T H E N A T I O N A L U N I V E R S I T Y
of S I N G A P O R E

S c h o o l o f C o m p u t i n g
Lower Kent Ridge Road, Singapore 119260

TRC1/05

Technical Indication Generation = Trend Classification + Genetic Algorithm

Bernard YAP Chur Jiun and Siau-Cheng KHOO

January 2005

TechnicalReport

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

JAFFAR, Joxan
Dean of School

Technical Indicator Generation = Trend Classification + Genetic Algorithm

Bernard Yap Chur Jiun, Siau-Cheng Khoo
{yapchurj,khoosc}@comp.nus.edu.sg

Department of Computer Science
National University of Singapore

1 Abstract

Technical indicators are used to interpret stock market trends and make investment decisions. The main difficulty of its usage lies in the fact that it is extremely hard to reliably construct an instance of any technical indicators with appropriate parameters for a particular series of stock prices. While genetic algorithm has been used to discover technical indicator in the past, not much justification have been given for such an approach.

In this work, we look into the basic assumptions of the working of technical indicators, and the suitability of applying genetic algorithms for indicator discovery. We argue for the use of technical indicator as an approximation to the ideal solutions for daily trading. Consequently, we propose a framework that applies genetic algorithm to discover good parameters for any technical indicators to work well on their respective targeted price trends. A widely used technical indicator, Exponential Moving Average Crossover, is used to illustrate how the proposed framework may be employed to construct good instances of the technical indicator for a targeted price trend.

1.1 Subject Descriptors

D.2.2 [Software Engineering] Evolutionary Prototyping

H.4.2 [Information System Application] Decision Support

1.2 Keywords

Technical Indicator, Genetic Algorithm.

Contents

1 Abstract **1**

1.1 Subject Descriptors 2

1.2 Keywords 2

2 Technical Indicator **6**

2.1 Difficulties Of Technical Indicators Usage 6

2.2 Exponential Moving Average Crossover 7

2.2.1 Exponential Moving Average 7

2.2.2 Crossover And Trade Signals 8

2.3 Trends And Technical Indicators 8

3 Trend Classification **10**

3.1 Motivation For Trend Classification 10

3.2 Brief Description Of Holt's Method 10

3.3 *t-series* As Measurement Of Trend Similarity 11

3.4 Trend Classification Using *t-series* 11

3.4.1 Defining Trend Similarity 12

3.4.2 The Classification Algorithm 12

4 Genetic Algorithm **14**

4.1 Background 14

4.2 Fitness Function 14

4.3 Chromosome Representation 14

4.4 The Crossover Operator 15

4.4.1 Boundary Crossing Crossover 15

4.5 The Mutation Operator 16

4.6 Selection Methods 16

4.6.1 Tournament Selection 16

5 Constructing Technical Indicators **17**

5.1 Problem Representation 17

5.2 Definition Of Fitness 17

5.3 Alternative Fitness Function 18

5.3.1 Defining Maximum Profit 18

5.3.2 Fitness, A Combinatorial View 18

5.4 Fitness Landscape 19

5.4.1 Dynamic Fitness Landscape 19

5.5 Search By Genetic Algorithm 20

5.6 Genetic Algorithm Implementation 21

5.7 Other Search Techniques 22

5.7.1 Simulated Annealing 22

5.7.2 Random Mutation Hill Climber 23

5.7.3 Parallel Random Mutation Hill Climber 23

6 Experiments And Results **24**

6.0.4 Legend Of Experiment Results 24

6.1 Experiment I 24

6.2 Experiment II 25

7 Summary **26**

8 References **27**

8.1 Conference papers 27

8.2 Books or reports 27

8.3 Unpublished reports and theses 27

List of Figures

1 Pseudo code of the classification algorithm 12
2 Boundary crossing crossover 15
3 Fitness landscapes of the same stock at different times 20
4 Pseudo code of the genetic algorithm implementation 23
5 Graph of GA/Max over test data length 26

List of Tables

1 Specification of genetic algorithm implementation 22
2 Experiment I results 25
3 Experiment II results 25

2 Technical Indicator

Technical indicators may be seen as a series of data points derived from the stock price data by applying certain formulae. Technical indicators are often used to alert, confirm, and predict trends with the help of other stock trading tools. In conjunction with appropriate rules, technical indicators can be used to generate trade signals, such as *Buy* and *Sell*, for making trading decisions. Examples of technical indicators includes Exponential Moving Average Crossover, Random Walk Breakout, and, Moving Average Convergence Divergence.

Making stock trading decision using technical indicators is a form of price prediction, as decision making in stock trading eventually boils down to accurate prediction of future trends. However, it is important to note that the basis of the working of technical indicators for stock price prediction involves the following assumptions.

