A Multiobjective Optimization Approach for Market Timing

Ismail Mohamed
University of Kent
Chatham Maritime, Kent, United Kingdom
im572@kent.ac.uk

Fernando E. B. Otero
University of Kent
Chatham Maritime, Kent, United Kingdom
f.e.b.otero@kent.ac.uk

ABSTRACT

The introduction of electronic exchanges was a crucial point in history as it heralded the arrival of algorithmic trading. Designers of such systems face a number of issues, one of which is deciding when to buy or sell a given security on a financial market. Although Genetic Algorithms (GA) have been the most widely used to tackle this issue, Particle Swarm Optimization (PSO) has seen much lower adoption within the domain. In two previous works, the authors adapted PSO algorithms to tackle market timing and address the shortcomings of the previous approaches both with GA and PSO. The majority of work done to date on market timing tackled it as a single objective optimization problem, which limits its suitability to live trading as designers of such strategies will realistically pursue multiple objectives such as maximizing profits, minimizing exposure to risk and using the shortest strategies to improve execution speed. In this paper, we adapt both a GA and PSO to tackle market timing as a multiobjective optimization problem and provide an in depth discussion of our results and avenues of future research.

ACM Reference Format:
Ismail Mohamed and Fernando E. B. Otero. 2020. A Multiobjective Optimization Approach for Market Timing . In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3390156

1 INTRODUCTION

The introduction of electronic exchanges in the USA following the market crash of 1987 was a pivotal point in the history of financial markets as it brought with it algorithmic trading [1]. A core issue faced by designers of algorithmic trading strategies is when to buy or sell a given security on a financial market, also known as market timing [2]. A common technique used to build market timing strategies is to use a collection of components that process the security's market context and return a recommendation on whether to buy or sell. Depending on the type of component, this context could be current and previous price action for the security at hand (technical analysis) or information regarding the entity that has issued the security on the market (fundamental analysis). Each component will have a set of parameters that affects its behavior and a weight that affects the impact of the recommendation returned by it. The overall decision on when to buy or sell then becomes the aggregate vote of the components selected to make up the strategy. It is the job of the designer of the strategy to select the appropriate components to use and set their parameters.

Designers of trading algorithms turned to computational intelligence metaheuristics to assist them in addressing the different issues faced when building such algorithms, including market timing. One such metaheuristic is Particle Swarm Optimization (PSO) [3]. Although popular in other domains, PSO has seen a more limited application within the domain of market timing. Compared to Genetic Algorithms (GA), which has been the most widely used in terms of volume [4, 5]. The earliest GA application in market timing was seen in 1999 [6], while PSO saw its first publication in 2011 [7]. The vast majority of these applications model market timing as a single objective optimization problem to the best of our knowledge, where the goal would be to tune the strategy to maximize a financial metric usually representing profits. In order to tune the strategy, they would either optimize the selection of a subset of components from a set of components with preset values for the parameters or tune the parameters of a collection of components that was selected beforehand. In [8, 9], the authors used PSO to compose a market timing strategy by optimizing both the selection of components and the adjustment of their parameters simultaneously. Another issue addressed by the authors in [9] was the potential for overfitting while training, leading to unbalanced and poor performance in testing. This is caused by adopting a training and testing strategy known as Step Forward Testing [2], where a stream of price action data for a particular stock is used to form the training and testing data by dividing them along an arbitrary point, then taking the earlier data for training and the later data for testing. Since the training data produced by this procedure may not contain all the possible trends of movement in price data, there is a high probability that the algorithm will overfit to the trends observed in the training data, and perform poorly when exposed to a previously unobserved trends while testing. Based on [2], the authors attempted to remedy this shortcoming by exposing the algorithms to a variety of trends during training and testing and named this approach Trend Representative Testing.

As we mentioned earlier, the vast majority of work done on market timing has been on the basis of modeling the problem as a single objective optimization one. This, however, may not be sufficient to produce market timing strategies that are suitable for live trading. Designers of market timing will usually chase multiple objectives at the same time. For example, the designers may seek strategies that maximize profits, minimize losses and exposure to risk and do so with the least amount of components (both for the sake of comprehensibility and speed of execution). In this paper we improve on the work in [9] by considering market timing as a multiobjective optimization problem, and explore the adaptations required to the algorithms and results obtained.

GECCO '20, July 8–12, 2020, Cancún, Mexico
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3390156

The remainder of this paper is structured as follows. We provide a brief literature review of market timing and another brief literature review of using PSO in multiobjective optimization problems in Section 2. We then revisit the issue of market timing, and explain how we plan to approach it as a multiobjective optimization problem in Section 3. In Section 4, we discuss how we adapted both the PSO and GA algorithms first introduced in [8, 9] to tackle market timing in a multiobjective fashion. In Section 5 we discuss our experimental setup and critique the results obtained. We finally round things off in Section 6 with a conclusion and suggestions for future research.

