1}^{n} rule_i \ (stk) \ast w_i \quad (1)
\]

As previously stated, these rules are supported by selected, lower level snapshot features and statistics available from the Analytics component. For example, one of the rules computes the ratio of the current change in price (after the dividend declaration date) to the historic (mean) change in price after that event. Once computed, the goodness of that ratio is categorized (good, acceptable, or poor) and a weight is applied to that rule. The Strategy component implements a standardized set of classes and methods. This design makes it straightforward to plug different equity trading strategies into the Stock Strategist framework.

F. Simulation

The Simulation component is an essential capability for backtesting a given strategy. It runs the decision cycle once per week over a given date range. After each cycle, the simulation time is advanced forward by one week. This component also logs the weekly portfolio value and all asset buy/sell transactions that occur over the simulated time period. The most critical ingredient of the Simulation is having sufficient and accurate data for a given date range. As previously noted, this task is accomplished by the Automated Data Acquisition component. The online sources we rely on only provide summary data for any given data; intra-day data is only available in real-time and is not stored in the database. The practical impact is that the simulation rarely utilizes real-time current data.

G. Search

The purpose of the Search component is to refine the current strategy in a way that increases (maximizes) portfolio return. The Search is performed with respect to key parameters for a specific strategy. For the most part, these parameters correspond to the strategy rule weights discussed in Section 4E. However, they can also include other parameters, such as the minimum Transaction Size and the size of the time window used for predictions. To conduct the search, these parameters are then passed as input to a given simulation instance. The simulation is then executed and the ending portfolio balance is passed back to the search routine. The fitness of the parameter set is computed simply as the ratio of the ending to starting balances (see Equation 2); the higher the ratio, the greater the fitness.

\[
Fitness = \frac{Portfolio\ Value\ (\$) - Starting\ Portfolio\ Value\ (\$)}{Ending\ Portfolio\ Value\ (\$) - Starting\ Portfolio\ Value\ (\$)} \quad (2)
\]

Although the parameter set optimized by the Search component is somewhat strategy dependent, it can be easily reconfigured. Currently, the search techniques utilized are Hill Climbing and evolutionary search using a Genetic Algorithm (GA). Unlike Hill Climbing algorithms, GAs can escape local optima traps [3]. While the GA is a highly effective technique for global search, the biggest impediment to its use is the amount of time to evaluate each generation, even with a small population. This is because each chromosome is a variation in trading strategy that takes several minutes to simulate. Thus, accelerating the Search component without degrading the fitness metric has been one of our top research priorities.

IV. EVOLUTIONARY APPROACH FOR BUY SIGNAL COMPUTATION

Researchers have long sought automated “black box” techniques to identify and trade financial instruments of all varieties. Applying GAs to the evolution of trading strategies is not new—there is a solid body of work in this area. Allen and Karjalainen [4] were among the first to develop a GA-based system for finding technical trading rules. Becker [5] expanded on this using a Genetic Programming (GP) approach and an emphasis on monthly (vs. daily) trading. Schoreels [6] employed GAs to design agent-based systems for trading. Subramanian [7] developed a similar agent-based system, but with an emphasis on reducing trade risk and volatility. In contrast to the focus on trading rules, other researches including Yang [8], Lin [9], and Lai [10] utilized GAs for the selection and optimization of stock portfolios.

The key component of these systems is the signal to buy or sell a given security. The goal of our experiments was to evolve a set of weights (W) for each rule component to improve the performance of the buy signal for our dividend stock trading strategy. In this case, the size of the weight set was 12. Two sets of weights were evolved. The first set (W_{0,1}) had a range of (0, 1). These weights serve to select the rule components of the Buy signal as indicated in Equation 1. The second set (W_{0,5}) has a range of (0, 5). These can both select and amplify the contribution of a given rule to the overall Buy signal. As indicated earlier, the fitness of a given weight set is determined by the overall portfolio return.

In this experiment, each type of weight set was evolved over a simulated one year period for eleven periods (2001-2011). At the beginning of each one year training period, the Stock Strategist has an initial balance of $200K to trade with. After each training period, the best performing
set of evolved weights were evaluated on two sets test scenarios:

*Portfolio Continuation (PC)* – The evolved weights are used to continue run the simulation where the training left off. Thus, the trading continues over a follow-on trading period (1 and 6 months, respectively) with the portfolio and balance that remained at the end of the trading period. Given the dynamic nature of the modern financial markets, it was judged that six (6) months was the maximum amount of time to safely continue using a set of weights before they became stale.

*Portfolio Restart (PR)* – The evolved weights are used to start a new trading simulation in the follow-on trading period (one and six months, respectively). In this case, however, the strategist starts from scratch with a $200K balance and no pending transactions. Thus, any decisions made during training (beyond the evolved weights) are forgotten.

The GA library used for this experiment was part of the AForge.NET Framework [11]. A population of 30 chromosomes was utilized over 50 generations. Elitist selection was used with a crossover rate of 0.75 and a mutation rate of 0.01.

V. RESULTS

Figures 4 and 5 show the results for the $W_{0.5}$ and $W_{0.1}$, respectively. These bar charts show the relative performance of each weight set vs. those of the default weight set ($W_D$). In $W_D$, the Buy signal is computed with each rule selected and given a unity weighting.

These results are noteworthy in two respects. The PC scenarios dramatically outperform PR scenario. In fact, the performance of the PR scenario is not much better than the default approach for computing the Buy signal. This indicates that the evolved rule weights are much better suited to running the existing portfolio versus starting over with a new portfolio using the same general strategy.

The $W_{0.5}$ variant evolves a solution that performs much better during training than the $W_{0.1}$ variant. This is not at all unexpected since the $W_{0.5}$ variant offers a much bigger search space. Unfortunately, this advantage does not hold up during testing. Indeed, the $W_{0.1}$ variant is much more consistent than the $W_{0.5}$ variant’s performance (training vs. test) on the PC cases. This is likely due to overlearning on the $W_{0.5}$ variant. Thus, while the $W_{0.1}$ variant is simpler, it is also more powerful due to its ability to better generalize its performance into the future.

VI. CONCLUSION

In this paper, we have introduced the Stock Strategist application. Initially, we are using this program to pursue a dividend stock trading strategy. We chose this strategy due to its relative simplicity, stability, and predictability vs. the myriad of other possible equity trading strategies. Our experiment in automatically refining our trading strategy yielded some interesting results in how to effectively structure GA-based training and apply the results. In particular, it indicates that simple, 0-1 rule selection is a better alternative to more complex weighting factors. The experiment also suggests that the evolved weights be used to continue the existing portfolio momentum, rather than restarting the portfolio. We have also experimented with techniques to accelerate the GA search and made substantial progress in this area (to be documented in an upcoming paper). Our future research seeks to expand the GA to cover additional algorithm parameters (such as the nominal size of a given stock buy) and strategies (such as using a limit strategy when initiating a stock purchase).

REFERENCES


