Using Population-based Metaheuristics and Trend Representative Testing to Compose Strategies for Market Timing

Ismail Mohamed and Fernando E. B. Otero
University of Kent, Chatham Maritime, Kent, U.K.

Keywords: Particle Swarm Optimization, Genetic Algorithms, Market Timing, Technical Analysis.

Abstract: Market Timing is the capacity of deciding when to buy or sell a given asset on a financial market. Market Timing strategies are usually composed of components that process market context and return a recommendation whether to buy or sell. The main issues with composing market timing strategies are twofold: (i) selecting the signal generating components; and (ii) tuning their parameters. In previous work, researchers usually attempt to either tune the parameters of a set of components or select amongst a number of components with predetermined parameter values. In this paper, we approach market timing as one integrated problem and propose to solve it with two variants of Particle Swarm Optimization (PSO). We compare the performance of PSO against a Genetic Algorithm (GA), the most widely used metaheuristic in the domain of market timing. We also propose the use of trend representative testing to circumvent the issue of overfitting commonly associated with step-forward testing. Results show PSO to be competitive with GA, and that trend representative testing is an effective method of exposing strategies to various market conditions during training and testing.

1 INTRODUCTION

Trading in financial markets traces its history as far back as the early 13th century. From humble beginnings where traders met to exchange basic commodities, financial markets have since evolved where securities, stocks, bonds, commodities, currencies and other financial instruments are traded electronically within fractions of a second. A number of developments after the market crash of 1987 in the USA heralded the birth of electronic exchanges, and ushering with it a new form of trading: algorithmic trading (Patterson, 2013). One of the issues faced by designers of algorithmic trading systems is that of market timing. Market timing is defined as the identification of opportunities to buy or sell a given tradable item in the market so as to best serve the financial goals of the trader (Kaufman, 2013). A common approach designers use to build strategies for market timing is to use components that would take market data as input and generate recommendations to buy, sell or do nothing. A collection of these components would form the core of the strategy, and it would be the job of the designer to select which components to use and to tune their parameters.

Since the introduction of electronic exchanges, designers of algorithmic trading systems have increasingly employed computational intelligence techniques, one of which is Particle Swarm Optimization (PSO). Despite its popularity in other domains, PSO has seen limited use in the financial domain and in particular within the market timing space. This is compared with other metaheuristics such as genetic algorithms (GA) and genetic programming (GP). Within a few years of the introduction of electronic exchanges and trading in the mid 1990s, GA and GP have been used to guide trading decisions and form the core of market timing strategies. The earliest PSO approach to market timing, on the other hand, was introduced in 2011 (Briza and Naval Jr., 2011). Compared to GA, PSO has seen relatively limited adoption in the literature, despite PSO having a performance advantage over GA according to some studies (Hu et al., 2015; Soler-Dominguez et al., 2017).

Recently, PSO was used to build market timing strategies using six technical indicators and tested on four stocks in a step-forward fashion (Mohamed and Otero, 2018). In this paper, we conduct an extensive study of the performance of PSO versus a standard GA implementation. We selected a set of sixty three signal generating components to evaluate how PSO handles component selection and optimization compared to a GA, the current incumbent in the domain.

Mohamed, I. and Otero, F.
Using Population-based Metaheuristics and Trend Representative Testing to Compose Strategies for Market Timing.
DOI: 10.5220/0008066100590069
In *Proceedings of the 11th International Joint Conference on Computational Intelligence (IJCCI 2019)*, pages 59-69
ISBN: 978-989-758-384-1

We also introduce the concept of trend representative testing and apply it to overcome limitations with step-forward testing – the current standard testing method when it comes to market timing.

The remainder of the paper is structured as follows. In Section 2 we review the approach to formalize the problem of market timing in such a fashion so as to consider both the selection of signal generating components and the tuning of their parameters. In Section 3 we cover related work done in the application of metaheuristics to the problem of market timing and the associated limitations. In Section 4 we introduce the concept of trend representative testing and discuss how it can potentially overcome limitations with current testing standards. In Section 5 we discuss how PSO and GA were adapted to tackle market timing. We discuss our experimental setup, present the results obtained and provide a critique of the performance of the various metaheuristics involved in Section 6. Finally, in Section 7 we present a conclusion and suggestions for future research.

2 MARKET TIMING

As discussed previously, market timing is the capacity of deciding when to buy or sell a given security. Traders in financial markets have continuously looked for ways to best decide when to take action with a given stock, and here we discuss the various schools of thought that have emerged over time. The schools of thought can be roughly categorized into two major styles: technical analysis and fundamental analysis.

Technical analysis can be defined as the analysis of a security's historical price and volume movements, along with current buy and sell offers, for the purposes of forecasting its future price (Kaufman, 2013). The philosophy behind technical analysis is built upon three pillars: price discounts all the information we need to know about the traded security, prices move in trends and history has a likelihood of repeating itself. The first pillar assumes that all the forces that can affect the price of a security have been accounted for and already exerted their influence when the actual trade took place. This includes the psychological state of the market participants, the expectations of the various entities trading in that particular security, the forces of supply and demand, and the current state of the entity which the stock represents amongst other factors. It is therefore sufficient to only consider price movements and their history, as they are a reflection of all these forces and their influence. The second pillar assumes that prices move in trends based on the actions of traders currently dealing in that security and their expectations. The third pillar assumes that markets, presented with an almost similar set of stimuli and circumstances, have a tendency of reacting in the same fashion as it had in previous exposures. This was proven empirically in the histories of various securities across many markets over time as traders react in the same consistent fashion to shifts in price.