2 RELATED WORK

In this section we will briefly describe related work done both within the domains of using computational intelligence in market timing, with a special focus on GA and PSO, and the work done on adapting PSO to tackle multiobjective optimization problems. We pay particular attention to PSO as our main goal is to build a variant of the metaheuristic that tackles market timing as a multiobjective optimization problem and produce competent strategies.

2.1 Market Timing

As mentioned earlier, GAs have seen a far wider adoption when it came to tackling market timing in comparison to PSO. One of the earliest works to tackle market timing was by Allen and Karjalanien [6], a GA is used 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 early approach was to use GA to directly optimize the parameters of one or more financial analysis indicators.Examples of such an approach can be seen in the work of de la Fuente et al. [10] and Subramanian et al. [11] – the latter tackling market timing using multiple fitness metrics, each in turn. Both of these approaches directly encode the indicator parameters into the GA chromosome, and use the evolutionary process 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 [5]. More recent approaches using GA to tackle market timing can be seen in the work of Kampouridis and Otero [12] and Kim et al. [13].

Particle swarm optimization (PSO) has not seen the popularity of GA in the space of market timing. The earliest PSO approach to tackle market timing was proposed by Briza and Naval, Jr. [7]. Inspired by Subramanian and colleagues [11], 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. Although the authors used a multiobjective approach to optimize for Sharpe Ratio and Percentage of Return, the number of technical indicators was limited to five and the indicators used predefined values for their parameters. The work presented in this paper significantly improves on it by considering 5 objectives, 63 technical indicators, optimizing both the number of indicators to use and their parameters. Chakravarty and Dash [14] 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. [15] used PSO to optimize a neural network that generated fuzzy rules for market timing and reported positive results.

More recently, Chen and Kao [16] 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 [17] 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. [18], 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 (a statistical method of analyzing time series data common in econometrics). Bera et al [19] 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 [20] 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 [21] 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.

Finally, the latest work on using PSO within the market timing domain has been [8, 9]. In [8] the authors proposed a PSO algorithm that can both select a suitable subset of components as well as tune their parameters. They then further improved on the algorithm to avoid the risk of overfitting by introducing Trend Representative Testing, where the algorithm is trained and tested on multiple trend type [9], and compared its performance against a GA.

2.2 Multiobjective PSO

The earliest approach to adapt PSO to tackle multiobjective optimization problems is the one by Moore and Chapman in 1999 [22]. This approach followed a dominance based scheme to attaining a Pareto front and employed the use of archives at two levels. The first archive is maintained per particle in the swarm and is used to track the non-dominated solutions discovered by each particle. The second (global) archive is used to track globally non-dominated solutions discovered by the swarm and, at the end of the algorithm run, it is used to represent the Pareto set discovered. When updating the state of particles via the PSO velocity equation, the authors would use a randomly selected solution from a particle's individual archive to stand in for a personal best. As for the neighborhood best, the selection is based on random selection amongst all the individual archives maintained by the neighbors of a given particle.

In 2002, Hu and Eberhart proposed a lexicographical approach for PSO to multiobjective optimization problems in [23]. Here, the authors set out to solve a MOP that utilized two fitness metrics. The authors used the simpler fitness metric to define the neighbors in a dynamic neighborhood of any given particle, and use the more complex fitness metric to select the neighborhood best. Personal bests in this scenario are the latest non-dominated solutions discovered by the particles as they traverse the search landscape. As this approach is limited to only being capable of optimizing two objectives, the authors revisited this scheme in [24] and adopted a dominance based scheme. An archive was added to keep track of non-dominated solutions discovered during the run of the algorithm, while measures were taken to improve the exploration aspect of the algorithm and allow it reach regions of the search landscape that were unreachable by the earlier algorithm. Coello Coello and Lechga proposed Multiobjective PSO (MOPSO), which is another dominance based algorithm to tackle MOPs [25]. The main feature of MOPSO is that it maintained a truncated archive, where priority in admission is given to new non-dominated solutions that occupy less densely populated regions of the current Pareto front. Personal bests are defined as the latest non-dominated solutions discovered by a given particle so far, while neighborhood bests are probabilistically selected from sparsely populated regions of the Pareto front.

In 2003, Zhang et al proposed a lexicographical based approach for tackling MOPs using PSO [26]. Here the authors maintained separate global and personal bests for every objective being optimized. Neighborhood bests are a synthetic average of all the objectives begin optimized. Personal bests could either be constructed in the same fashion or selected at random from the set of tracked personal bests across all objectives. Mostaghim and Teich attempted to improve the performance of MOPSO proposed by Coello Coello and Lechuga [27]. Their contribution can be summed up as the introduction of a new measure to improve the selection of neighborhood bests that lead to a faster convergence and improved diversity in the Pareto set returned. Another PSO variant proposed in 2003 is that of Zhang and Huang [28]. The main contribution of the authors here is that selection of a neighborhood best is based on probabilistic choice amongst solutions maintained in the archive that considers their distance from the particle at hand, favoring closer solutions. Personal bests in this scenario are the last non-dominated solutions discovered by the particles throughout the run of the algorithm.

