Combining technical trading rules using particle swarm optimization

Fei Wang\textsuperscript{a,}\textsuperscript{,} Carolina Ribeiro, Philip L.H. Yu\textsuperscript{b}, David W. Cheung\textsuperscript{a}

\textsuperscript{a}Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong
\textsuperscript{b}Department of Statistics and Actuarial Science, The University of Hong Kong, Pokfulam Road, Hong Kong

\textbf{A R T I C L E   I N F O}

\textbf{Keywords:}
Technical trading rules
Particle swarm optimization
Bootstrapping

\textbf{A B S T R A C T}

Technical trading rules have been utilized in the stock market to make profit for more than a century. However, only using a single trading rule may not be sufficient to predict the stock price trend accurately. Although some complex trading strategies combining various classes of trading rules have been proposed in the literature, they often pick only one rule for each class, which may lose valuable information from other rules in the same class. In this paper, a complex stock trading strategy, namely performance-based reward strategy (PRS), is proposed. PRS combines the two most popular classes of technical trading rules - moving average (MA) and trading range break-out (TRB). For both MA and TRB, PRS includes various combinations of the rule parameters to produce a universe of 140 component trading rules in all. Each component rule is assigned a starting weight, and a reward/penalty mechanism based on rules’ recent profit is proposed to update their weights over time. To determine the best parameter values of PRS, we employ an improved time variant particle swarm optimization (TVPSO) algorithm with the objective of maximizing the annual net profit generated by PRS. The experiments show that PRS outperforms all of the component rules in the testing period. To assess the significance of our trading results, we apply bootstrapping methodology to test three popular null models of stock return: the random walk, the AR(1), and the GARCH(1,1). The results show that PRS is not consistent with these null models and has good predictive ability.

\textcopyright 2013 Elsevier Ltd. All rights reserved.

\section{1. Introduction}

Technical trading rules are widely used in the financial markets as technical analysis tools for security trading. Typically, they predict the future price trend by analyzing historical price movements and initiate buy/sell signals accordingly. Technical trading rules have been developed for more than a century and many empirical studies including, but not limited to, Brock, Lakonishok, and LeBaron (1992), Cencay (1998), Kestner (2003), Austin, Bates, Dempster, Leemans, and Williams (2004), Hsu and Kuan (2005), Lento and Gradojevic (2007), Metghalchi, Marcucci, and Chang (2012), and Chiang, Ke, Liao, and Wang (2012), provided supporting evidence to the significant profitability of various technical trading rules. Until nowadays trading rules are commonly used by practitioners to make trading decisions in many financial markets (Menkhoff, 2010).

Instead of asking whether specific rules work, Allen and Karjalainen (1999) proposed using genetic algorithms (GA) (Holland, 1992), a class of machine learning algorithms, to discover profitable technical trading rules. The targeted rules were logical combinations of many fundamental technical indicators using arithmetic operators and logical functions. Similarly, Dempster and Jones (2001) used genetic programming (GP) (Koza, 1994) which is an extension of GA to develop a trading system consisting of different technical indicators for a foreign exchange market. In their studies, a set of training data was used to find the optimal trading rules, then these rules were tested on an out-of-sample data. However, the discovered rules did not show consistent and robust profitability for the testing data even though they had significant performance for the training data. One reason may be that these studies ignored the existing profitable trading rules in the literature and the discovered rules were totally data driven.

In practice, investors may not stick to only a single rule without considering the available information generated from other technical trading rules. Pring (1991) also argued that no single trading rule can ever be expected to forecast all price trends and it is important to combine simple trading rules together to get a complex trading strategy. Hsu and Kuan (2005) first examined the profitability of three classes of complex trading strategies: learning strategies (LS), vote strategies (VS) and fractional position strategies (FPS). Their results, however, showed that these complex trading strategies did not provide significant improvement as compared with simple trading rules. The failure of these trading strategies may be because they are relatively primitive. For example, LS picked the best simple trading rule for trading decision making...
each time instead of combining all rules in an appropriate manner. For VS and FPS, both of them regarded all simple trading rules as equally important without considering their relative performances.

Unlike the above primitive combination methods, Subramanian, Ramamoothy, Stone, and Kuipers (2006) proposed a weighted combination of technical trading rules. In their study, each rule is assigned a weight, and the strategy’s signal is determined by the weighted sum of all component rules’ signals. They created this combination by applying a GA to optimize the best set of weight vector. Thereafter Briza and Naval (2011) proposed a similar stock trading strategy whose weight vector was optimized by particle swarm optimization (PSO) (Kennedy & Eberhart (1995)). Both strategies were found to outperform the best component trading rules in terms of profit in the testing period. However, they only considered a commonly used rule for each class of trading rule in their studies. This may not guarantee that the trading rules under consideration always perform better than those not considered. Results from Brock et al. (1992), Sullivan, Timmermann, and White (1999) and Hsu and Kuan (2005) also support that the profitability of various rules belonging to the same type varies significantly. Therefore it is important to include various combinations of parameters for each class of rule as many as possible to get a comprehensive coverage of simple trading rules. Note that above approaches (Subramanian et al., 2006; Briza & Naval, 2011) assumed the weights of component rules were held fixed during the entire trading period. However, component rules’ performances may not be stable and hence a trading strategy with a static choice of component weights may be hard to perform well consistently over time. In this regard, an objective of this work is to consider a dynamic updating scheme for component weights.

As discussed above, the optimization of complex trading strategies is to find the optimal combination of simple trading rules, or in other words the optimal set of parameter values with the goal of making profit as high as possible. As opposed to traditional function optimization problems, the evaluation functions of complex trading strategies are non-differentiable. Therefore, the classic mathematical optimization methods such as linear programming and Newton’s method are not practical. In the literature, GA and PSO are the two most popular stochastic optimization algorithms used for financial forecasting purposes. For example, Allen and Karjalainen (1999) used GA to learn technical trading rules for the S&P 500 index; Esfahani and Mousavi (2011) generated technical trading rules for decision making in stock markets by using GP; Hsu et al. (2011) presented a new funds trading strategy which combines turbulent particle swarm optimization (TPSO) and mixed moving average techniques, and recently Serrminis, Theofilatos, Karathanasopoulos, Georgopoulos, and Dunis (2013) introduced a hybrid of PSO and adaptive radial basis function for foreign exchange forecasting. More applications of GA and PSO in financial market prediction can be found from Dempster and Jones (2001), Briza and Naval (2011), Wong, Leung, and Guo (2012), and Kuo and Hong (2013). Compared to GA, PSO is not only easy to be implemented, but also could achieve the same performance as GA with higher computing efficiency (Lee, Lee, Chang, & Ahn, 2005; Babaoglu, Findik, & Ulker, 2010). Therefore PSO seems to be a better choice for trading rules optimization and is adopted in this work.