Techniques employing technical analysis will often take the form of functions known as indicators. Indicators will take in price history, along with a set of parameters that govern various aspects of an indicators behavior, and return a signal: an indication of whether it is favorable to buy or sell at the current moment. As an exhaustive list of all indicators currently available to the modern trader would easily fill multiple volumes, it is beyond the scope of this paper. Instead, the reader is directed to the works of Pring (Pring, 2002) and Kaufman (Kaufman, 2013) for a more detailed look at the world of technical analysis.

Fundamental analysis is the process of deriving the value of a security's value by analysis of the financial state of the company it represents (Penman, 2013). This will include analyzing current and previous financial documents and accounting records for the company, considering current management personnel and their performance in the past, sales performance history, earning history, current market sentiment towards the company and macroeconomic conditions, amongst many other factors. After considering these factors, analysis can arrive at a fair value for the security and a projection for it moving forward. Fundamental analysis is built on the core assumption that a discrepancy occurs between a security's fair price and market price as the market moves to close that gap. Recommendations for buying or selling the security are therefore based on identifying this discrepancy and how best to utilize it to achieve returns. As the rate of release of information could become an issue with some of the traditional information sources, fundamental analysis has grown to include techniques such as sentiment analysis over social media streams.

Instead of following a purest approach, a large number of traders would base their strategies on techniques from both schools. It is quite common for traders to use fundamental analysis for portfolio composition, then use technical analysis for market timing. It would actually be prudent to use methods from both schools to hedge the trader's risk in one or more of the components being mistaken or fed false data[1], and thus produce signals that might result in losses. In this paper, we have chosen to focus on using technical analysis indicators for the purposes of market timing.

As alluded to earlier, a market strategy could be built using a collection of components, where every component t consumes information regarding a security and returns a signal indicating whether to buy or sell. A component's signal is limited to three values: 1 for a buy recommendation, -1 for a sell recommendation and 0 for a hold recommendation. Every component will also have a weight and a set of parameters. The parameters will control the behavior of that component and are unique to every component type. The weight controls how much influence the component has on the overall signal produced by the candidate solution. The overall signal of the candidate solution is taken as the aggregation of weighted components, and interpreted as follows: buy when positive, sell when negative or hold otherwise. Formally, we can present this formulation as follows:

$$solution = \{w_1 t_1, ..., w_n t_n\}, \forall t_i : \{t_i^1, ..., t_i^x\} \quad (1)$$

$$signal = \sum_{i=1}^{n} w_i t_i \quad (2)$$

where x denotes the number of parameters for the component at hand, w represents the weight assigned to the component at hand, t represents a single component and n is total number of components within the solution. The weights for the components are all normalized to be between 0 and 1, and have a total sum of 1. As we are seeking to select the least possible subset that achieves our set objectives, varying combinations of components, along with varying values for component weights and parameters, will produce a rich landscape of candidates that return different signal values for the same market conditions.

3 RELATED WORK

In two recent and comprehensive studies, Hu et al. (Hu et al., 2015) and Soler-Dominguez et al. (Soler-Dominguez et al., 2017) investigate the use of computational intelligence techniques in finance. While the study by Soler-Dominguez et al. was more holistic in its coverage, the study done by Hu et al. was focused on the use of computational intelligence in the discovery of trading strategies. Both studies considered a large number of metaheuristics that belong under the computational intelligence umbrella, and that included evolutionary algorithms (GA, GP, differential evolution), swarm intelligence (PSO, ACO, artificial bee colony optimization), stochastic local search (simulated annealing, ILS, tabu search, GRASP), fuzzy systems and neural networks amongst others. Both studies cover a combined time span starting with the early 1990's and ending with recent times.

By surveying the techniques covered in both studies, we can see that genetic algorithms (GA), and to a slightly lesser extent genetic programming (GP), are the most applied metaheuristics when it comes to the issue of market timing. One of the earliest works using GA was by Allen and Karjalanien (Allen and Karjalainen, 1999). In it, Allen and Karjalanien use a GA to develop trading rules based on technical analysis indicators, and benchmarked their results against a buy-and-hold strategy and out of sample data. Another approach is to use GA to directly optimize the parameters of one or more financial analysis indicators, be they fundamental or technical in nature. Examples of such an approach can be seen in the work of de la Fuente et al. (de la Fuente et al., 2006) and Subramanian et al. (Subramanian et al., 2006) – the latter tackling market timing as a multi-objective optimization problem. Both of these approaches directly encode the indicator parameters into the GA chromosome, and use the metaheuristic to arrive at the values for these parameters that produce the best results. Other approaches since then use GA to improve the fitness of another primary metaheuristic in charge of producing the trading signals by optimizing its parameters. These primary, signal-producing metaheuristics included fuzzy systems, neural networks, self-organizing maps (SOM) and a variety of classification algorithms. A thorough breakdown of such synergistic approaches can be seen in (Hu et al., 2015). More recent approaches using GA to tackle market timing can be seen in the work of Kampouridis and Otero (Kampouridis and Otero, 2017) and Kim et al. (Kim et al., 2017).

Particle swarm optimization (PSO) has not seen the popularity of GA and GP in the space of market timing. Though introduced much later than GA, PSO has started seeing some adoption in the area. The earliest PSO approach to tackle market timing was proposed by Briza and Naval, Jr. (Briza and Naval Jr., 2011). Inspired by Subramanian and colleagues (Subramanian et al., 2006), the authors optimized the weights of instances of five technical indicators who had preset parameter values according to industry wide standards. All the weighed instances of the technical indicators would then produce a cumulative signal whether to buy or sell. The authors approached

[1] An example of this would be using a purely fundamental approach while trading Enron before its crash and bankruptcy in late 2001. A post-mortem investigation by the U.S. Securities and Exchange Commision (SEC) showed that the information published in the firm's financial documentation were false, leading to investments by market participants that were built on mislead assumptions.

the issue of market timing as a multi-objective optimization problem, and optimized for percentage of return and the Sharpe ratio. Chakravarty and Dash (Chakravarty and Dash, 2012) also used PSO for market timing by utilizing it to optimize a neural network capable of predicting movements in an index price. Similarly Liu et al. (Liu et al., 2012) used PSO to optimize a neural network that generated fuzzy rules for market timing and reported positive results.