Assumption 1 *The stock price series is not completely random, or otherwise no prediction is possible.*

Assumption 2 *The technical indicator used is capable of predicting trends in the price series in question.*

The assumptions above are the very basis of the justification for any attempt to stock price prediction in general and technical indicator in particular. While they are regarded as assumptions, the above statements are well accepted with the active development of various techniques on stock forecasting serving as the evidence.

In the context of this work, we further assume the following to reduce the complications involved in reasoning about stock trading. Nevertheless, we believe that the idea developed in this work can be extended to cover situations with more complex decisions.

Assumption 3 *When trading stock, only the following decisions are possible. (1) Buy the maximum possible amount of stocks. (2) Sell all possessed stocks. (3) Do nothing (also known as "hold").*

2.1 Difficulties Of Technical Indicators Usage

The main difficulty of applying technical indicators to stock trading is about constructing instances of technical indicator with good parameters. The

difficulties concerning the construction technical indicator instances may be summarized as follow.

- The search space is exponentially large. The number of possible technical indicator instances is given as R^N where R is the range of each parameter and N is the number of parameters.
- The fitness landscape of the search space is not well defined (discussed in Section 5.4), and thus no efficient systematic search techniques may be used in the search for good instances.
- While the abstract definition of good technical indicator instances is straight forward, the technical definitions remain elusive.

2.2 Exponential Moving Average Crossover

A widely used technical indicator, exponential moving average is often employed as a prediction or verification tool for changes in stock trends. This section shall give a brief overview of this technical indicator.

2.2.1 Exponential Moving Average

Exponential moving average is superior over simple moving average in the sense that it gives more weight to recent data values. It is therefore more responsive to recent trends, as the influence of past data values fades exponentially as we move forward in time. Let $P \geq 1$ be the period of exponential moving average and v_i the data value on day i, the exponential moving average on day i, referred to as ema_i, is defined as follow.

$$f = \frac{2}{1+P} \tag{1}$$

$$sma_i = \frac{\sum_{k=0}^{P} v_{i-k}}{P} \tag{2}$$

$$ema_i = v_i \times f \ + \ sma_{i-1} \times (1-f) \tag{3}$$

2.2.2 Crossover And Trade Signals

The idea of the exponential moving average crossover indicator is to have two exponential moving averages (ema^L and ema^S) of different periods, P_L and P_S. The crossover of the two exponential moving averages indicates a switch in trend and generates trade signals. Trade signals generated at the point of crossover are given below, with $LONG$ representing the *Buy* signal and $SHORT$ the *Sell* signal.

$$signal_i = \begin{cases} LONG & \text{if } ema_i^L < ema_i^S \wedge ema_{i-1}^L > ema_{i-1}^S \\ SHORT & \text{if } ema_i^L > ema_i^S \wedge ema_{i-1}^L < ema_{i-1}^S \end{cases} \tag{4}$$

Where ema_i^L and ema_i^S represent the ema of day i forP_L and P_S respectively.

2.3 Trends And Technical Indicators

Drawing from the assumptions introduced in Section 2, we shall investigate the relationships between trends and the working of technical indicators. The purpose of using technical indicators in stock trading is, of course, to gain profit. It is observed that the maximum profit is gained by *selling* at every peak and *buying* at every trough. Therefore, stock trading is, at its core, to accurately predict the changes in trends, or in the other words, the peaks and troughs of the price series.

For the detection of peaks and troughs, the technical indicator used must be able to give information about current trends. That is, the technical indicator must be able to tell if the price series is currently in the beginning, middle, or the end of an upward/downward trend. Whatever method is employed by a technical indicator, it is essentially making prediction about future price data based on the past price data. A closer look at technical indicator in general revealed that technical indicators draw conclusions about the trends of past price data.

We shall now introduce an intuitive notion of trends in stock price data. Putting aside any technical definitions of trend, it may be intuitively viewed as a reproducible characteristic in the data series. What a technical indicator does, distilled, is nothing more than detecting (or in some cases, assuming) trends in the past price data, and projecting future price values accordingly.

Theorem 1 *For any technical indicator, we can always construct a price series that does not conform to the trend established by a particular technical indicator.*

Technical indicators make prediction solely based on past data and have no knowledge of the future data at all. Therefore, the proof for Theorem 1 is a simple one, as a conflicting price series can be easily constructed by starting with a conforming price series which conflicting price values are added to and its length increased as necessary. The proof is possible because being a specific trend, it cannot cover all possible price series otherwise there is no reason of talking about trends in the first place.

Corollary 1 *For all technical indicators, there exist price series which it cannot accurately gives the correct decision*[1] *for every price value in the series.*

In the other words, there is a limit to the expressive power for all technical indicators. Corollary 1 follows directly from Theorem 1 because if a price series conflicts with the defined trend then the correctness of decisions made based on the trend about the conflicting price series is not ensured. The above findings are further supported by experiment results presented Section 6.2.

[1] *Buying* at peak and *selling* at trough. *Hold* otherwise.