In this paper, we present a complex stock trading strategy called performance-based reward strategy (PRS). PRS combines two types of the most popular trading rules: moving average (MA) and trading range break-out (TRB). In all 140 trading rules are created as component rules of PRS by taking different parameter values of MA and TRB rules. All parameter values are well chosen to represent a wide coverage of the parameters for each rule class (Brock et al., 1992; Sullivan et al., 1999). Each component rule is assigned a starting weight which indicates its significance in trading decision. A reward/penalty mechanism based on component rules’ performance is proposed to update all component rules’ weights over time. The trading signal of PRS is determined by the weighted sum of component rules’ signals and two additional signal threshold parameters. Together with component rules’ starting weights and other five parameters of PRS (to be discussed later), there are altogether 145 parameters for PRS. We use an improved time variant particle swarm optimization (Ratnaweera, Halgamuge, & Watson, 2004) to optimize the best set of the 145 parameters.

To assess PRS performance in the stock market, we apply bootstrapping methodology to determine whether PRS makes profit by finding some useful information hiding in the stock market or by good fortune. Three popular null models – random walk, AR(1) and GARCH(1,1) – are used to generate a great number of bootstrapping samples. Then we compare the excess return of PRS on original stock data and the bootstrapping samples to find the evidence of strong prediction ability of PRS.

The rest of this paper is organized as follows: Section 2 gives details of the proposed complex trading strategy PRS. Section 3 briefly introduces PSO and describes how PRS is optimized with PSO. Empirical results are presented and discussed in Section 4. Section 5 accesses the significance of the trading results by bootstrapping methodology. Conclusions and future works are drawn in Section 6, and Appendix A gives details of all trading rules used in this study.

2. Performance-based reward strategy

2.1. Component trading rules

Technical trading rules have been developed for more than a century and are widely used in financial market as a technical analysis tool for stock trading. Various kinds of trading rules were proposed in past decades. As Pring (1991) argued that no single trading rule can ever be expected to predict all price trend, it is important to combine these simple rules together to get a complex trading strategy. In fact, almost all traders, investment firms and fund managers make trading decisions with the help of a trading strategy consisting of a set of technical indicators instead of a single trading rule (Pardo, 2008). In this study, we consider two types of the simplest and most popular technical trading rules in the literature – moving average (MA) and trading range break-out (TRB).

Essentially, a moving average is the mean of stock prices over a moving window of n days as follows:

\[ p = \frac{1}{n} \sum_{i=t-n+1}^{t} p_i \]  

where \( t \) is the current trading day and \( p_i \) is the close stock price on day \( i \). It is recalculated and updated each trading day. In MA rules, there are two averages (long-period and short-period averages) over two moving windows of \( m \) days and \( n \) days, respectively, where \( m > n \). Consider a trading day \( t \), a MA rule initiates buy (sell) signal if the short-period moving average is above (below) the long-period moving average. It is the simplest form of this rule (Brock et al., 1992) and is used in this paper.

The second technical trading rule is trading range break-out (TRB). It calculates the highest close price \( H \) and the lowest close price \( L \) over a fixed \( n \) days interval as follows:

\[ H = \max(p_{t-1}, p_{t-2}, \ldots, p_{t-n}) \]
\[ L = \min(p_{t-1}, p_{t-2}, \ldots, p_{t-n}) \]  

The highest and lowest price form a running channel (trading range) for each day’s stock price and the trading signals are invoked by the stock price’s breakout from the channel. Suppose the close
price of trading day \( t \) is \( p_t \), a buy signal is generated when \( p_t > H \) and a sell signal is generated when \( p_t < L \).

There are two parameters (length of two moving windows) for MA rules and only one parameter (length of channel) for TRB rules. Some parameter values are suggested as standard and are widely used in practice, e.g. MA of 10 and 40 days (Kestner, 2003) and TRB of 50 days (Brock et al., 1992). Given such a complex and dynamic market, however, it is not a good choice to consider only one rule for a specific class of trading rule because there is no guarantee that the selected rule is always better than those not considered. Therefore, we specify a set of parameter values for each kind of trading rule and combine them as component trading rules of PRS. Totally 119 MA rules and 21 TRB rules are generated in this study. More details of these component rules’ parameter values are listed in Appendix A.

2.2. Signal generation of PRS

In PRS, each component rule \( r_i \) is assigned a starting weight \( w_i \), which measures the influence of \( r_i \) to the signal generation of PRS. Consider a trading day \( t \), each component rule initiates a signal \( s_{i,t} \). This signal \( s_{i,t} \) takes value 1 and -1 if the signal is ‘buy’ and ‘sell’, respectively. If no signal is generated, \( s_{i,t} \) is set to zero. The signal of PRS on trading day \( t \) is given by:

\[
s_{t} = \sum w_{i} s_{i,t}
\]

where the sum of the all weights \( \sum w_{i} \) should be 1 so that \( s_{t} \) is between -1 and 1.

Note that \( s_{t} \) summarizes all component rules’ prediction on stock trend. If \( s_{t} \) is close to 1, this means most of the influential component rules suggest buy signals. On the contrary, \( s_{t} \) nearing -1 means more influential rules suggest sell signals. So we propose that PRS initiates a buy (sell) signal if \( s_{t} \) is greater (smaller) than a positive buy (negative sell) threshold, \( \theta_{1} \) (\( \theta_{2} \)); otherwise PRS does not initiate any signal and investors do nothing on that day. Higher threshold indicates that the strategy is more strict in buy or sell and lower threshold represents a more tolerant strategy. The signal generation of PRS is shown in Fig. 1.