More recently, Chen and Kao (Chen and Kao, 2013), used PSO to optimize a system that relied on fuzzy time series and support vector machines (SVM) to forecast the prices of an index for the purposes of market timing. Ladyzynski and Grzegorzewski (Ladyzynski and Grzegorzewski, 2013) used a combination of fuzzy logic and classification trees to identify price chart patterns, while PSO is used to optimize the parameters of the aforementioned hybrid approach. In their results, the authors have noted that use of PSO vastly improved the predictive capacity of the fuzzy logic and classification tree hybrid, and that the overall system proved to be promising. In the work by Wang et al. (Wang et al., 2014), a combination of a reward scheme and PSO was used to optimize the weights of two technical indicators. The Sharpe ratio was used to measure the performance of the hybrid, and in their results, the authors note that their system outperformed other methods such as GARCH. Bera et al (Bera et al., 2014) used PSO to only optimize the parameters of a single technical indicator. Although trading on the foreign exchange instead of the stock exchange, the authors note that their system has shown to be profitable in testing. Sun and Gao (Sun and Gao, 2015) used PSO to optimize the weights on a neural network that predicted the prices of securities on an exchange. The authors note that their system was able to predict the price with an error rate of around 30% when compared to the actual prices. Karathanasopoulos and colleagues (Karathanasopoulos et al., 2016) used PSO to optimize the weights on a radial basis function neural network (RBF-NN) that is capable of predicting the price of the crude oil commodity. Though not on the stock exchange, the trading of commodities occurs on similar exchanges and uses many of the same market timing techniques. Compared to two classical neural network models, the authors note that their PSO-augmented approach significantly outperformed them in predictive capacity.

In the nine discussed publications on the use of PSO in market timing, we can see a salient trend: PSO is used in a secondary role to optimize a primary metaheuristic or computational intelligence technique that is responsible for signal generation. There are only three exemptions: (Briza and Naval Jr., 2011), (Wang et al., 2014) and (Bera et al., 2014). In these three publications, the authors used PSO as the only metaheuristic, and within them PSO was either used to optimize the weights of a set of technical indicators or their parameters, but not both in unison. The only attempt we are aware of that considers both the selection of technical indicators and optimize their parameters was by Mohamed and Otero in (Mohamed and Otero, 2018). In it, the authors used PSO to both optimize the parameters of six technical indicators, as well as tune the weights of the generated signals and prune ineffective ones. They tested their work against four stocks and show that using PSO was a viable approach albeit with some caveats: only a limited number of indicators was used (6 in total); a small number of datasets (4 in total); and no baseline algorithm is used to assess the performance of PSO.

The work proposed in this paper addresses limitations identified in the literature so far: (1) it considers the optimization of both selection of technical indicators and their parameter values; and (2) avoid the tendency of strategies to overfit when using step-forward testing by proposing the use of trend representative testing.

4 TREND REPRESENTATIVE TESTING

The prominent method of testing a market timing strategy in the literature surveyed is a procedure known as step-forward testing (Kaufman, 2013; Hu et al., 2015; Soler-Dominguez et al., 2017). In step-forward testing, a stream of security data is divided chronologically into two sections: the earlier section for training and the later for testing. Although relatively easy to implement, this approach has a number of shortcomings. During training with this method, the training mechanism might only be exposed to the upwards, downwards or sideways trends available in the data. This raises the likelihood of the strategy produced to be overfit to one of those trends only and the chances of the strategy performing poorly in opposite trends. For example, if during training, all the data available represented uptrends, then a strategy trained with such data would produce poor results in a downtrend, and vice versa. Standard strategies taken during training, such as k-fold cross validation, are not easily applicable due to the structure of the data.

In order to overcome the limitations associated with step-forward testing, we propose the use of trend representative testing, as suggested by domain experts (Kaufman, 2013). The idea behind trend representative testing is to have a library of security data that

represents all three trends at various intensities and time lengths. Exposing strategies to these various trends during training improve their chances of performing better under numerous market conditions.

The process of building the trend representative dataset is as follows. First, raw security data is downloaded from a publicly available data provider. Second, this data is scanned for price shocks, and when one is detected, the raw data is divided into two cords – one before and one after the price shock respectively. The reason we remove price shocks is that they are outlier events, and including them in training data would imply that these highly irregular and very disruptive events can be readily predicted, which is not the case. Besides being catastrophic events, price shocks are rare and training a strategy to use them would be highly impractical. Price shocks are defined as price actions that are five times the Average True Range (ATR) within a short period of time (Kaufman, 2013). The cords are then subsampled with moving sliding windows of various sizes to produce strands that can be used for training and testing. The final step is to categorize these strands into upwards, downwards and sideways trends based on the Directional Index technical indicator (Pring, 2002) to ascertain intensity and direction.

In order to use this new dataset, we propose randomly picking a representative upwards, downwards and sideways strand from the pool of strands designated for testing. This triplet is then used by the algorithm within an iteration to measure the fitness of its candidate solutions. First a candidate solution is tested against each individual strand, and the results averaged to arrive at the final fitness for that candidate within that particular iteration. After all the candidates for the iteration have been assessed, a new triplet is selected for the next iteration and the process is repeated until the algorithm has completed its run. The idea behind trend representative testing is that we want discourage niching or specializing in one particular trend type and instead promote discovering market timing strategies that fair well against various market conditions.