More recent approaches include [5], [29], [30], [31] and [32]. In an attempt to improve the diversity of solutions in the Pareto front and prevent premature convergence, the author in [5] propose a multiobjective PSO with novel measures to address these challenges. Based on a dominance-based approach, the authors utilize a bounded external archive to maintain the non-dominated solutions discovered during the search. Each particle also maintains a personal archive of non-dominated solutions, from a which a personal best is selected to participate in velocity and state update. With each iteration, a subset of the members in the external archive is selected to form candidates from which a neighborhood best is selected to participate in velocity update. These candidates are selected based on density and potential metrics, where density estimates how close a given solution is to other solutions on the current Pareto front and potential estimates how close a given solution would be to the true Pareto front. The candidates in this subset are then scored based on entropy, and one is probabilistically chosen to represent the neighborhood best per particle. As for selecting a personal best, one is chosen from the personal archive that minimizes the hyperbox formed between the current particle's state, its previous velocity, its neighborhood best and the current contents of its archive. With the influence of density, potential and entropy, the multiobjective PSO traverses the solution landscape and returns the contents of the external archive at the end of the search as the discovered Pareto set. Another unique feature about this approach is that it is capable of adjusting its intertia, cognitive and social biases dynamically based on the delta of entropy displayed by the positions of the particles in the swarm between every iteration.

In [29], the authors proposed a PSO algorithm to perform clustering in a multiobjective fashion. The swarm's goal is to minimize the distance between two data points within a cluster (cohesion) and maximize the number of clusters within the dataset (connectivity). A particle here represents a possible assignment of a data point to a cluster. Using a dominance-based approach, the authors maintain an archive per particle that keeps track of the last discovered non-dominated solution discovered by each particle. The neighborhood best is selected at random from an archive formed by the union of all the particle archives. An external archive is used to keep track of all the non-dominated solutions discovered by the swarm, and at the end of algorithm's run, is used to represent the discovered Pareto set.

In [30], authors proposed dubbed Vortex Multi-Objective Particle Swarm Optimization (MOVPSO). The main shortcoming identified by the authors in typical dominance-based approaches of PSO (mainly based on the model defined in [22]) is the lack of diversity in the Pareto front. To address this shortcoming, the proposed algorithm traverses the search space using two alternating behaviors: convergence and dispersion. During convergence, the positions of the particles in the swarm is evaluated, and the swarm is attracted to positions that are furthest away from the swarm's center of mass based on Euclidean distance. The trajectories taken during convergence are linear. After convergence, the particles undergo dispersion from that previous convergence point, in trajectories that are circular in motion, with an ever increasing radius. With the constant alternation between convergence and dispersion, the authors hoped that the algorithm would be better in discovering the search space and returning a Pareto front with a higher diversity than the typical approaches.

In [31] the authors attempt to simplify the state update mechanism and improve diversity of solutions discovered with their Diversity Enhanced Multiobjective PSO (DEMPSO). In DEMPSO, the velocity update equation drops the cognitive component and relies solely on the bias and social components. A global archive is used to maintain all the non-dominated solutions discovered in DEMPSO. The neighborhood best for each particle is based on the member of the global archive that has the farthest cosine distance from its current position. The algorithm also balances exploration and exploitation by dynamically shifting between the two behaviors based on the current particles' velocities.

Finally, in [32], the authors tackle multiobjective problems with a relatively high number of objectives using a proposed PSO approach that promotes diversity in the Pareto front by utilizing multiple subswarms. Considered as a hybrid between dominance-based and lexicographical approaches, the proposed algorithm uses a set of subswarms in its search process, one for each objective function to be optimized. Each subswarm specializes in optimizing a single objective function. An external, bounded archive is used to keep track of all non-dominated solutions discovered across all the particles in all of the subswarms. When selecting a neighborhood best for velocity update, a particle considers a non-dominated solution from the archive based on only two objective functions. The first is the objective being optimized by the current subswarm. For the second, we consider the remaining objectives being optimized and normalize their values based on the solutions in the archive. We then compare the performance of the current particle against the normalized values of these objectives and select the objective where the current particle is doing the worst. The neighborhood best is then chosen out of the archive where it dominates the current particle based on these two objectives. Personal bests are considered to be the last best solution encountered in terms of the objective function being optimized by the particle's subswarm. This coevolutionary particle swarm based approached showed promising performance when compared against a number of other multiobjective optimization algorithms while testing them using two standard testing suites.