2.3. Reward and penalty of component rules

Each component rule \( r_i \) of PRS is associated with a weight \( w_i \). It represents the influence of \( r_i \) to the trading decision making. However, these rules’ performance may change during trading, especially over a long time period. It is reasonable to reward a good rule by adding more weight to it and to penalize a bad rule by deducting some weight from it. The good or bad is measured by the profitability of rules in recent time, and the updating of weights should be conducted at regular intervals. As a result, two time spans – memory span \( \theta_{3} \) and review span \( \theta_{4} \), which are introduced in the learning strategy (Hsu & Kuan, 2005), are used here. Memory span is a historical period used for evaluating the rule performance. Review span is the time interval over which the weights of component rules should be updated. We set \( \theta_{3} \geq \theta_{4} \) as suggested by Hsu and Kuan (2005).

Suppose on trading day \( t \), we evaluate all component rule’s performance and update their weights accordingly. Let \( P_t \) denotes the profit of rule \( r_i \) from day \( t - \theta_3 \) to day \( t - 1 \). For those non-profitable rules, we deduct their weights by a constant:

\[
w_i = w_i - \frac{\theta_3}{N} \quad \text{if} \quad P_t < 0
\]

where \( N \) is the total number of component rules and \( \theta_3 \) is a parameter called reward factor controlling the degree for penalty and reward. It is noted that \( w_i \) should not be negative, so \( w_i \) is set to zero if it is below the weight threshold \( \frac{\theta_3}{N} \).

All weights deducted from the non-profitable rules are summed to form a temporary weight \( W \). Then we increase the weight of those profitable rules using following equation:

\[
w_i = w_i + \frac{W}{N} \quad \text{if} \quad P_t > 0
\]

where \( N_+ \) is the number of profitable weights found in the memory span.

Note that the sum of all weights would not be changed by the weight penalty and reward, and hence the mean weight of component rules is a constant which is equal to 1/140 (0.0071). However, if most of the rules are non-profitable and only a few rules are profitable, above reward/penalty approach may add too much weight to those few profitable rules. Imagine that there are 100 rules in which only one rule is profitable in memory span, the weight increment of the only profitable rule is 99 times of the weight decrement of any other rule. The reward may be too much, especially when the rule is just profitable in a short period of time. To avoid a huge reward, we replace Eq. (4) with:

\[
w_i = w_i - \frac{\theta_3}{N} \frac{N_+}{N} \quad \text{if} \quad P_t < 0
\]

where term \( \frac{N_+}{N} \) guarantees that the penalty weight of any non-profitable rule and the reward weight of any profitable rule is capped at \( \frac{\theta_3}{N} \). It is noteworthy that when all rules are profitable or all of them are non-profitable, our reward/penalty mechanism would not be triggered as in this case there is no need to reward or penalize any rule.

In the above reward/penalty mechanism, the reward factor \( \theta_3 \) controls the degree for penalty and reward. If \( \theta_3 \) is equal to zero, all weights become constant throughout the whole trading period and PRS degenerates into a weight strategy (WS). The performance of WS is tested as well and compared with PRS in Section 4 to see the advantages of using the proposed reward/penalty mechanism.

2.4. Trading with trading strategies and performance evaluation

In this section, we discuss how to use technical trading strategies such as PRS, TRB and MA rules to trade in a stock market. Consider a stock with close prices over a time period, we start trading with a trading strategy based on an initial equity \( E_0 \). To make the trading result more convincing (Hsu & Kuan, 2005), transaction cost is considered in our study. In addition, short selling is not allowed so that the equity is always positive.

Suppose the first ‘buy’ signal is generated on day \( b \) and the close price on that day is \( p_b \), \( k \) shares could be bought using equity \( E_0 \) (equal to \( E_1 \) for the first trade) with transaction cost \( c \):
The shares held are sold till a ‘sell’ signal is detected on later day $s$. After paying for the transaction cost, the trader collects equity:

$$E_s = p_s k (1 - c) = E_k - \frac{p_k (1 - c)}{p_k (1 + c)}$$

where $p_k$ is the close price on day $s$. The day $b$ ($s$) is then the starting (ending) day of this trade.

As trading continues, the equity $E_t$ is used to buy shares when a new ‘buy’ signal is detected. On the last day of trading, we sell all shares in hand (if still in market) to obtain the final equity $E_f$ of this stock. Suppose there are $m$ trades in trading period, $b_i$ and $s_i$ ($i = 1, \ldots, m$) are the $i$-th trade’s starting and ending day, respectively. Then the final equity $E_f$ of this stock is given as follows:

$$E_f = E_t = \prod_{i=1}^{m} p_{s_i} (1 - c)$$

where transaction cost $c$ is set to 0.1% and initial equity $E_t$ is set to $100,000 in this study.

Given a financial market with a number of stocks, each stock in the market is assigned the same amount of initial equity. During trading, all equity available for a stock can only be used to invest in this stock. Because we do not allow short selling, the equity for each stock is always positive. On the last day of trading, we sell all stocks in hand and sum their equities together to obtain the final equity. Then the annual net profit made by the trading strategy can be calculated as follows:

$$\text{Annual net profit} = \left( \frac{\sum E_{k,i}}{\sum E_{t,i}} \right)^{\frac{1}{y}} - 1.0 \times 100\%$$

where $E_{k,i}$ and $E_{f,i}$ is the initial and final equity of the $k$-th stock, respectively, and $y$ is the length of trading period in years.

In Eq. (10), we have already included the consideration of transaction cost so that the annual net profit represents the average annual profit net of the transaction cost. For each buy or sell, 0.1\% of the total turnover is cut from the equity as transaction cost in our study.

3. Particle swarm optimization for PRS

For PRS, there are 140 starting weights ($w_1$ to $w_{140}$), two thresholds ($\theta_1$, $\theta_2$), two time spans ($\bar{\theta}_1$, $\bar{\theta}_2$), and a reward factor ($b_k$) to be determined, thus the search space dimension for this optimization problem is 145. Unlike the traditional function optimization problem, the evaluation function of trading rules optimization is non-differentiable. Therefore, identifying such a high dimensional parameter vector is a tough optimization problem. In the literature, genetic algorithms (GA) and particle swarm optimization (PSO) are the most popular optimization algorithms used to optimize trading rules. In this work, we choose PSO rather than GA because PSO is more easily to be implemented and can achieve the same performance with higher computing efficiency as compared with GA (Lee et al., 2005).