5 METAHEURISTICS AND MARKET TIMING

In this section we will explore how to encode our market timing formalization, explaining how the metaheuristics were adapted to use this encoding and tackle the problem of market timing.

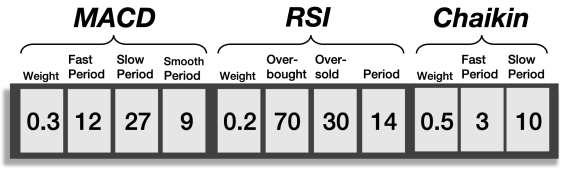

Figure 1: An example of an encoded candidate solution with three components.

5.1 Individual Representation and Fitness

Before discussing how the metaheuristics were adapted to tackle market timing with the aforementioned formalization, let us first consider how we can encode a candidate solution and assess the fitness of these individuals. The first step to work with the proposed formalization of market timing is to find an appropriate encoding for candidate solutions. A candidate solution would be a collection of signal generating components, each with a weight and set of parameters. For example, if we had three signal generating components such as the technical indicators Moving Average Converge Diverge (MACD), the Relative Strength Indicator (RSI) and the Chaikin Oscillator (Chaikin), we could possibly have a candidate solution of:

$$\begin{aligned} &0.3\times MACD(Fast\ Period = 12, Slow\ Period = 27,\\ &\qquad Smooth\ Period = 9)\\ &+0.2\times RSI(Overbought = 70, Oversold = 30,\\ &\qquad Period = 14)\\ &+0.5\times Chaikin(Fast\ Period = 3, Slow\ Period = 10) \end{aligned}$$

where the number preceding the indicator represents its weight and the values in the brackets represent the values of parameters per indicator. A visual example can be seen in Figure 1.

As the metaheuristics used are all based on a population of individuals, where each individual represents a candidate solution, we choose to encode a candidate solution as a multi-tier associative array. The top level binding associates an indicator identifier with a set of of its parameters. The bottom level binding associates a parameter identifier with its value. These parameters are dependent on indicator type, but all indicators have an instance of a weight parameter.

We choose to encode individuals in this manner as this allows us the flexibility of using as many components as we would like, without having to worry about the type and amount of parameters per component used or memorize a mapping of positions as would have been the case of using an array. The semantics of how to handle the components and tune the weights and parameters is then left to be implemented

in the metaheuristics, as will be discussed shortly. A new candidate solution is generated by instantiating components from the available catalog with random values for the parameters and adding it to the dictionary representing the individual.

As for assessing the fitness of an individual, we chose to maximize the Annualized Rate of Return (AROR) generated by backtesting the individual at hand. Backtesting would simulate trading based on the aggregate signal produced by a solution over a preset time period for a given asset. As for AROR, this can be defined as:

$$AROR_{simple} = \frac{E_n}{E_0} \times \frac{252}{n} \quad (3)$$

where E_n is final equity or capital, E_0 is initial equity or capital, 252 represents the number of trading days in a typical American calendar and n is the number of days in the testing period.

5.2 Genetic Algorithms

In order to apply genetic algorithms (GA) to tackle market timing using our proposed formalization, we started with a typical implementation of GA and modified its operators in order to accommodate how we encode candidate solutions. First, individuals selected for crossover are chosen using a typical tournament procedure, with the tournament size being a user-defined parameter. In preparation for crossover, the components in the genotype of individuals selected are ordered by key. A crossover point is then selected at random such that it lands between the definition of two components but not within them. Using the example mentioned earlier, we can generate a crossover point either between the definition of MACD and RSI, or between RSI and Chaikin. An example of a crossover operation can be seen in Figure 2. When a mutation event is triggered, a random component in an individual's genotype is replaced by a newly instantiated copy of the same component type. This newly instantiated copy would use random values for the constituent parameters, but still within valid range.

Crossover and mutation are used to generate a new population for the next generation, and this procedure continues until the allocated budget of generations are exhausted. An archive is used to keep track of the elite individuals per generation, with the most fit individual in the archive reported at the end of the GA run as the proposed solution.

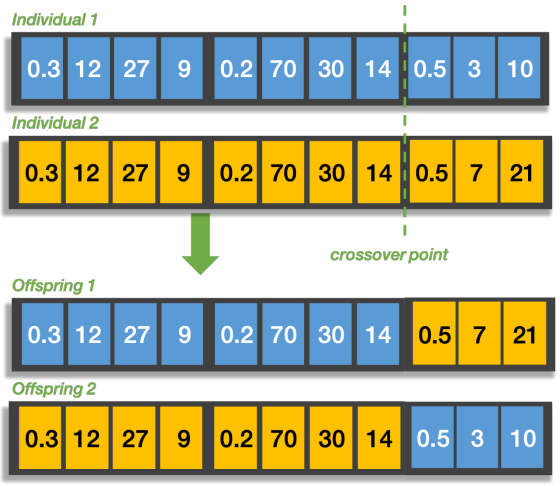

Figure 2: An example of a crossover operation.

5.3 Particle Swarm Optimization

We now turn our attention to adapting Particle Swarm Optimization (PSO) to tackle market timing. PSO will use the same encoding of the solution to represent a particle within its swarm, but the standard dynamics of PSO operators will have to be changed in order to adapt it to the problem at hand and use the solution encoding proposed.