3 FINANCIAL METRICS

As we mentioned earlier, market timing is the issue of deciding when to buy or sell a given security on a financial market. In this paper we aim to model the issue of market timing as a multiobjective one, and this begins with defining the objectives we aim to optimize. These are presented next.

3.1 Annualized Rate of Returns (AROR)

The Annualized Rate of Returns (AROR) is defined as the amount of returns on initial capital invested adjusted to reflect an annual rate. AROR can be calculated as follows:

$$AROR_{simple} = \frac{E_n}{E_0} \times \frac{252}{n} \tag{1}$$

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. For our purposes, we aim to maximize this metric. When reading AROR values, a value of one means that we are just breaking even on our investments. Values above one indicate gains, while values below one indicate losses.

3.2 Annualized Portfolio Risk

Annualized portfolio risk is defined as the volatility of returns encountered during trading within a specific time period. Annualized portfolio risk is calculated as follows:

$$RISK = \sigma(returns) \times \sqrt{252} \tag{2}$$

where $\sigma(returns)$ is the standard deviation of the returns on the trades performed during backtesting. During multiobjective optimization, our aim is to minimize this metric. Since the annualized portfolio risk is presented in units of currency, ideally you would want to minimize this variance in returns as much as possible. A strategy with low variance means that the gains obtained were evenly distributed across the transactions that were made throughout the lifetime of the strategy. This would indicate that the strategy is stable – more or less of such transactions would result in gains or losses that can be easily estimated.

3.3 Value at Risk (VaR)

Value at risk represents the likely value of the initial capital in terms of units of currency we are liable to lose based on a given confidence level. This is calculated as follows:

$$VaR = \mu(daily\ returns) - c\sigma(daily\ returns) \tag{3}$$

where *daily returns* represents the returns achieved day by day during backtesting, μ represents the mean of those returns and σ represents the standard deviation of the daily returns. The confidence level c is a user controlled parameter that represents the statistical confidence of losses the user would like to see. A default value of 1.65 (representing 95% confidence) is used. This equation returns a negative number, and as we aim to minimize this metric, the result is always multiplied by -1.

3.4 Transaction Count

The number of transactions produced by a market timing strategy is significant in that it gives an indication of how stable that strategy is, and that is based on the concept of sample error. It is therefore advantageous to maximize the number of data points available, and hence maximize the number of transactions generated by the market timing strategy. As transactions have a cost in real trading, a system that generates a disproportionate number of transactions would result into a hefty cost that could obliterate any returns gained. In order to avoid such situations, and reach reasonable trade-offs, we simulated transaction cost while backtesting based on a fixed commission model. To calculate this value, we first calculate a working commission, which is a percentage of the value of the asset being bought or sold multiplied by the number of shares involved in the transaction. If the working commission does not drop below a minimum threshold or exceed a maximum threshold, then the working commission is considered the transaction cost; otherwise, the minimum or maximum threshold is considered the transaction cost depending on whether the working commission was below the former or exceeded the latter. This fixed commission model is based on Interactive Brokers (IB) fixed commission model. [1] The values used for the parameters of the commission model are:

- Percentage: 0.005%
- Minimum Threshold: $1.0
- Maximum Threshold: 1% of total transaction value.

[1] https://www.interactivebrokers.com/en/index.php?f=1590&p=stocks1

3.5 Solution Length

Solutions that are longer in length imply the involvement of a large number of signal generating components (technical indicators in our case). Reducing the number of components involved will reduce computational cost and improve comprehensibility by the end user. Therefore, we aim to minimize the number of components involved in a solution to attain these aforementioned goals. The solution length is determined by counting the number of components within the solution that have a weight above zero, i.e. components that have a positive contribution towards the aggregate signal that is generated. Any components that have a weight of zero have their contributions to the aggregate signal nullified, and are therefore not considered for solution length.

4 PROPOSED ALGORTITHMS

As this work is based on [9], the encoding of the problem and hence the individual representation for either the GA or PSO algorithms discussed in the paper is reused. We also use the basic algorithm structures for GA, PSO and PSO^S. The main modifications done in this paper are in regards to adapting these algorithms to perform multiobjective optimization. We decided to adopt a Pareto dominance based approach for multiobjective optimization, which would result in the algorithms returning a Pareto set of solutions across the five objectives being optimized. In the following subsections, we discuss the extensions proposed in order to tackle market timing using a dominance-based approach to multiobjective optimization. These include modifications to fitness evaluation, global archive management, GA selection operators, tracking personal bests for PSO and neighborhood selection in PSO. We also discuss a new pruning procedure for the PSO with Stochastic State Update (PSO^S) algorithm from [9]. In total, we propose three multiobjective algorithms for market timing.