3.1. Particle swarm optimization

Particle swarm optimization (PSO) is a stochastic evolution algorithm based on swarm intelligence, which was first introduced by Kennedy and Eberhart (1995). Since its inception, PSO has shown great success in solving function optimization problems and has been widely applied in a variety of engineering applications (Robinson & Rahmat-Samii, 2004; El-Zonkoly, Khalil, & Ahmied, 2009; Coelho, 2010; Hsu et al., 2011).

PSO is motivated by the behavior of bird flocks in finding food. Suppose a flock of birds want to find food, but they do not know where the food is before they find it. However, this bird flock can always find their food at last. PSO uses a swarm of particles to simulate these birds. Each particle has a random initial position $\mathbf{x}$ and velocity $\mathbf{v}$. The position $\mathbf{x}$ is an $n$-dimensional vector which represents a possible solution to the $n$-dimensional optimization problem. The objective function targeted to be optimized is used to evaluate the fitness of each particle’s position. Higher fitness means a better position. For each particle, PSO uses $\mathbf{x}$ to record the best position this particle has arrived. For the whole swarm, $\bar{x}$ is used to record the global best position achieved by all particles. At time $t$, PSO updates each particle’s velocity using the following equation:

$$\mathbf{v}_{t+1} = \lambda \mathbf{v}_t + c_1 \mathbf{r}_1 (\mathbf{x}_t - \mathbf{x}_s) + c_2 \mathbf{r}_2 (\mathbf{x}_t - \mathbf{x}_g)$$

where $\lambda$ is the inertia weight, $c_1$, $c_2$ are the acceleration coefficients, $r_1$ and $r_2$ are two random numbers in the range between 0 and 1 (Shi & Eberhart, 1998).

The first term of Eq. (11) indicates an inertia for a particle wandering in the search space. The second term represents self-cognition of past experience of a particle, i.e. the particle tends to move towards its past best position. Similarly, the third term indicates that particles have social cognition to the whole swarm and are attracted by the global best position. For PSO, inertia weight $\lambda$ controls the influence of previous velocity on a particle. A large $\lambda$ allows particles to explore more search space for the optimal position (global exploration), while small $\lambda$ helps search in a local area for the exact solution (local exploitation). Coefficients $c_1$ and $c_2$ control the influence of $\mathbf{x}$ and $\mathbf{x} \bar{s}$ on particles’ movement. A higher value of $c_1$ means each particle is more likely to be attracted to a different position, so the whole swarm is more widespread in the search space. Its effect is similar to $\lambda$. In contrast a high value of $c_2$ leads all particles converge to the current global best position.

After updating the velocity, each particle will move to a new position according to:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}$$

This particle movement will repeat iteratively till a termination criterion is met, e.g. the number of iterations or all particles converge to a quasi-optimal position.

In PSO, the trade-off between global exploration and local exploitation of particles is the main influencing factor to PSO’s performance (Shi & Eberhart, 1998). Generally, exploration should be enhanced at the early stage of searching so that more search space can be explored by particles. While at the later stage the algorithm should focus on exploitation to find the exact and accurate optimum.

To achieve these goals, Shi and Eberhart (1999) suggested to reduce PSO’s inertia weight $\lambda$ linearly over the iterations so as to help particles to find the optimal position more efficiently. In later work, Ratnaweera et al. (2004) introduced a time-varying acceleration coefficients (TVAC) in PSO, which linearly reduces the first acceleration coefficient $c_1$ and linearly increase the second acceleration coefficient $c_2$ over the iterations. Based on these studies, in this paper we linearly updates $\lambda$, $c_1$ and $c_2$ according to the following iterative equations:

$$\lambda_t = (\lambda_f - \lambda_i) \frac{t}{\text{max} t} + \lambda_i$$

$$c_{1t} = (c_{1f} - c_{1i}) \frac{t}{\text{max} t} + c_{1i}$$

$$c_{2t} = (c_{2f} - c_{2i}) \frac{t}{\text{max} t} + c_{2i}$$

where $\lambda_i$, $\lambda_f$ and $\lambda_f$ denote the initial, current and final value of $\lambda$, respectively (similar for $c_1$ and $c_2$), $t$ is the current iteration number.
and maxt is the maximum number of iterations. Following Shi and Eberhart (1999) and Ratnaweera et al. (2004), we set \( x_1 = 0.9 \), \( \dot{x}_p = 0.4 \), \( c_1 = 2.5 \), \( c_2 = 0.5 \), \( c_3 = 0.5 \) and \( c_4 = 2.5 \) in this study.

### 3.2. Optimization of PRS

As aforementioned, the search space dimension of PRS optimization is 145. The mapping of the 145-dimensional position vector to the PRS parameters for each particle in PSO is displayed in Fig. 2. The first 140 dimensions are the weights of component rules. The following dimensions are buy threshold \( (\theta_1) \), sell threshold \( (\theta_2) \), memory span \( (\theta_3) \), review span \( (\theta_4) \) and reward factor \( (\theta_5) \), respectively.

Recall that the sum of component rules' starting weights is restricted to be 1, i.e. \( \sum w_i = 1 \). It is difficult to satisfy this constraint if the weights are optimized directly by PSO because of its stochastic nature. In this regard, a new parameter \( x_i \) is introduced here and a one-to-one transformation between \( w_i \) and \( x_i \) is used in this study:

\[
 w_i = \frac{e^{x_i}}{\sum e^{x_i}} \tag{14}
\]

As Eq. (14) guarantees that the sum of \( w_i \) is 1 regardless of the value of \( x_i \), the new parameter \( x_i \) is optimized to get the best set of starting weights via Eq. (14).

The ultimate goal of any stock trading strategy is to make profit from the stock market, so the parameters optimization of PRS is led by this goal. Fig. 3 gives the fitness evaluation of a particle in the swarm. PRS's parameters are first initialized using the particle’s position, and then the annual net profit generated by the PRS on the stock data is returned as the fitness.