3 Trend Classification

3.1 Motivation For Trend Classification

The motivation for trend classification is to establish some relationships between the training and testing data used for evaluating technical indicators constructed with genetic algorithm. Without any concrete relationships, the relevance of the technical indicators constructed from the training data to the testing data is just doubtful.

In the other words, it is not possible to assess if technical indicators that work well on the training data will work well on the testing data, except by applying them on the testing data. Thus, a good solution when evaluated against the training data may turn out to be a bad one when evaluated against the testing data. This is highly undesirable, as it disrupts the fundamental requirement for consistency of the training and testing evaluations.

With trend classification in place, the training and testing data may be arranged such that they are different and yet exhibit the same trend. Given the relationships between trends and technical indicators discussed in Section 2.3 earlier, we may draw the conclusion that technical indicators that work well on the training data will work well on the testing data as well.

3.2 Brief Description Of Holt's Method

Holt's method, also known as single exponential smoothing, is often used to extract trends from data that is believed to contain trends but not cyclic patterns. It is defined by the following equations that give a level series (y') and trend series (t) or *t-series* from the original data series y.

$$y'_k = \alpha\, y_k + (1-\alpha)\,(y'_{k-1} + t_{k-1}) \quad [k > 0] \tag{5}$$

$$t_k = \beta\,(y'_k - y'_{k-1}) + (1-\beta)\, t_{k-1} \quad [k > 0] \tag{6}$$

Due to the recursive nature of the above equations, it is not possible to evaluate the equations without assuming a few starting values, namely y'_0 and t_0. In this work, the following assumptions are made about the starting values.

$$\begin{aligned} y'_0 &= y_0 \\ t_0 &= y_1 - y_0 \end{aligned} \tag{7}$$

Consequently, subjecting to the values of α and β, the first elements of the series y' and t may or may not be accurate due to the assumed starting values. As the influence of the assumed values fade, because of the exponential nature of the equations, the y' and t series may then be considered accurate.

$$\alpha = \beta = 0.5 \tag{8}$$

It is worth mentioning that, in this work, the Holt's method parameters α and β are defined in Equation 8 above. Minimal effort was made to adjust the parameters due to the fact that the initial guess of 0.5 serves the purpose for trend classification just nice. However, the parameters may be adjusted as we impose stricter conditions for the notion of similar trends in our future works.

3.3 *t-series* As Measurement Of Trend Similarity

The key observation about doing trend classification by applying Holt's method is that the *t-series* gives the trend information. To be more precise, each element in the *t-series*, (t_k), gives information about how much does the data point y_k increases or decreases from y_{k-1}.

It is clear that, assuming identical α and β values, different data series with identical trends will result in identical *t-series*. A close look at Equation 6 shows that any element t_k in a *t-series* is only dependent on the previous element t_{k-1} and the difference between the data points y_k and y_{k-1}. Therefore, identical *t-series* can only be the result of two data series with identical trends (or trend), as the computation of *t-series* does not depend on anything else.

Without any loss of generality, it may be stated that similar *t-series* implies similar data series. Thus, we concluded that the *t-series* of a data series computed by Holt's method is a suitable measurement of trend similarity.

3.4 Trend Classification Using *t-series*

The *t-series* computed using Holt's method is used as a measurement of trend similarity for the purpose of classifying data series.

3.4.1 Defining Trend Similarity

Two data series of the same length are considered to have similar trends if and only if the *mean absolute error (mad)* of the *t-series* corresponding to the two data series is less than or equal to the error e. With y_n as the original data series and t_n the *t-series*.

$$similar?(y_1, y_2) = true \longleftrightarrow mad(t_1, t_2) \leq e \tag{9}$$

Where,

$$\begin{aligned} &length(t_1) = length(t_2) \\ &mad(t_1, t_2) = \sum_{i=1}^{length(t_1)} |t_{1i} - t_{2i}| \\ &e = 0.05 \times length(t_1) \end{aligned} \tag{10}$$

3.4.2 The Classification Algorithm

```
function classify( src, y )
{
    i = 0
    sz = length(src)
    similar = {}
    t_src = t_series(src)
    t = t_series(y)
    while ( i < length(y)-sz ) {
        if ( mad(t_src,t[i:i+sz]) <= sz*0.05 ) {
            similar = similar + { y[i:i+sz] }
            i = i + sz
        } else {
            i = i + 1
        }
    }
    return similar
}
```

Figure 1: Pseudo code of the classification algorithm

Given a source trend pattern src of length l_{src}, and a data series y. The *t-series* of both src and y are computed before t_y is searched for non-overlapping sub-series of length l_{src} that are *similar* to t_{src}. The *similar* sub-series found are classified to have the same trend as src. Pseudo code of the classification algorithm is given above as the function *classify* that returns the set of non-overlapping data series of similar trend.

4 Genetic Algorithm

This section briefly discusses some aspects of genetic algorithm in general that are relevant to this work. The reader may refer to Reference [9] for a good introduction to genetic algorithm.