4.1 General Modifications

The first modification we performed was to redefine fitness. As [9] used AROR as its sole optimized objective, evaluating fitness was straight forward: higher values are better. In this paper, we pursue the optimization of five financial metrics, and since we are using a dominance based approach, a solution can only be considered to be fitter than another if, and only if, the solution at hand is not worse than the one it is being compared to in all five metrics and better than it in at least one metric. All non-dominated solutions that are discovered throughout the run of the algorithm are then collectively known as the Pareto set, and that is the final result returned by the algorithms.

As we need to keep track of all the non-dominated solutions as they are discovered, a global archive is maintained by all three algorithms for the storage of all non-dominated solutions found. Upon the discovery of a new solution, it is compared against the current occupants of the global archive. If the solution is dominated by any of the current occupants, it is rejected. If the new solution remains non-dominated after comparison with current archive, then it is admitted. Upon admittance, if the new solution dominates one or more occupants, these are removed from the archive. The archive we used for our algorithms are unbounded, meaning they can store an arbitrary number of solutions with no limitations in terms of size. The global archive is updated at the end of every iteration of the algorithm. For GA this occurs after the generation of a new population, and for the PSO algorithms, this occurs after the particle state updates.

4.2 GA Modifications

Besides adopting a multiobjective fitness function and a global archive, the only other modification to the GA first described in [9] is modifying the tournament selection used in selecting parents for crossover and mutation. In order to perform tournament selection, we first select random individuals from the current population and attempt to admit them to a temporary archive. Candidates that are dominated by other candidates will either not be allowed admittance into the archive or be removed. A random selection is then made form the surviving, non-dominated occupants to represent the selected individual. This is performed twice with every crossover or one with every mutation event to produce the required parent(s) for the operation.

4.3 PSO Modifications

As with the GA, our first modification was to adopt a multiobjective fitness function and a global archive to maintain a Pareto front. For both PSO variants, our second modification is related to how we maintained personal bests for the particles to participate in the velocity update function. All the particles in our multiobjective PSO maintain a personal archive were non-dominated personal solutions are stored. After a state update, and the subsequent fitness evaluation, the particle state is considered for admittance into its own personal archive. As with the global archive, these personal archives are also unbounded. When it comes time to select a personal best to participate in the velocity update function, we select one at random from the the particle's personal archive. Our final modification to both PSO algorithms was how a neighborhood best is selected. For neighborhood selection, we use a temporary archive that stores the non-dominated neighbors of the particle at hand. Each neighboring particle will also select a solution at random from its own personal archive to be admitted into the temporary archive on the condition that it is non-dominated by the current solutions. A solution is then selected at random from this temporary archive to represent the neighborhood best.

Another issue we faced, was how to adapt the stochastic state update procedure of PSO^S to work with a multiobjective fitness. The solution we proposed for calculating the probabilities of updating the cognitive and social components is to use the dominance score. Given two solutions x and y, we define the the dominance score $d(x, y)$ as the number of objective where solution x is better than solution y. We can calculate the update probabilities in the following manner:

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

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

- x: particle

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

PSO		PSO^{SP}		**GA**	
Population	50	Population	43	Population	49
Iterations	67	Iterations	83	Generations	24
Neighbors	50	Neighbors	43	Mutation Probability	0.8087
c1	3.7543	c1	4.3644	Crossover Probability	0.1409
c2	2.502	c2	2.5136	Tournament Size	32
Clamp	Scaling	Clamp	Clerc		
Scaling Factor	0.5314	Scaling Factor	–		
		Pruning?	True		
		Pruning Threshold	0.0959		
		Pruning Deadline	2		

- i: current particle index
- y: personal best
- $\hat{y}$: neighborhood best
- $d(x, y)$: the dominance score of x over y
- $rand()$: random number between 0 and 1

We also modified the pruning procedure for PSO^S. In [9] the pruning procedure permanently removed components from all particles in the swarm if they fell below a threshold. During hyperparameter optimization, it was determined that this pruning was ineffective and it was turned off by the tuning algorithm. In this paper, we modified the pruning procedure to only ignore components when they fall below a specific threshold. This nullifies the impact of those components but allows for the possibility of their reactivation later in the algorithm's run during a velocity update procedure. We refer to this modified PSO^S with the updated probability of update calculation and updated pruning procedure as PSO^{SP}.

Table 2: Best Performance Per Objective. For every optimized objective, we find the best performing instance. The test strand where this is observed is in brackets next to the primary objective name. The best discovered solution per algorithm observed within the strand and objective at hand is then listed. The top performing solution is highlighted in bold. In case of a tie, we consider the objectives in a lexicographical approach using the following order: AROR, Portfolio Risk, VaR, Transactions Count and Solution Length.