### 4. Application for NASDAQ100

#### 4.1. Data

The constituent stocks of NASDAQ100, which are 100 of the largest domestic and international non-financial stocks on the NASDAQ Stock Market, are considered in our study. The daily close stock price from 1994 to 2010 are collected from Reuters 3000xtra. Because not all stocks were issued before 1994, only 52 stocks having data through the whole period are considered in our experiments. An eight years’ daily data from 1995 to 2002 is used for optimizing, in other words, training the PRS. Another eight years’ daily data from 2003 to 2010 is used for testing the profitability of PRS. The data in 1994 is reserved for data preparation in training as all component rules need data from previous days to generate signals (e.g. TRB with 250 days channel needs data over past 250 days to calculate the trading range).

#### 4.2. Optimization set-up

The swarm size is set to 250 and the maximum number of iterations is set to 400. There is also a stopping criterion, that is, if there is no improvement in the global best for at least 50 iterations, the optimization will stop. Table 1 gives the search space boundary of PRS optimization. Using a wide range of \( x \) would make the starting weights of PRS component rules vary significantly after optimization. Our experiment reveals that this may overfit the training, i.e. PRS may depend on only several component rules with relatively high weights, and produce extremely good performance in the training period but possibly poor performance in the testing period. To overcome this, the range of \( x_i \) is set to \([-1.1\) so that the range of starting weights is approximately \([0.001, 0.05]\). It is noteworthy to comment that the weights at the other trading days can be out of this range after a number of weight adjustments via the reward/penalty mechanism.

For buy threshold and sell threshold, \([-0.9, 0.9]\) is wide enough to cover the threshold range, so they are set as the lower and upper bound of \( \theta_2 \) and \( \theta_1 \), respectively. There are about 21 trading days in a month and about 252 trading days in a year, so the minimum review span \( \theta_4 \) is set to be shortly less than one month and the maximum memory span \( \theta_3 \) is set to be a little bit longer than one year in terms of trading days. Because \( \theta_3 \gg \theta_4 \), the minimum value of \( \theta_3 \) and the maximum value of \( \theta_4 \) are both set to 150. Reward and penalty for component rules should not be too significant each time, so the maximum value of reward factor \( \theta_5 \) is set to 1. The minimum value of \( \theta_5 \) is 0 means that there is no reward and penalty in this case.

### 4.3. Performance

After optimization, we obtain the optimal set of PRS parameters that makes PRS earn significantly in the training period. The optimal starting weights of the 140 component rules are displayed in Fig. 4. Table 2 gives the summary statistics of these starting weights and the other 5 optimal PRS parameters. From the table, we could find that the buy threshold \( \theta_1 \) and sell threshold \( \theta_2 \) are 0.0438 and \(-0.8051\), respectively. It means that PRS is more rigorous in the control of sell signal than buy signal. It can also be found that the memory span \( \theta_3 \) is about one trading year (252 days) and the review span \( \theta_4 \) is just several days less than one month and the maximum memory span \( \theta_3 \) is set to be a little bit longer than one year in terms of trading days. Because \( \theta_3 \gg \theta_4 \), the minimum value of \( \theta_3 \) and the maximum value of \( \theta_4 \) are both set to 150. Reward and penalty for component rules should not be too significant each time, so the maximum value of reward factor \( \theta_5 \) is set to 1. The minimum value of \( \theta_5 \) is 0 means that there is no reward and penalty in this case.

#### 4.3. Performance

After optimization, we obtain the optimal set of PRS parameters that makes PRS earn significantly in the training period. The optimal starting weights of the 140 component rules are displayed in Fig. 4. Table 2 gives the summary statistics of these starting weights and the other 5 optimal PRS parameters. From the table, we could find that the buy threshold \( \theta_1 \) and sell threshold \( \theta_2 \) are 0.0438 and \(-0.8051\), respectively. It means that PRS is more rigorous in the control of sell signal than buy signal. It can also be found that the memory span \( \theta_3 \) is about one trading year (252 days) and the review span \( \theta_4 \) is just several days less than one trading month (84 days). The reward factor \( \theta_5 \) is close to the upper bound 1, which implies the great change of component rules’ weights during trading.

As discussed in Section 2.3, the strategy becomes weight strategy (WS) if reward factor \( \theta_5 \) is set to zero. The profitability of PRS and WS are then tested in the testing period and compared with the best MA rule (125–150) and the best TRB rule (125) of this period in terms of the annual net profit (ANP). In addition to the annual net profit, two more performance measurements – annual sharpe ratio and payoff ratio – are considered for comparison. Sharpe ratio (Sharpe, 1994) measures the excess return per unit of deviation in the investment equity and is typically used as risk measurement. Sharpe ratio is defined as:

\[
\text{Sharpe ratio} = \frac{E[R_e - R_f]}{\sigma} = \frac{E[R_e - R_f]}{\sqrt{\text{var}[R_e - R_f]}} \tag{15}
\]

where \( R_e \) is the equity return, \( R_f \) is the risk free rate such as the US 3-month Treasury Bill Rate. \( E[R_e - R_f] \) is the expected differential return of the equity return over the risk free rate, and \( \sigma \) is the standard deviation of this differential return. In this study, we use the US 3-month Treasury Bill Rate from 2003 to 2010 as the risk free rate.

Payoff ratio is simply the average profit of winning trades divided by the average loss of losing trades. It is widely used by traders to compare the expected return to the amount of capital at risk. The higher the sharpe ratio and payoff ratio, the better the trading rule. The results are shown in Table 3. The number of trades (No. trades) and the percentage of profitable trades (Win%) for each trading rule in the testing period are also given.
In the testing period, the MA rule with 125 and 150 days moving averages and the TRB rule with 125 days trading range make the highest annual net profit among 119 MA rules and 21 TRB rules, respectively. From Table 3, we can find that MA (125–150) is the highest profitable rule among all component trading rules. Both PRS and WS make higher annual net profit than the best MA and TRB rules in the testing period, which means combining various technical trading rules together appropriately can really get better performance. When compared with WS, PRS generates higher annual net profit as well as higher sharpe ratio and payoff ratio. The result indicates that it worths to conduct the reward/punishment mechanism which makes PRS more adaptive to the performance change of component rules.