4.1 Background

Evolutionary algorithm was inspired by the theory of natural selection[2] and evolution, which is usually referred to as the theory of "the survival of the fittest". Evolutionary algorithm, in general, employs computer science techniques inspired by biological mechanisms such as natural selection, recombination, and mutation.

Genetic algorithm is a class of evolutionary algorithm that is often used to find approximate solutions to difficult optimizing problems by applying the principles of evolutionary biology and genetic information exchange. The search space is represented as a population of abstract entities, called chromosomes, that are evolved towards better solutions by the use of various genetic operators, such as crossover and mutation.

4.2 Fitness Function

Fitness function measures the optimality of a solution represented by a chromosome. The optimality of a chromosome dictates its chance of producing offspring that are (hopefully) better. The ideal fitness function must be closely correlated to the defined optimality of the algorithm, and allows fast computation of fitness values.

The concept of fitness landscape is widely used in discussions of various search techniques to describe the search space. It visualizes the relationships between genotypes[3] and the fitness values of chromosomes, which indirectly represent the rates of reproduction in the case of genetic algorithm.

[2]Proposed by Charles Darwin and Alfred Russel Wallace in 1858.

[3]Here refers to the values or configuration represented by the chromosome.

4.3 Chromosome Representation

Linear representation is generally used to encode information in chromosomes. A chromosome (solution) is encoded as a set of values of variables, known as genes, that are arranged in the form of a list. This is traditionally done by using a bit string where each gene is represented by one or more bits. Higher level data structures, such as lists of floating point variables, have been shown to be equally effective, and employed in favour of its more intuitive data abstraction.

4.4 The Crossover Operator

Crossover is the computer science equivalent of genetic information exchange between chromosomes. Many different crossover methods are used in various different representations. The most common crossover method used with linear representation is the N-point crossover. Tree based crossover operators, such as sub-tree swapping crossover, is widely used in genetic programming[4].

4.4.1 Boundary Crossing Crossover

Traditionally, crossover is done at the bit level. The crossover is applied to bit strings representing chromosomes, as introduced in Section 4.3, without paying any attention to the boundaries between the genes. In the example that follows, the binary coded chromosome pair of $(2, 1)$ and $(1, 2)$ produces the pair of offspring $(2, 0)$ and $(1, 3)$ (each gene is 2-bit in length).

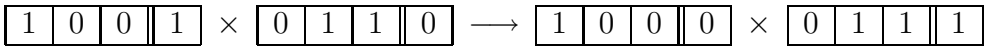

Figure 2: Boundary crossing crossover

The introduction of new values (0 and 3 respectively) in the offspring shows that boundary crossing crossover does more than exchanging information between chromosomes. It actually has the effect of mutation (see Section 4.5), which introduce new values into the produced offspring. With the above observation, we concluded that when non-boundary crossing crossover is used in genetic algorithms, the mutation rate should be increased to account for the loss of probability of introducing new values given by boundary crossing crossover.

[4]Genetic algorithm that uses tree-like representations for program optimization.

4.5 The Mutation Operator

The mutation operator, analogous to biological mutation, is often used to maintain genetic diversity in the population of chromosomes. From the searching perspective, it is an important tool to prevent the search from being trapped at a local maximum. Mutation usually involves reassigning one or more genes in a chromosome with random values, which has the possibility of introducing new values into the chromosomes.

The introduction of new gene values into the population of chromosomes is of utmost importance because, (1) The starting population may not contain all values needed in order to produce the optimal chromosome, mutation may introduce the missing values into the population. (2) In event of the genetic algorithm being trapped at a local maximum, mutation introduce new values into the population to help untrap the genetic algorithm.

4.6 Selection Methods

An important aspect of genetic algorithm involves the selection of chromosomes for producing offspring (reproduction). The chance of a chromosome being selected is directly proportional to its fitness as defined by the fitness function. A detail description of selection method in general is beyond the scope of this work, thus we shall give a brief account of *tournament selection* which is used in the genetic algorithm implemented in Section 5.6.

4.6.1 Tournament Selection

Tournament selection involves the random selection of S (known as the size of the tournament) chromosomes from the population as tournament participants. Tournament selection of size S is called S-way tournament selection in the context of this work. The tournament is simply the computation of the fitness of each participants, which are then used to select the best two participants for reproduction.

5 Constructing Technical Indicators

Constructing good instances of a particular technical indicator inevitably boils down to searching for good parameters. In this work, genetic algorithm is proposed as the search (or rather, optimizing) technique for constructing good instances of technical indicator in general, and the *exponential moving average crossover* indicator particular. This section shall detail the framework proposed for construction of technical indicator instances.

5.1 Problem Representation

An instance of any technical indicator is represented by a set of values corresponding to the parameters of the technical indicator in question. The set of values, known as the genes, are stored in a list data structure, known as the chromosome, in the form of floating point variables.