Primary Objective (Strand)			PSO^{SP}	**GA**	**PSO**
AROR (ATRO1)	▶	**AROR**	**2.5162E+01**	1.6296E+01	1.8093E+01
		Portfolio Risk	**3.3176E+06**	2.0895E+06	2.5298E+06
		VaR	**3.9490E+05**	2.6933E+05	1.8831E+05
		Transactions Count	**7.8000E+01**	7.0000E+01	6.8000E+01
		Solution Length	**2.0000E+00**	5.7000E+01	3.6000E+01
Portfolio Risk (MGA1)		**AROR**	**2.9173E+00**	2.8743E+00	2.8743E+00
	▶	**Portfolio Risk**	**0.0000E+00**	0.0000E+00	0.0000E+00
		VaR	**0.0000E+00**	0.0000E+00	0.0000E+00
		Transactions Count	**2.0000E+00**	2.0000E+00	2.0000E+00
		Solution Length	**1.0000E+00**	3.8000E+01	3.8000E+01
VaR (JBLU1)		**AROR**	**1.4984E+01**	1.2095E+01	1.3313E+01
		Portfolio Risk	**1.0970E+06**	1.0717e+06	1.0115E+06
	▶	**VaR**	**0.0000E+00**	0.0000E+00	0.0000E+00
		Transactions Count	**1.0200E+02**	6.8000E+01	7.6000E+01
		Solution Length	**3.8000E+01**	3.5000E+01	5.8000E+01
Transactions Count (LUV1)		**AROR**	**-5.1663E-02**	-9.9755E-02	-8.5752E-02
		Portfolio Risk	**4.6062E+05**	4.7781E+05	4.3228E+05
		VaR	**5.8006E+04**	6.5248E+04	6.0277E+04
	▶	**Transactions Count**	**6.3400E+02**	6.1600E+02	6.1800E+02
		Solution Length	**5.1000E+01**	1.0000E+01	4.4000E+01
Solution Length (ATRO1)		**AROR**	**1.8919E+01**	3.6600E+00	1.2000E+01
		Portfolio Risk	**2.5523E+06**	1.0528E+06	1.5216E+06
		VaR	**2.9994E+05**	9.1406E+04	1.4375E+05
		Transactions Count	**7.8000E+01**	6.6000E+01	6.0000E+01
	▶	**Solution Length**	**1.0000E+00**	2.0000E+00	2.6000E+01

5 EXPERIMENTAL SETUP AND RESULTS

In order to evaluate the performance of the three algorithms, we first performed hyperparameter optimization on them using IRace [33]. The IRace procedure was provided with a budget of 300 evaluations and set to return a single configuration for the parameters of the algorithm being tuned. All algorithms have a tuned population size and number of iterations. The additional parameters tuned for GA are mutation probability, crossover probability and tournament size. For both PSO variants, the parameters also included neighborhood size, cognitive coefficient, social coefficient and velocity clamping technique. For PSO^{SP}, pruning frequency and pruning threshold are also considered for tuning. The final parameter values for the algorithms discovered by IRace can be seen in Table 1.

All three algorithms are then trained and tested using Trend Representative Testing, and utilizing the same training and testing datasets as described in [9]. For each algorithm, we hold one partition for testing, and use the other nine partitions for training, and go through the available strands to produce ten distinct training and testing datasets. Each algorithm is run against each dataset ten times to factor in the effects of stochasticity. This would result in 100 experiments being run per algorithm. Also, as in [9], all algorithms had access to a set of 63 technical indicators to compose individuals from. All the indicator parameters are initialized to random values, with those representing lengths of history constrained to be within 1 and 45, guaranteeing at least five buy, sell or hold signals to be generated with a typical US trading year of 252 days.

Table 2 shows the best performing solutions discovered per objective being optimized. For every objective we identify the test strand where the best value was achieved, followed by the best solutions discovered by each algorithm in that strand. In the case of more than one solution being non-dominated, we follow a lexicographical approach in order to determine the winning solution based on the following order: AROR, Portfolio Risk, VaR, Transactions Count and Solution Length. We can see from Table 2 that PSO^{SP} was the algorithm that returned the best value for all objectives. We cannot, however, claim that the solutions discovered by PSO^{SP} are Pareto-dominant when compared to the other algorithms with the exception of the case of Portfolio Risk with the MGA1 test strand. This is also an interesting solution as it shows the possibility of achieving returns (a positive AROR), using a singular technical indicator (Solution length), with no losses at all (zero Portfolio Risk and VaR) albeit with a low confidence (a low Transactions Count). With multiobjective optimization, the designer of a market timing strategy now has a choice to prioritize their objectives as they see fit. In between the best performing solutions reported per objective in Table 2 is a multitude of solutions representing various compromises between the optimized objectives from which the designer of the market timing strategy can choose a compromise that best

Table 3: Hypervolume results for each algorithm over the ten datasets. The minimum, mean and max values are obtained by running each algorithm ten times on each dataset. Best mean results are highlighted in bold.