Table 4 gives more details about the 52 stocks' profits (each stock is assigned an initial equity of $100,000) of PRS, WS and the two best component rules in the testing period. PRS, WS and TRB (125) earn profit in 41 out of 52 stocks, which is 7 stocks more than MA (125–150). The total gain (20.89 million) of profitable stocks generated by PRS is higher than all other rules. And the total loss (−0.20 million) of nonprofitable stocks generated by PRS is lower than them too. It shows that PRS can generate significant profit (41 stocks with 20.89 million net profit) from those profitable stocks and recover from those few nonprofitable stocks (11 stocks with −0.20 million loss).

To get a more comprehensive understanding to the performance of these trading rules, their equity curves in the testing period are given in Fig. 5. The initial equity for single stock is $100,000 so the initial equity for the market (52 stocks) is $5.2 million. There are exactly 2015 trading days in the testing period which ranges from January 2, 2003 to December 31, 2010, and hence there are approximately 252 trading days per year. It can be seen that PRS keeps the highest cumulative equity during the eight years of testing period.

In the training period, the MA rule with 225 and 150 days moving averages and the TRB rule with 125 days trading range make the highest annual net profit among 119 MA rules and 21 TRB rules, respectively. From Table 3, we can find that MA (125–150) is the highest profitable rule among all component trading rules. Both PRS and WS make higher annual net profit than the best MA and TRB rules in the testing period, which means combining various technical trading rules together appropriately can really get better performance. When compared with WS, PRS generates higher annual net profit as well as higher sharpe ratio and payoff ratio. The result indicates that it worths to conduct the reward/punishment mechanism which makes PRS more adaptive to the performance change of component rules.

Table 4 gives more details about the 52 stocks' profits (each stock is assigned an initial equity of $100,000) of PRS, WS and the two best component rules in the testing period. PRS, WS and TRB (125) earn profit in 41 out of 52 stocks, which is 7 stocks more than MA (125–150). The total gain (20.89 million) of profitable stocks generated by PRS is higher than all other rules. And the total loss (−0.20 million) of nonprofitable stocks generated by PRS is lower than them too. It shows that PRS can generate significant profit (41 stocks with 20.89 million net profit) from those profitable stocks and recover from those few nonprofitable stocks (11 stocks with −0.20 million loss).

To get a more comprehensive understanding to the performance of these trading rules, their equity curves in the testing period are given in Fig. 5. The initial equity for single stock is $100,000 so the initial equity for the market (52 stocks) is $5.2 million. There are exactly 2015 trading days in the testing period which ranges from January 2, 2003 to December 31, 2010, and hence there are approximately 252 trading days per year. It can be seen that PRS keeps the highest cumulative equity during the eight years of testing period.

In the training period, the MA rule with 200 and 250 days moving averages generates the highest annual net profit among all of
the 119 MA rules, and the TRB rule with 200 days trading range generates the highest annual net profit among all of the 21 TRB rules. To demonstrate that the performance of these rules may fluctuate over time, we also test the performance of these two rules in the testing period. The results are shown in Table 5. The MA (200–250) makes the highest profit among all MA rules in the training period, but its performance drops a lot in the testing period, even is below the average performance of all MA rules. Although the TRB (200) performs well in both of the training and testing period, it is not the best TRB any more in the testing period. This result gives support to our complex trading strategy with adequate combination of technical trading rules.

In the testing period from 2003 to 2010, all PRS component rules’ weights are dynamically changed by the reward/penalty mechanism. The first updating occurs at the end of year 2003 as the memory span is 251 days and each trading year has about 252 trading days. Then all weights are updated every 78 days (review span) in the following 7 years, and hence each weight is updated 24 times in all. In Fig. 6 we display the summary statistics (maximum, minimum and mean weight) of each component rule’s 25 weights (1 starting weight and updated 24 times) in the testing period. It can be found that some rules’ weights vary dramatically (e.g. MA (25–70) and TRB (60)) but some others are just changed slightly (e.g. MA (125–150)). Despite the weight variation, some rules such as MA (25–70) always have weights higher than the mean weight of all rules (dashed line in figure). Some rules such as TRB (125), on the contrary, always have weights lower than the mean. Fig. 7 provides more details about weights variation of several typical rules labeled in Fig. 6. It shows the profile plots of three MA rules’ and three TRB rules’ weights in the testing period. As previously mentioned, MA (25–70) keeps high weight during the whole trading period even though there is small fluctuation, and TRB (125) has weights lower than the mean all the time. In addition, as shown by rule MA (10–15) and TRB (60), a high weight rule may be penalized much and becomes a low or even zero weight rule, and vice versa.

### 4.4 The effect of transaction cost

To access the impact of transaction cost on the optimization of PRS, we consider two more transaction costs, 0.2% and 0.5%, in our application. First we look at the performance of PRS optimized

---

**Table 4**

Stock profit summary of PRS, the best MA and TRB rules in the testing period.

<table>
<thead>
<tr>
<th></th>
<th>Profitable stocks(^a)</th>
<th>Non-profitable stocks(^a)</th>
<th>All stocks (profit ratio)(^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PRS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. Stocksa</td>
<td>41</td>
<td>11</td>
<td>52 (78.8%)</td>
</tr>
<tr>
<td>Gain (million)</td>
<td>20.89</td>
<td>−0.20</td>
<td>20.69</td>
</tr>
<tr>
<td><strong>WS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. stocks</td>
<td>41</td>
<td>11</td>
<td>52 (78.8%)</td>
</tr>
<tr>
<td>Gain (million)</td>
<td>16.02</td>
<td>−0.28</td>
<td>15.74</td>
</tr>
<tr>
<td><strong>MA (125–150)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. stocks</td>
<td>34</td>
<td>18</td>
<td>52 (65.4%)</td>
</tr>
<tr>
<td>Gain (million)</td>
<td>15.89</td>
<td>−0.45</td>
<td>15.44</td>
</tr>
<tr>
<td><strong>TRB (125)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. stocks</td>
<td>41</td>
<td>11</td>
<td>52 (78.8%)</td>
</tr>
<tr>
<td>Gain (million)</td>
<td>12.15</td>
<td>−0.25</td>
<td>11.90</td>
</tr>
</tbody>
</table>

\(a\) Profitable stock means the stock whose final equity is more than its initial equity and vice versa.