In this work, an instance of the *exponential moving average crossover* consist of two parameters (also known as genes), *long_period* and *short_period*[5]. Other possible parameters, including but not limited to *short_ratio*[6], are left out for simplicity. More parameters may be included in future works to further investigate the behavior of the indicator.

5.2 Definition Of Fitness

In search for good technical indicator instances, we shall define the fitness of a technical indicator instance. Let i be the technical indicator instance, and *data* be the data series against which the fitness is evaluated, the fitness of i is defined as the percentage of profit gained from trading (with starting capital k) by applying i over *data*.

$$fitness(i, data) = \frac{profit(i, data, k) - k}{k} \times 100\% \tag{11}$$

The above defines the primary fitness function used in this work for evaluating an instance of technical indicator. Due the many multiplication and division operations involved in calculating the profit, an alternative fitness function consist of only addition and comparison operations is introduced in

[5]Periods of long/short exponential moving average.

[6]Ratio of amount of stock to short-sell over amount of stock possessed.

later sections to speed up the search. However, the primary fitness (referred to as $fitness$) is used in the discussion of fitness landscape in Section 5.4 for the sake of clarity.

5.3 Alternative Fitness Function

5.3.1 Defining Maximum Profit

The maximum profit one can possibly gained from trading over an infinite price series is undoubtedly infinite. Therefore, it is only useful to define maximum profit over a fixed period of time. Let v be a price series of length l_v, the maximum profit over v by trading with starting capital k, is defined as the maximum net profit gained from trading over the price series with starting capital k.

Theorem 2 *Under Assumption 3, for any given price series of fixed length there is only one possible sequence of trading decisions that results in the maximum profit.*

The proof for Theorem 2 is given as follow. First, we show that substituting any element in the sequence of trading decision that leads to maximum profit (referred to as seq_{max} hereafter) with a different trading decision results in less profit at the point of substitution. From Assumption 3, it follows that seq_{max} gives the (non-negative) maximum profit possible for each point in the price series. It is thus clear that any substitution in seq_{max} results in less profit.

Second, we show that the loss of profit at any point in seq_{max} cannot be regained in later trading. At any point in seq_{max}, the available capital k' is strictly the sum of the starting capital k and all profit previously gained. Since the profit possibly gained is directly proportional to the available capital, it is not possible to gained more profit than that gained in seq_{max} if there are any loss of profit ($k' < k$) in previous trading. With that, Theorem 2 is verified in the context of this work.

5.3.2 Fitness, A Combinatorial View

In Section 5.2, the $fitness$ of technical indicator instances is defined according to the profit gained. In view of the complexity of the fitness function highlighted earlier, we shall transform the search for maximum profit into

the search for the sequence of trading decision that leads to the maximum profit or seq_{max} as defined in Section 5.3.1.

An alternative fitness function ($fitness2$) is introduced as the percentage of matches the decision sequence generated by a technical indicator instance against seq_{max}. While the maximum fitness value given by $fitness2$ is equivalent to that of $fitness$, it is not a drop-in replacement for the previously defined fitness function. It is trivial to show that there is a possibility that a technical indicator instance that gives higher fitness as defined by $fitness2$ generates less profit than one with a lower $fitness2$ value.

However, the fact that $fitness2$ may be computed a lot faster, as seq_{max} may be pre-computed, making it a good approximation of $fitness$ to be used in the repeating fitness evaluations in the search for good technical indicator instances. Search techniques such as genetic algorithm can run at a faster pace without significant trade-off in accuracy.

5.4 Fitness Landscape

Before we move on for an account of how good instances of technical indicators are constructed, we would like to characterize the fitness landscape (or search space) of the problem. We shall look at the fitness landscapes of *exponential moving average crossover* for data series of length 20 as examples.

As noted in Section 5.1, an instance of *exponential moving average crossover* is fully represented by two parameters, namely $long\_period$ and $short\_period$. Thus, its fitness landscapes may be conveniently visualized as two dimensional surface graphs (in which darker is better).

We argue that the fitness landscape for any technical indicator is not well defined. The main argument lies in the fact that while data series of stock price contain trends, the actual price values are subject to random noise. The $fitness$ of the technical indicator instances, as defined in Section 5.2, is solely dependent on the price values. It is clear that the individual fitness is subject to the randomness in the price values. The fitness landscape, being an aggregation of individual fitness values, is therefore random and thus not well defined.