The basic PSO model can be seen in Algorithm 1. This model supports both l-best and g-best neighborhood structures, based on the neighborhood size. When the neighborhood size is less than the total size of the population, we have an l-best neighborhood structure, otherwise we have a g-best neighborhood structure. A number of modifications are introduced in order to adapt PSO to tackle market timing. The first modification was that we pushed down the implementation of the addition, subtraction and multiplication operators required by the velocity update mechanism (lines 8–15) to be at the component level and not the metaheuristic level, in order to be agnostic to the types of signal generating components and their parameters. This would no longer limit the parameter types to either be in the binary or real-valued domains. A designer is now free to include a signal generating component of an arbitrary number of parameters and parameter types as long as the implementation of that component provides overrides to the necessary operators. Secondly, we also adopted a number of measures to promote convergence within the swarm and prevent velocity explosion (Clerc, 2002).[2] The

[2]Early experiments indicated the tendency of particles to adopt ever increasing values for velocity if left unchecked, leading to the particles quickly seeking the edges of the search space and moving beyond it.

Algorithm 1: Basic PSO high-level pseudocode.

```
initialize swarm S
repeat
    for every particle x_i in S do
        if f(x_i) > personal_best(x_i) then
            personal_best(x_i) ← f(x_i)
        end if
        for every component j in particle i do
            bias ← α v_ij(t)
            cognitive ← c1 r1 (y_ij(t) − x_ij(t))
            social ← c2 r2 (ŷ_ij(t) − x_ij(t))
            v_ij(t+1) ← bias + cognitive + social
            if j ∈ R then
                x_ij(t+1) ← x_ij + v_ij(t+1)
            else if j ∈ [0,1] then
                Pr(x_ij(t+1) → 1) : sigmoid(v_ij(t+1))
            end if
        end for
    end for
until stopping criteria met
return fittest particle
```

first of the measures is that we used a decreasing inertia schedule. This means that for every step of the algorithm, inertia for every particle decreases by an amount defined by a function based on the number of iterations left. The second measure we adopted was that we scaled down $v_{ij}(t+1)$ before updating a particle's state by a user-defined factor. As an alternative to velocity scaling, we also considered Clerc's Constriction as defined in (Clerc, 1999). The PSO would now be able to optimize both the parameters and weights of these components in relation to a financial fitness metric.

A common problem faced by search metaheuristics, PSO included, is getting stuck in local optima. Multiple measures have been devised since the introduction of the basic model to remedy that problem with varying degrees of success (Engelbrecht, 2005). Inspired by the work done by Abdelbar with Ant Colony Optimization (ACO) in (Abdelbar, 2008), we introduced a variation of PSO that stochastically updates the velocity of its particles only when it is favorable in terms of fitness. We will refer to this variation of PSO in the remainder of this paper as PSO^S. The modifications required for PSO^S are reviewed next.

Every particle x in the swarm S represents a candidate solution. From our earlier discussion, this means that a particle's state is a collection of weighted components, where each component has its own set of parameters. A particle starts out with an instance of all the available signal generating components, each instantiated with random weights and parameter values. In contrast with the basic PSO model, the cognitive and social components of the velocity update equation are modified to be calculated as:

$$cognitive = \begin{cases} y_i(t) - x_i(t) & \text{if rand()} < |\frac{f(y_i(t))}{f(x_i(t))+f(y_i(t))}| \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$social = \begin{cases} \hat{y}_i(t) - x_i(t) & \text{if rand()} < |\frac{f(\hat{y}_i(t))}{f(x_i(t))+f(\hat{y}_i(t))}| \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

- x: particle
- i: current particle index
- y: personal best
- $\hat{y}$: neighborhood best
- $f(x)$: the fitness of x
- $rand()$: random number between 0 and 1

According to equations 4 and 5, the cognitive and social components will only stochastically influence velocity update if there is an improvement in fitness, following a hill climbing fashion.

6 EXPERIMENTAL SETUP AND RESULTS

In order to evaluate the effectiveness of the proposed formulation in composing effective market timing strategies, we tested our PSO, PSO^S and GA. As all three algorithms have parameters, testing was preceded with hyper-parameter optimization performed using the iterated racing procedure (IRace) (López-Ibáñez et al., 2016). The IRace procedure was run with a budget of 300 iterations and a survivor limit of one, in order to arrive at a single configuration for each metaheuristic. The results of the IRace procedure can be seen in Table 1. In regards to PSO, IRace arrived at swarm sizes that are similar for both variants. The configuration discovered for PSO uses a much lower number of iterations when compared with the configuration for PSO^S. We can also see that the PSO configuration is slightly more reliant on the cognitive component with an l-best neighborhood spanning half the swarm, while the PSO^S configuration is slightly more reliant on the cognitive component with a g-best neighborhood structure. Both PSO configurations favored velocity scaling over Clerc's constriction, with PSO using relatively larger steps while PSO^S uses relatively smaller steps based on the scaling factors. After hyper-parameter optimization, we ended up with two PSO configurations: a fast acting l-best PSO and a slower g-best PSO with stochastic state update. As for GA, the findings of IRace show that the configuration had a relatively high mutation rate and a relatively low crossover rate when

Table 1: IRace discovered configurations for each of the algorithms tested.

PSO		**PSO^S**		**GA**	
Population	45	Population	59	Population	53
Iterations	28	Iterations	261	Generations	266
Neighbors	26	Neighbors	59	Mutation Probability	0.6306
c1	2.4291	c1	3.2761	Crossover Probability	0.455
c2	3.4185	c2	2.363	Tournament Size	22
Clamp	Scaling	Clamp	Scaling		
Scaling Factor	0.8974	Scaling Factor	0.551		

compared to typical values used for those parameters. This suggests that the solution landscape is on the rugged side, with sharp peaks and deep valleys that require radical moves to traverse. The discovered population size and the number of generations are similar in size to those of PSO^S.