\(b\) Profit ratio = (number of profitable stocks)/(total number of stocks).

\(c\) No. stocks is the number of stocks.

---

**Table 5**

Annual net profit (ANP) of the MA and TRB rules in the testing period.

<table>
<thead>
<tr>
<th>Rule</th>
<th>ANP (%)</th>
<th>MA (200–250)</th>
<th>Best MA (125–150)</th>
<th>Worst MA (1–5)</th>
<th>MA mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANP (%)</td>
<td></td>
<td>12.1245</td>
<td>18.8043</td>
<td>3.9078</td>
<td>13.4732</td>
</tr>
<tr>
<td>Rule</td>
<td>TRB (200)</td>
<td></td>
<td>Best TRB (125)</td>
<td>Worst TRB (10)</td>
<td>TRB mean</td>
</tr>
<tr>
<td>ANP (%)</td>
<td></td>
<td>15.4974</td>
<td>16.0409</td>
<td>3.7756</td>
<td>12.7453</td>
</tr>
</tbody>
</table>
with different transaction costs over the testing period. The results are shown in Table 6. It is not surprising to see that all of the three performance measurements, annual net profit (ANP), Sharpe ratio and payoff ratio, decrease in transaction cost as more transaction cost is paid for each trade. The impact of transaction costs on PRS not only lies in the profit, but also lies in the parameters of PRS. The high transaction cost may lead to a more conservative PRS. This can be evidenced from Table 7 which shows how two of the PRS parameters (memory span \( h_3 \) and reward factor \( h_5 \)) vary with the transaction cost. The increasing memory span indicates that PRS tends to evaluate component rules over a longer performance review period, and the decreasing reward factor implies that PRS tends to update the weights of component rules less dramatically.

5. Bootstrapping test of PRS

First used by Brock et al. (1992), bootstrapping methodology has been widely used in the literature to assess the trading rule performance in both of the stock and foreign exchange market (Levich & Thomas, 1993; Neely, Weller, & Dittmar, 1997; Mills, 1997; Sullivan et al., 1999; Lento & Gradojevic, 2007). In bootstrapping, a null model for data-generating process is fitted to the stock prices and the model's parameters are estimated based on the stock prices. Residuals from the null model are then re-sampled with replacement and substituted to generate a large number of simulated stock prices sets. The simulated stock prices will retain all the characteristics of the original stock prices such as drift in price and volatility, but lose all serial dependence not captured by the model. The bootstrapping simulations are then used to test the trading rule and determine whether it simply exploits known

<table>
<thead>
<tr>
<th>Transaction cost (%)</th>
<th>ANP (%)</th>
<th>Sharpe ratio</th>
<th>Payoff ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>22.2210</td>
<td>1.0897</td>
<td>8.4389</td>
</tr>
<tr>
<td>0.2</td>
<td>19.9830</td>
<td>1.0369</td>
<td>8.3488</td>
</tr>
<tr>
<td>0.5</td>
<td>19.1129</td>
<td>0.9608</td>
<td>6.2302</td>
</tr>
</tbody>
</table>

Fig. 6. Maximum, minimum and mean weight of each component rule's 25 weights (1 starting weight and updated 24 times) in the testing period. The first 119 rules are MA rules and the other 21 rules are TRB rules. The mean weight of each component rule is indicated by the solid bold line between its maximum and minimum weights. As highlighted in Section 2.3, the mean weight of all component rules is constant \((1/140 = 0.0071)\) during trading and is indicated as the horizontal dashed line.

Fig. 7. The profile plots of the weights of six component rules in the testing period. The mean weight of all component rules is 0.0071 and represented as the horizontal dashed line. The first weight updating occurs at the end of year 2003 as the memory span is 251 days (Table 2) and each year has about 252 trading days. Then all weights are updated every 78 days (review span), and hence each weight is updated 24 times in all.

Table 6

Performance of PRS with different transaction costs.
statistical properties of the original stock prices. If the trading rule results based on bootstrapping simulations differ from the results based on original data greatly, it indicates that the trading rule could make profit by finding some useful information hiding in the original price data rather than by good fortune.

Given a set of stock prices \( p = \{ p_0, \ldots, p_T \} \), its return series are calculated as follows:

\[
 r_t = \ln \frac{p_{t+1}}{p_t}, \quad t = 1, \ldots, T
\]  

A set of resampled residuals \( e' = \{ e'_1, \ldots, e'_T \} \) is obtained by drawing \( T \) items with replacement from the original residuals \( e = \{ e_1, \ldots, e_T \} \) generated by the null model. Typically, a new set of returns \( r' = \{ r'_1, \ldots, r'_T \} \) is generated with the scrambled residuals \( e' \). A simulated stock price series is then generated according to the scrambled returns:

\[
 p'_t = p'_{t-1} \exp (r'_t), \quad t = 1, \ldots, T
\]  

Table 7

<table>
<thead>
<tr>
<th>Transaction cost (%)</th>
<th>( h_1 ) (memory span)</th>
<th>( h_2 ) (reward factor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>251</td>
<td>0.9660</td>
</tr>
<tr>
<td>0.2</td>
<td>290</td>
<td>0.9049</td>
</tr>
<tr>
<td>0.5</td>
<td>299</td>
<td>0.8283</td>
</tr>
</tbody>
</table>

Fig. 8. Bootstrapping test from (a) random walk, (b) AR(1) and (c) GARCH(1,1) model. The excess annual net profit generated by PRS on the original stock prices (0.69%) is located in the interval \( \left( 0.5\%, 1.5\% \right) \) and counted to the black bar (indicated by arrow) in histogram.
where \( p_0 = P_0 \). If the null model is true, the simulated stock price series possesses the statistical properties of the original prices and the trading rule results should be similar between the original stock prices and the simulated prices.