5.4.1 Dynamic Fitness Landscape

The fitness landscape of a technical indicator over a particular stock is dynamic. That is, it changes as we move forward in time while maintaining a

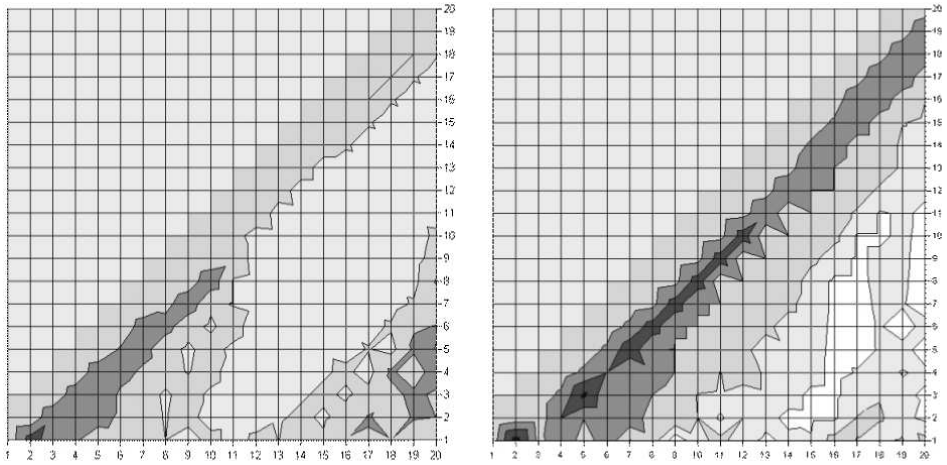

Figure 3: Fitness landscapes of the same stock at different times

window of fixed length. By observation, it is obvious that a stock exhibits different trends at different times (see figure 3). It follows that an instance of technical indicator that works well on a particular trend may fail when the fitness landscape changes. Thus, we argue that a good framework of technical indicator for stock trading must be adaptive to changes of the fitness landscape, and create new instances of indicator as needed.

To meet the adaptive requirement discussed above, the search algorithm for good parameters should be able to continue the search with its current states as the underlying fitness landscape changes to eliminate delays caused by restarts. Although experiments conducted in this work do not take into consideration the notion of dynamic fitness landscape, planned future works on real time trading using adaptive technical indicators shall be grounded on the concept of dynamic fitness landscapes.

5.5 Search By Genetic Algorithm

This section summarizes why genetic algorithm is used for the search of good technical indicator instances in this work. Genetic algorithm is capable of searching without prior knowledge of the fitness landscape. Given the random and constantly changing nature of the problem's fitness landscape discussed in the previous section, genetic algorithm's capability of search without prior knowledge makes it a very good candidate for this search problem.

With a fitness landscape that is both random and not well defined, it is safe to assume that the fitness landscape is filled with undesirable local

maxima. The mutation operator employed in genetic algorithms ensures that the search has a means to untrap itself in event of being trapped at a local maximum. The fact that genetic algorithm has large populations of potential solutions increases the chance of having potential solutions within the attraction basin of the global maximum.

Furthermore, genetic algorithm possesses the unique characteristic of being adaptive. It is most suitable for conducting searches in dynamic fitness landscapes (search spaces that changes with time) as detailed in Section 5.4.1. The adaptive characteristic comes from the fact that a vast amount of different chromosomes (potential solutions) are present at any time, provided that there are adequate mutations that prevent the population from converging completely. That way, no matter how the fitness landscape changes over time, it is very likely that there are chromosomes around the new global maximum.

In the case of *exponential moving average crossover*, a set of good parameters may be intuitively viewed as a combination of good parameter subsets, for a good solution corresponds to two exponential moving averages that characterize the long and short term trends of the data series. In the other words, good chromosomes are built out of good building blocks. Therefore, the crossover operation, which conforms to the *building block hypothesis*, is expected to speed up the convergence towards optimality. Unfortunately, there is insufficient evidence to show that the above is true for all technical indicator.

5.6 Genetic Algorithm Implementation

In this work, a genetic algorithm was implemented to investigate the feasibility of constructing good instances of technical indicator (*exponential moving average crossover*) via the search for good parameters using genetic algorithm. The specification of the genetic algorithm implemented is shown in the table below. The reason behind the rather high rate of effective mutation is discussed in Section 4.4.1.

The actual genetic algorithm implementation is presented in Figure 4 in the form of pseudo code. The line marked by (*) serves the purpose of maintaining the diversity in the population. With half of the new population consist of the best individuals while the others are among the worst, the population is prevented from converging entirely. Such behavior is needed to adapt to dynamic fitness landscapes as discussed in Section 5.4.1.

Chromosome representation	list of floating-point values
Number of genes	2
Population size	100
Termination criterion	after 10 generations
Crossover type	non-boundary crossing 1-point crossover
Effective Mutation rate	1/3
Fitness function	$fitness2$ (as defined in Section 5.3)
Selection method	4-way tournament selection

Table 1: Specification of genetic algorithm implementation

5.7 Other Search Techniques

Expectedly, there exist other search techniques that may be suitable for the search of good technical indicator parameters. This section shall review a few of the most notable alternatives.