All algorithms are trained and tested using trend representative testing. The data used contains 30 strands, representing 10 upwards, 10 downwards and 10 sideways trends at various intensities. The details of the trend dataset can be seen in Table 2. The columns in Table 2 describe the symbol of the source stock data, the beginning date, the ending date and the trend of every strand in the dataset. The data has then been split into 10 datasets, where each dataset would contain one of each trend for testing and the remaining 27 are then used for training. Each step in the training and testing procedure is repeated 10 times to cater for the effects of stochasticity.

In total, 63 signal generating components were used in both training and testing. These 63 components are of the technical variety, and contain a selection of momentum indicators, oscillators, accumulation/distribution indicators, candlestick continuation pattern detectors and candlestick reversal pattern indicators. Where the components took parameters that affected periods of data to look at, an upper limit of 45 days was set, so that we could get at least 5 trading signals within a single trading year (which is on average compromised of 252 days in the US market). Any other parameter was initialized to random values and the best performing setting discovered by the algorithms as they traverse the solution landscape.

Table 3 shows the minimum, median, mean and maximum fitnesses achieved by each algorithm, dataset and trend. GA showed a slight edge over the PSO variants with all three trends in dataset 1, while PSO^S showed a clear advantage with the downtrend in dataset 7. The basic PSO variant takes most of the wins based on means, when compared to GA and PSO^S. By looking at overall averages in Table 4, we can see that all three algorithms showed a higher overall fitness during a downtrend when compared with the other two trends, leaving us with an unbalanced performance. Nevertheless, performing better in downtrends is positive when compared with buy-and-hold strategies which would fail under such conditions. This issue of unbalanced performance with various trend types can perhaps be overcome if the problem of market timing is approached as a multi-objective one, where we try to discover a Pareto front with solutions that maximize fitness across all three trends. By having the performance of the three algorithms explicitly compared across a variety of trends, we have a better approximation of the performance of the strategies produced by these algorithms under live market conditions, and therein lies the advantage of using trend representative testing. With step-forward testing, we are limited to the price movements in the training section of the data. This can easily lead to strategies that are overfit to one particular type of trend, because that was all they were exposed to during training. With trend representative testing, we explicitly avoid this issue by exposing our algorithms to a variety of trends during both training and testing.

Table 2: Data strands used for training and testing.

Id	Symbol	Begin Date	End Date	Length	Trend
BSX1	BSX	2012-10-10	2013-07-09	185	↑
LUV1	LUV	2008-08-22	2010-05-07	430	↔
KFY1	KFY	2007-05-16	2007-10-12	105	↓
EXC1	EXC	2003-04-14	2003-08-20	90	↑
LUV2	LUV	2004-12-03	2005-05-04	105	↔
KFY2	KFY	2007-03-20	2007-09-21	130	↓
AVNW1	AVNW	2005-07-18	2006-01-12	125	↑
PUK1	PUK	2010-08-12	2012-04-03	415	↔
LUV3	LUV	2008-09-02	2009-01-30	105	↓
KFY3	KFY	2003-03-13	2003-08-04	100	↑
EXC2	EXC	2002-10-03	2003-08-04	210	↔
LUV4	LUV	2003-11-21	2004-04-01	90	↓
EXC3	EXC	2003-05-12	2003-10-15	110	↑
PUK2	PUK	2005-05-12	2006-03-13	210	↔
MGA1	MGA	1996-02-29	1996-07-08	90	↓
ED1	ED	1997-07-02	1997-11-20	100	↑
EXC4	EXC	1999-08-20	2000-03-30	155	↔
PUK3	PUK	2002-03-19	2002-07-25	90	↓
BSX2	BSX	2009-04-22	2009-09-18	105	↑
ED2	ED	2011-12-15	2012-05-16	105	↔
JBLU1	JBLU	2003-05-15	2003-11-10	125	↓
MGA2	MGA	2012-12-28	2013-10-14	200	↑
MGA3	MGA	1995-09-19	1996-12-13	315	↑
ATRO1	ATRO	1997-06-04	1997-11-28	125	↓
AVNW2	AVNW	2003-03-07	2003-08-05	105	↑
EXC5	EXC	2015-03-12	2016-09-02	375	↔
AVNW3	AVNW	2013-06-11	2013-11-20	115	↓
IAG1	IAG	2015-11-09	2016-08-24	200	↑
MGA4	MGA	1995-10-17	1996-04-22	130	↔
IAG2	IAG	2012-01-19	2012-06-04	95	↓

Table 5 shows the rankings of the algorithms after performing the non-parametric Friedman test with the Holm's post-hoc test by trend type on the mean results (García et al., 2010). The first column shows the trend type; the second column shows the algorithm name; the third column shows the average rank,

Table 3: Computational results for each algorithm over the 10 datasets. The min, median, mean and max values are determined by running each algorithm 10 times on each dataset. The best result for each dataset and trend combination is shown in bold.