Follow Brock et al. (1992), excess return (excess annual net profit in this paper) is used as the fitness of a trading rule and is given as follows:

\[
\text{Excess annual net profit} = \Delta P = P_t - P_{bh}
\]  

(18)

where \( P_t \) and \( P_{bh} \) is the annual net profit generated by PRS and buy-and-hold strategy (buy the first day, sell the last day), respectively. The simulated \( p \)-value is calculated as the fraction of simulations generating excess annual net profit larger than the original result. Lower \( p \)-value means more significant of the trading rule performance on the original stock prices.

5.1. Null models for bootstrapping

Follow the previous study, three null models were tested: a random walk model, an autoregressive AR(1) model and a GARCH(1,1) model. The random walk model is simply:

\[
r_t = e_t
\]  

(19)

where \( r_t \) is the return on day \( t \) and residuals \( e_t \) are IID random variables. Therefore, the bootstrapped stock prices were simulated by taking the scrambled returns of original stock prices.

The AR(1) model is:

\[
r_t = b + r_{t-1} + e_t, \quad |b| < 1
\]  

(20)

where \( r_t \) is the return on day \( t \) and \( e_t \) are the IID residuals. The AR model suggests that returns over short periods of time are positively correlated. The parameters \( (b, \rho) \) and residuals were estimated through ordinary least squares (OLS). The residuals were then scrambled and simulated stock prices were generated using the estimated parameters and scrambled residuals.

The last model is GARCH(1,1) model:

\[
\begin{align*}
    r_t &= \mu + \sigma_t e_t \\
    e_t &= \sigma_t e_t, \quad e_t \sim N(0, 1) \\
    \sigma_t^2 &= \omega + \alpha_1 e_{t-1}^2 + \beta_1 \sigma_{t-1}^2
\end{align*}
\]

(21)

where \( \omega > 0, \alpha_1 \geq 0, \beta_1 > 0 \) and \((\alpha_1 + \beta_1) < 1\), and \( e_t \) is the IID residuals. The GARCH model suggests that periods of large (low) price changes tend to be followed by periods of large (low) price changes. The parameters \((\mu, \omega, \alpha_1, \beta_1)\) and residuals \( e_t \) were estimated by maximum likelihood. Bootstrapped stock prices were obtained by re-sampling the residuals \((e_1, \ldots, e_T)\) and using Eqs. (17) and (21).

5.2. Bootstrapping test results

As previous studies (Brock et al., 1992; Neely et al., 1997) suggested, 500 bootstrapping simulations were sufficient to estimate a concrete \( p \)-value. The original stock prices are from the 52 stocks in the testing period, and hence each simulation includes 52 bootstrapped stock prices with a length of 8 years. Fig. 8 shows the excess annual net profit histogram of 500 bootstrapping simulations based on each null models. The length of each interval is 1%. For excess ANP greater than or equal to 5.5%, they are counts together in the histogram. And for excess ANP less than −19.5%, they are counts together. The excess annual net profit generated by PRS on the original stock prices is 0.69%. Therefore it is located in the interval [0.5%, 1.5%] and is counted to the black bar (indicated by arrow) in all sub-figures of Fig. 8. For random walk model, there are only 4 out of 500 simulations generate higher excess annual net profit than for the original data. For AR(1) model and GARCH(1,1) model, 7 and 5 out of 500 simulations yield greater excess annual net profit, respectively. Therefore, the simulated \( p \)-value of random walk, AR(1) and GARCH(1,1) model are 0.008, 0.014 and 0.010, respectively. It demonstrates that the trading result of PRS is not consistent with these null models and PRS has good predictive ability in the testing data set.

6. Conclusions and future works

In this paper, we propose a complex stock trading strategy namely performance-based reward strategy (PRS) by combining a large number of technical trading rules. The contribution of this study can be concluded as three points. First, a reward/penalty mechanism based on component rules’ recent trading performance is proposed to update their weights over time. Regardless of the performance fluctuation of component rules over a long time, PRS is able to learn good rules from them and update their component weights dynamically. Second, an improved time variant particle swarm optimization (TVPSO) is adopted to optimized PRS parameters in an eight-year training period. The out-of-sample test shows that PRS can be successfully optimized by TVPSO. Third, our empirical results found that PRS is able to identify the optimal combined trading strategy which outperforms individual trading rules in terms of annual net profit. The results are further accessed by bootstrapping methodology with three null models: a random walk model, an autoregressive AR(1) model and a GARCH(1,1) model. The bootstrapping test implies that PRS is able to find some useful information which is not captured by these models in the stock market and generates high profit.

For our future research, more types of technical trading rules could be used as component rules of PRS. Some examples are relative strength index rules based on the momentum oscillator, and on-balance volume rules based on the daily stock volume and so forth. However, this may lead to a large number of PRS component trading rules, and many of these component rules are likely to perform poorly all of the time. Although these poorly performed component rules will be penalized to receive small weights, combining many such trading rules may still badly affect the combined signal of PRS. Therefore, to avoid this, an appropriate rule screening method should be considered before combining the signals of component rules. Furthermore, the optimization of complex trading strategy involves a large number of component rules will take quite a long time for even just one trial. This is a main barrier in searching for optimal trading strategy. One common approach to speedup the optimization is to employ parallel computing, and several widely used distributed computing platforms such as Hadoop and Spark can be considered. Finally, some nonlinear weighting methods such as artificial neural networks (ANN) can also be considered in our future study.

Acknowledgment

The authors would like to thank the editor and the two reviewers for their thoughtful comments that led to a great improvement of this article.

Appendix A. Parameter values of PRS component rules

A.1. Moving average rules

\[
\begin{align*}
    m &\quad \text{(number of days in a long-period moving average)} = 5, 10, 15, 20, 25, 30, 40, 50, 75, 100, 125, 150, 200, 250 \quad (14 \text{ values}) \\
    n &\quad \text{(number of days in a short-period moving average)} = 1, 2, 5, 10, 15, 20, 25, 30, 40, 50, 75, 100, 125, 150, 200 \quad (15 \text{ values})
\end{align*}
\]
Because $n$ should be less than $m$, the total number of MA rules generated is 119.

### A.2. Trading range break-out rules

$n$ (number of days for a trading range) = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 75, 80, 90, 100, 125, 150, 175, 200, 250 (21 values).

Because there is only one parameter, the total number of TRB rules generated is 21.

### References