5.7.1 Simulated Annealing

Inspired by annealing in metallurgy, simulated annealing is a probabilistic heuristic approach to optimizing problems. The probability of state transition p, from s to s', is given by the following equations.

$$p = P(\delta E, \tau) \tag{12}$$

$$\delta E = E(s') - E(s) \tag{13}$$

Where $E(s)$ represents the energy level of the state s, which corresponds to the fitness value of s. While the probability function P given in Equation 12 is usually exponential decreasing with time τ, it may be any function that is inversely proportional to τ. Given a decreasing transition probability function (which is comparable to mutation rate of genetic algorithm), transitions to states of lesser fitness becomes less likely and eventually impossible as $\tau \to \infty$.

Random transformations of the fitness landscape may transform the global maximum into a local maximum. If simulated annealing has a large τ value at the point of transformation, it will not be able to escape the newly created local maximum. Consequently, any changes in the search space or fitness

```
function GA()
{
    Population = initialize( 100 )
    for ( g = 1 to 10 ) { // run for 10 generations
        aux = {}
        for ( i = 1 to 50 ) {
            p1, p2 = tournament4( Population ) // parents
            c1, c2 = crossover( p1, p2 ) // children
            mp1 = mutate( p1 ) // mutated parents
            mp2 = mutate( p2 )
            aux = aux + { p1, p2, c1, c2, mp1, mp2 }
        }
        aux = sort( aux )
        Population = aux[0:50] + aux[150:200]  // (*)
    }
    return Population
}
```

Figure 4: Pseudo code of the genetic algorithm implementation

landscape as discussed in Section 5.4.1 requires simulated annealing to be restarted. Thus genetic algorithm is preferred over simulated annealing.

5.7.2 Random Mutation Hill Climber

Random mutation hill climber is similar to simulated annealing, but its mutation rate (corresponding to the state transition probability of simulated annealing) is maintained, not decreased, over time. With constant mutation rate, random mutation hill climber needs not to be restarted when the search space (and its corresponding fitness landscape) changes over time (as described in Section 5.4.1).

5.7.3 Parallel Random Mutation Hill Climber

Parallel random mutation hill climber is essentially multiple random mutation hill climbers running in parallel. Being a multi-point search technique, it possesses exploration power that is comparable to genetic algorithm. However, it lacks the notion of construction, that is combining good sub-solutions into good solutions, that genetic algorithm is capable of.

6 Experiments And Results

Past stock price data of Microsoft Corp. (MSFT) from 1/2/1990 to 12/31/1999 was used in the experiment. Arbitrarily selected price series were used as trend patterns for classification. The classification algorithm in Section 3.4.2 was then used to classify all available data and select those that are similar to the trend selected earlier. A fitness function $fitness2$, as introduced in Section 5.3, that based on the trend pattern series was defined to be used with the genetic algorithm.

The instances of *exponential moving average crossover* indicator produced by the genetic algorithm (See Section 5.6) were then evaluated against some randomly selected price series of similar trend using the fitness function, $fitness$ (See Section 5.2). The results are shown below with maximum profit and profit gained by the *Buy And Hold* strategy serving as comparisons.

6.0.4 Legend Of Experiment Results

- *Price pattern*: The actual sub-series of the Microsoft Corp. (MSFT) price series used as the trend patterns.
- *Test data*: $n \times m$ indicates n sets of m days price data. A total of nm trading days worth of price data.
- *Instance*: Representation the instance of *exponential moving average* indicator in the form of $(long\_period, short\_period)$.
- *Max*: The maximum profit possibly gained from the test data, as defined in Section 5.3.1.
- *BAH*: Profit gained from the test data by applying the *Buy And Hold* strategy.
- *GA*: Profit gained from the test data by applying the instance of *exponential moving average* produced by the genetic algorithm.

6.1 Experiment I

Experiment I verifies the feasibility of the framework proposed in this work for constructing good technical indicator instances. As a result of trend classification, the instances of *exponential moving average* indicator (shown

in the form of $(long\_period, short\_period)$ produced by the genetic algorithm are significantly better than trading by the *Buy And Hold* strategy. This result also serves as the justification for the introduction of trend classification for evolving technical indicator instances for targeted price trends.

Price pattern	*Test data*	*Instance*	*Max*	*BAH*	*GA*
9-27-1999 – 10-22-1999	33 × 20	(3,1)	16.985%	4.219%	10.312%
6-19-1997 – 7-17-1997	29 × 20	(8,4)	16.538%	3.904%	5.005%
2-5-1999 – 3-6-1999	35 × 20	(8,7)	14.735%	1.700%	6.895%
8-15-1991 – 9-12-1991	15 × 20	(9,8)	15.893%	5.419%	9.042%
3-25-1991 – 4-22-1991	19 × 20	(9,6)	15.167%	2.840%	4.813%

Table 2: Experiment I results

6.2 Experiment II

Experiment II investigates how well does the proposed framework scale as the length of the trend pattern increases. It is shown that the increment of trend pattern length does not have a heavy impact on the technical indicators. It is thus concluded that the framework scales pretty well (linearly decreasing, discussed below) for longer trend pattern to the point of 60 days in length.