			PSO^S				GA				PSO			
Dataset	**Trend**	**Test_Strand**	**Min**	**Median**	**Mean**	**Max**	**Min**	**Median**	**Mean**	**Max**	**Min**	**Median**	**Mean**	**Max**
0	↑	IAG1	-9.71	-5.79	-6.19	-3.42	-10.35	-5.99	-6.25	-1.39	-4.41	-4.26	**-4.01**	-3.04
	↔	MGA4	0.93	1.19	1.25	1.70	0.39	0.88	0.91	1.60	0.66	1.78	**1.60**	2.09
	↓	IAG2	0.69	0.93	1.22	2.17	0.80	1.95	1.71	2.16	2.17	2.17	**2.17**	2.17
1	↑	BSX1	-0.85	-0.34	-0.39	-0.11	-0.67	-0.35	**-0.36**	-0.13	-5.91	-0.31	-0.85	-0.02
	↔	LUV1	-1.10	-0.34	-0.42	-0.08	-1.33	-0.10	**-0.28**	-0.01	-1.71	-0.11	-0.45	-0.04
	↓	KFY1	2.00	2.07	2.07	2.17	2.01	2.08	**2.22**	2.67	1.86	2.09	2.14	2.66
2	↑	EXC1	2.80	2.83	**2.85**	2.92	2.80	2.82	2.83	2.90	2.80	2.80	2.80	2.80
	↔	LUV2	2.17	2.31	2.30	2.46	2.09	2.27	2.31	2.64	2.21	2.43	**2.44**	2.62
	↓	KFY2	1.65	2.00	2.04	2.85	1.61	1.75	1.77	2.05	1.56	1.98	**2.15**	3.61
3	↑	AVNW1	-3.58	-1.84	-1.21	1.22	-3.83	-2.01	-1.59	1.29	-1.97	0.70	**0.10**	1.26
	↔	PUK1	-2.45	-1.25	-1.00	0.38	-2.84	-1.44	-1.32	0.00	-2.37	-0.05	**-0.39**	0.00
	↓	LUV3	1.58	2.67	3.02	6.12	-0.85	2.91	2.03	4.63	1.08	3.06	**3.17**	4.70
4	↑	KFY3	1.39	2.42	2.40	2.94	2.08	2.49	2.45	2.67	2.67	2.72	**2.74**	2.80
	↔	EXC2	0.08	1.13	0.99	1.62	0.16	0.91	0.84	1.55	-0.53	1.31	**1.03**	1.60
	↓	LUV4	2.77	2.85	**2.85**	2.95	2.80	2.83	2.84	2.96	2.80	2.80	2.80	2.80
5	↑	EXC3	1.52	2.02	2.00	2.39	1.39	1.64	1.71	2.14	1.96	2.03	**2.12**	2.38
	↔	PUK2	-1.51	-0.23	-0.28	0.76	-4.10	-0.10	-0.88	0.51	0.06	0.59	**0.58**	1.07
	↓	MGA1	2.80	3.14	**3.15**	3.80	2.80	2.99	2.98	3.15	2.80	2.80	2.80	2.80
6	↑	ED1	-0.03	1.26	1.21	2.32	0.38	1.14	1.10	1.98	1.75	1.75	**1.88**	2.44
	↔	EXC4	1.20	1.97	**2.35**	3.96	1.53	2.27	2.33	3.72	0.95	2.07	2.33	3.47
	↓	PUK3	2.25	2.43	2.60	3.66	2.38	2.45	2.57	3.66	2.80	2.80	**2.80**	2.80
7	↑	BSX2	3.25	3.42	3.48	3.76	3.13	3.42	**3.51**	4.03	2.28	3.22	3.13	3.32
	↔	ED2	2.35	2.52	2.51	2.67	2.46	2.57	**2.58**	2.70	2.36	2.44	2.45	2.63
	↓	JBLU1	6.59	10.68	**10.28**	13.09	-0.07	9.41	7.99	12.06	0.62	8.08	7.56	11.95
8	↑	MGA2	-4.87	-4.63	-3.29	0.00	-6.38	-4.34	-4.50	-3.39	-4.69	-4.33	**-3.28**	-0.04
	↔	MGA3	-1.96	-0.10	-0.08	1.34	-1.09	-0.08	-0.13	0.51	-0.17	0.27	**0.23**	0.48
	↓	ATRO1	4.76	9.51	**9.17**	11.51	-8.86	9.75	8.49	16.08	5.29	9.10	8.68	11.31
9	↑	AVNW2	3.94	4.77	**4.63**	5.67	2.09	4.14	4.20	5.65	2.68	3.67	3.48	3.99
	↔	EXC5	-0.59	-0.35	-0.19	0.56	-1.18	-0.40	-0.27	0.96	-0.61	0.44	**0.27**	0.87
	↓	AVNW3	1.86	2.01	**2.02**	2.20	1.75	1.96	1.97	2.14	1.75	1.92	1.94	2.10

Table 4: Overall average fitness by trend for each algorithm.

	Algorithm		
Trend	PSO^S	GA	PSO
Downtrend	3.84	3.46	3.62
Sideways	0.74	0.61	1.01
Uptrend	0.55	0.31	0.81

Table 5: Average rankings of each algorithm according to the Friedman non-parametric test with the Holm post-hoc test over the mean performance. No statistical differences at the significance level 5% were observed.

Trend	Algorithm	Ranking	p-value	Holm
Downtrend	PSO^S (control)	1.7	–	–
	PSO	2.0	0.6708	0.05
	GA	2.3	0.1797	0.025
Sideways	PSO (control)	1.7	–	–
	GA	2.0	0.5023	0.05
	PSO^S	2.3	0.1797	0.025
Uptrend	PSO^S (control)	1.9	–	–
	GA	1.9	0.9999	0.05
	PSO	2.2	0.5023	0.025

where the lower the rank the better the algorithm's performance; the fourth column shows the p-value of the statistical test when the average rank is compared to the average rank of the algorithm with the best rank (control algorithm); the fifth shows the Holm's critical value. Statistically significant differences at the 5% level between the average ranks of an algorithm and the control algorithm are determined by the fact that the p-value is lower than the critical value, indicating that the control algorithm is significantly better than the algorithm in that row. The non-parametric Friedman test was chosen as it does not make assumptions that the data is normally distributed, a requirement for equivalent parametric tests. We can see from this table that PSO and PSO^S were ranked higher than GA in both downtrends and sideways, with a close tie for uptrends, although not at a statistically significant level. This suggests that PSO, both in its basic and modified flavors, is competitive with GA when it comes to the domain of market timing. PSO, in particular, has an advantage over GA in that it achieves these highly competitive results with an order of magnitude fewer number of iterations using a similar population size. The stochastic state update modification has given PSO^S a small improvement in ranking when tested in downtrends and uptrends over the PSO, although this is achieved with a greater number of iterations.