				PSO^{SP}			GA			PSO	
#	**Trend**	**Test Strand**	**Min**	**Mean**	**Max**	**Min**	**Mean**	**Max**	**Min**	**Mean**	**Max**
0	↑	IAG1	0.0000E+00	2.0721E+12	2.0721E+13	0.0000E+00	3.1374E+13	1.5687E+14	7.5013E+14	**1.0792E+15**	1.2775E+15
	↔	MGA4	7.5915E+15	**8.6809E+15**	9.4711E+15	5.2677E+15	7.4559E+15	8.3000E+15	4.1511E+15	5.1502E+15	5.6475E+15
	↓	IAG2	3.7947E+15	**4.2788E+15**	4.7965E+15	1.0276E+15	1.9300E+15	2.2820E+15	7.4536E+14	1.1568E+15	1.3039E+15
1	↑	BSX1	3.0031E+13	9.9077E+14	1.4692E+15	1.0081E+13	1.8054E+15	3.0269E+15	1.9595E+15	**2.2582E+15**	3.1386E+15
	↔	LUV1	0.0000E+00	2.4946E+13	2.4946E+14	0.0000E+00	1.8595E+14	4.6486E+14	0.0000E+00	**1.9513E+14**	4.4559E+14
	↓	KFY1	8.8162E+15	**9.4288E+15**	9.5916E+15	2.4293E+15	5.1271E+15	5.7078E+15	2.1772E+15	2.7837E+15	2.9949E+15
2	↑	EXC1	8.6084E+15	**9.3953E+15**	1.0252E+16	2.9952E+15	4.4676E+15	5.5422E+15	1.6888E+15	2.3700E+15	2.9973E+15
	↔	LUV2	9.8965E+15	**1.0258E+16**	1.0538E+16	5.4433E+15	6.7389E+15	7.7529E+15	3.1646E+15	3.7100E+15	4.4641E+15
	↓	KFY2	1.0929E+16	**1.2033E+16**	1.2391E+16	4.8750E+15	6.8397E+15	8.0802E+15	4.4121E+15	4.9381E+15	5.5544E+15
3	↑	AVNW1	2.8912E+13	4.0510E+14	7.1632E+14	2.2933E+15	**2.5120E+15**	2.6314E+15	1.9179E+15	2.0253E+15	2.1264E+15
	↔	PUK1	0.0000E+00	2.8179E+14	3.9323E+14	7.8278E+14	**6.6284E+15**	1.1050E+16	2.7412E+15	4.1300E+15	5.2560E+15
	↓	LUV3	7.1381E+15	**7.8621E+15**	8.2233E+15	1.5829E+15	2.8642E+15	3.6973E+15	1.8231E+15	2.4935E+15	2.6640E+15
4	↑	KFY3	5.9623E+15	**7.2981E+15**	8.0712E+15	1.9308E+15	3.5694E+15	4.7005E+15	2.0784E+15	2.6586E+15	2.8978E+15
	↔	EXC2	2.4848E+16	**2.6919E+16**	2.8877E+16	1.2030E+16	1.5234E+16	1.6574E+16	1.0438E+16	1.1680E+16	1.2540E+16
	↓	LUV4	1.0567E+16	**1.1611E+16**	1.2368E+16	5.5361E+14	3.6987E+15	6.7676E+15	1.6656E+15	2.3846E+15	2.7778E+15
5	↑	EXC3	8.8289E+15	**9.7774E+15**	1.0698E+16	4.6275E+15	6.1406E+15	6.9482E+15	3.5527E+15	4.1234E+15	4.3049E+15
	↔	PUK2	3.2566E+15	4.6627E+15	6.3107E+15	7.3783E+15	**1.3314E+16**	1.7129E+16	9.4559E+15	1.0565E+16	1.1319E+16
	↓	MGA1	2.1257E+16	**2.5686E+16**	2.6937E+16	3.5782E+15	4.6755E+15	5.2255E+15	1.8923E+15	3.2313E+15	3.9332E+15
6	↑	ED1	6.4009E+15	**6.8579E+15**	7.1890E+15	2.3051E+15	3.3556E+15	3.9156E+15	1.4573E+15	2.0448E+15	2.3906E+15
	↔	EXC4	3.0371E+16	**3.2708E+16**	3.3832E+16	9.9233E+15	1.3657E+16	1.5948E+16	9.3957E+15	1.0610E+16	1.1396E+16
	↓	PUK3	4.8598E+15	**5.7253E+15**	6.1008E+15	5.9158E+14	1.6142E+15	2.1978E+15	5.3062E+14	1.1570E+15	1.4815E+15
7	↑	BSX2	1.6475E+16	**1.7237E+16**	1.7544E+16	6.8725E+15	9.8105E+15	1.1704E+16	4.6320E+15	5.7433E+15	6.2048E+15
	↔	ED2	1.2493E+16	**1.2777E+16**	1.3624E+16	7.4084E+15	8.8559E+15	9.1930E+15	4.2265E+15	4.8188E+15	5.2441E+15
	↓	JBLU1	7.4437E+15	**8.4528E+15**	8.8882E+15	5.2696E+15	7.3373E+15	8.3671E+15	2.7088E+15	3.4958E+15	4.1904E+15
8	↑	MGA2	0.0000E+00	1.8604E+14	2.4925E+14	4.6791E+14	**2.3694E+15**	3.5465E+15	9.8324E+14	1.5455E+15	2.5241E+15
	↔	MGA3	1.9961E+15	5.0144E+15	7.3013E+15	4.6563E+15	**7.0662E+15**	1.1523E+16	4.6429E+15	5.6118E+15	6.3861E+15
	↓	ATRO1	1.1505E+16	**1.3654E+16**	1.5298E+16	7.6311E+14	6.8626E+15	8.5909E+15	5.8541E+15	6.7805E+15	7.1962E+15
9	↑	AVNW2	7.9967E+15	**8.8008E+15**	9.5728E+15	3.6122E+15	5.2354E+15	6.1116E+15	4.1268E+15	4.5777E+15	4.8163E+15
	↔	EXC5	9.9980E+15	2.1932E+16	2.6254E+16	7.0217E+15	**2.8121E+16**	3.6686E+16	2.0235E+16	2.2887E+16	2.5011E+16
	↓	AVNW3	9.0977E+15	**1.0261E+16**	1.1159E+16	3.5222E+15	6.1280E+15	7.6136E+15	3.3619E+15	3.8850E+15	4.2041E+15