Price pattern	*Test data*	*Instance*	*Max*	*GA*	*GA/Max*
9-27-1999 – 10-8-1999	33 × 10	(3,1)	7.543%	5.042%	0.668
9-27-1999 – 10-22-1999	33 × 20	(3,1)	16.985%	10.312%	0.607
9-27-1999 – 11-5-1999	33 × 30	(3,1)	27.218%	15.146%	0.556
9-27-1999 – 11-19-1999	33 × 40	(3,1)	38.249%	20.091%	0.525
9-27-1999 – 12-6-1999	28 × 50	(8,7)	52.881%	25.388%	0.480
9-27-1999 – 12-20-1999	24 × 60	(8,7)	68.843%	30.612%	0.445

Table 3: Experiment II results

The ratio GA/Max decreases linearly as the length of trend pattern increases, as shown in Figure below. This result agrees with Corollary 1 in Section 2.3 on the limited expressive power of technical indicators. As the length of the trend pattern increases, the trend pattern may become more complex and contains more distinctive characteristics, as well as more random noise, than the technical indicator can account for. Therefore, the ratio of profit gained over the maximum possible decreases.

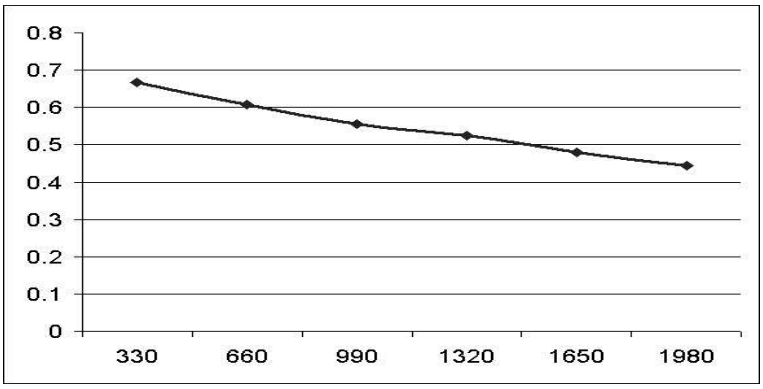

Figure 5: Graph of GA/Max over test data length

7 Summary

In this work, we have looked into the basic assumptions of the working of technical indicators for stock trading (Section 2). The suitability of genetic algorithms for indicator construction was shown by both theoretical arguments (Section 5.5) and empirical results (Section 6).

Good profit (compared against *Buy And Hold*) gained by the instances of technical indicator produced by genetic algorithm suggests that it is a good approximation to the ideal solutions for daily trading. It further suggests that trend classification (Section 3) ensures the evolved technical indicator instances work well on their respective targeted price trends.

While the framework proposed for technical indicator construction using genetic algorithm (Section 5) was introduced in the context of the *exponential moving average crossover* indicator, it may be conveniently employed for constructing other technical indicators by introducing appropriate chromosome representations for the intended indicators.

8 References

8.1 Conference papers

1. Kenneth A. DeJong. *Genetic Algorithms are NOT Function Optimizers*. The Second Workshop on Foundations of Genetic Algorithms (FOGA), 1992, Vail, Colorado. (pp. 5–18)

2. Stephaine Forrest, Melainie Mitchell. *Relative Building-Block Fitness and the Building Block Hypothesis*. The Second Workshop on Foundations of Genetic Algorithms (FOGA), 1992, Vail, Colorado. (pp. 109–126)

3. Stephen Chen, Stephen F. Smith. *Putting the "Genetics' Back into Genetic Algorthms (Reconsidering the Role of Crossover in Hybrid Operators)*. Proceedings of the 1998 Foundations of Genetic Algorithms (FOGA-5) workshop, 1998, Leiden, The Netherlands. (pp. 103–116)

4. William M. Spears. *Crossover or Mutation?*. The Second Workshop on Foundations of Genetic Algorithms (FOGA), 1992, Vail, Colorado. (pp. 221–238)

5. William M. Spears, Kenneth A. DeJong. *Dining with GAs: Operator Lunch Theorems*. Proceedings of the 1998 Foundations of Genetic Algorithms (FOGA-5) workshop, 1998, Leiden, The Netherlands. (pp. 85–102)

8.2 Books or reports

6. Jason Kingdon. *Intelligent systems and financial forecasting*. London ; New York : Springer, c1997.

7. Martin J. Pring. *Technical analysis explained : the successful investor's guide to spotting investment trends and turning*. New York : McGraw-Hill , c1991.

8. W.B. Langdon. *Genetic programming and data structures*. Boston : Kluwer Academic Publishers, c1998.

9. William B. Langdon, Riccardo Poli. *Foundations of genetic programming*. New York : Springer, 2002. Edition 3rd ed.

8.3 Unpublished reports and theses

10. Tan Han Wei. *Genetic Algorithms for Technical Indicator Discovery*. 2004.