7 CONCLUSION

In this paper, we reviewed a formulation for the market timing problem and introduced three new contributions: (i) using trend representative testing to expose potential solutions to various market conditions while training and testing; (ii) designed GA and PSO algorithms to tackle market timing and finally (iii) compared our proposed GA and PSO algorithms using 30 strands (stocks undergoing a particular trend) and 63 signal generating components. Our results showed that the PSO variants are competitive to GA – which is the most widely used metaheuristic in market timing – and ranked better when it came to performance, with one variant doing so at a fraction of the number of iterations used by GA.

We suggest the following avenues of future research. First, use a more sophisticated measure of financial fitness. This would allow us to simulate hidden costs of trading such as slippage. Second, approach the problem of market timing as a multi-objective one by trying to maximize performance across the three types of trends and against multiple financial objectives. Finally, adapt more metaheuristics to tackle market timing and compare its performance against the currently proposed ones in significantly larger datasets. We could then use meta-learning to understand if and when metaheuristics perform significantly better than others under particular conditions and use that information to build hybrid approaches that use more than one metaheuristic to build strategies for market timing.

REFERENCES

Abdelbar, A. (2008). Stubborn ants. In *IEEE Swarm Intelligence Symposium, SIS 2008*, pages 1 – 5.

Allen, F. and Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 51(2):245–271.

Bera, A., Sychel, D., and Sacharski, B. (2014). Improved Particle Swarm Optimization method for investment strategies parameters computing. *Journal of Theoretical and Applied Computer Science*, 8(4):45–55.

Briza, A. C. and Naval Jr., P. C. (2011). Stock trading system based on the multi-objective particle swarm optimization of technical indicators on end-of-day market data. *Applied Soft Computing*, 11(1):1191–1201.

Chakravarty, S. and Dash, P. K. (2012). A PSO based integrated functional link net and interval type-2 fuzzy logic system for predicting stock market indices. *Applied Soft Computing*, 12(2):931–941.

Chen, S.-M. and Kao, P.-Y. (2013). TAIEX forecasting based on fuzzy time series, particle swarm optimization techniques and support vector machines. *Information Sciences*, 247:62–71.

Clerc, M. (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1951–1957.

Clerc, M. (2002). Think locally act locally-a framework for adaptive particle swarm optimizers. *IEEE Journal of Evolutionary Computation*, 29:1951–1957.

de la Fuente, D., Garrido, A., Laviada, J., and Gómez, A. (2006). Genetic algorithms to optimise the time to make stock market investment. In *Genetic and Evolutionary Computation Conference*, pages 1857–1858.

Engelbrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons Ltd.

García, S., Fernández, A., Luengo, J., and Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sci.*, 180(10):2044–2064.

Hu, Y., Liu, K., Zhang, X., Su, L., Ngai, E. W. T., and Liu, M. (2015). Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, 36:534–551.

Kampouridis, M. and Otero, F. E. (2017). Evolving trading strategies using directional changes. *Expert Systems with Applications*, 73:145–160.

Karathanasopoulos, A., Dunis, C., and Khalil, S. (2016). Modelling, forecasting and trading with a new sliding window approach: the crack spread example. *Quantitative Finance*, 7688(September):1–12.

Kaufman, P. J. (2013). *Trading Systems and Methods*. John Wiley & Sons, Inc, 5th edition.

Kim, Y., Ahn, W., Oh, K. J., and Enke, D. (2017). An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms. *Applied Soft Computing*, 55:127–140.

Ladyzynski, P. and Grzegorzewski, P. (2013). Particle swarm intelligence tunning of fuzzy geometric protoforms for price patterns recognition and stock trading. *Expert Systems with Applications*, 40(7):2391–2397.

Liu, C. F., Yeh, C. Y., and Lee, S. J. (2012). Application of type-2 neuro-fuzzy modeling in stock price prediction. *Applied Soft Computing*, 12(4):1348–1358.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Mohamed, I. and Otero, F. E. (2018). Using Particle Swarms to Build Strategies for Market Timing: A Comparative Study. In *Swarm Intelligence: 11th International Conference, ANTS 2018, Rome, Italy, October 29–31, 2018, Proceedings*, pages 435–436. Springer International Publishing.

Patterson, S. (2013). *Dark Pools: The Rise of A.I. Trading Machines and the Looming Threat to Wall Street*. Random House Business Books.

Penman, S. H. (2013). *Financial Statement Analysis and Security Valuation*. McGraw-Hill.

Pring, M. (2002). *Technical Analysis Explained*. McGraw-Hill.

Soler-Dominguez, A., Juan, A. A., and Kizys, R. (2017). A Survey on Financial Applications of Metaheuristics. *ACM Computing Surveys*, 50(1):1–23.

Subramanian, H., Ramamoorthy, S., Stone, P., and Kuipers, B. (2006). Designing Safe, Profitable Automated Stock Trading Agents Using Evolutionary Algorithms. In *Genetic and Evolutionary Computation Conference*, volume 2, page 1777.

Sun, Y. and Gao, Y. (2015). An Improved Hybrid Algorithm Based on PSO and BP for Stock Price Forecasting. *The Open Cybernetics & Systemics Journal*.

Wang, F., Yu, P. L., and Cheung, D. W. (2014). Combining technical trading rules using particle swarm optimization. *Expert Systems with Applications*, 41(6):3016–3026.