fits their needs. This is only possible by using a dominance based multiobjective optimization approach to market timing.

When comparing AROR to the values obtained in [9], we can see that the best performing solution discovered in AROR is higher than the maximum value discovered in [9]: 25.16 compared to 16.08 respectively. A comparison of the AROR values can be seen in Table 4. The maximum values are used in the comparison since AROR is an objective that is maximized. It is interesting to note that all the maximum values obtained via the multiobjective versions of the algorithms are higher than their single objective counterparts with the exception of two cases.

In order to evaluate the dominance aspect of the algorithms, we compare the hypervolume covered by the Pareto fronts produced by them. Hypervolume is a measure of the combined dominated space across all objectives being optimized covered by the solutions in the Pareto set returned by the algorithm. We use the implementation in [34, 35] to measure hypervolume. The hypervolume measurements can be seen in Table 3. The hypervolume measurements used a reference point of (0, 1000000, 1000000, 0) for AROR, Portfolio Risk, VaR and Transactions Count respectively. A value of zero was used for AROR as this represents a break-even situation where the strategy neither lost capital or made any profits. The value of 1000000 is used for Portfolio Risk and VaR as this is the same value used for initial capital when performing backtesting, and would represent risking the full volume of the capital allocated for investment. A value of zero is used for the number of transactions as this is technically the least amount of transactions that the market timing strategy can generate: none. As for solution length, we find the largest value reported in the Pareto set being tested and use that as the reference point. When considering hypervolume, larger values are better, and these are highlighted in bold based on the mean values obtained per test strand and algorithm. We can see that PSO^{SP} has outperformed the other two algorithms when it came to downtrends. As for uptrends, we have five wins for PSO^{SP}, two for GA and two for PSO. In sideways movements, we have five wins for PSO^{SP}, three for GA and a single win for PSO. In order to see if any of the algorithms has a statistically significant advantage when it came to dominance, we used the Friedman non-parametric test with the Holm correction on the mean hypervolumes attained by all three algorithms divided by trend type [36]. The results of the Friedman test can be seen in Table 5. We can see that PSO^{SP} had a statistically significant advantage over GA and PSO in downtrends; no statistically significant differences when it came to uptrends and sideways movements.

Although hypervolume begins to provide some idea of the performance of the algorithms on the five objectives, it does not provide information regarding the spread of the solutions on the Pareto front. Having solutions that are evenly spread across the Pareto

Table 4: A comparison of the AROR values between the single objective and multiobjective optimization algorithms. The maximum values obtained in the experiments are used since AROR is an objective that is maximized. The higher values per algorithm pair (single objective versus multiobjective) is highlighted in bold.

			Algorithm					
#		Test Strand	PSO^S	PSO^{SP}	GA	GA (MO)	PSO	PSO (MO)
0	↑	IAG1	-3.42	**0.04**	-1.39	**0.06**	-3.04	**0.97**
	↔	MGA4	1.70	**2.08**	1.60	**2.14**	2.09	**2.30**
	↓	IAG2	2.17	**2.90**	2.16	**2.92**	2.17	**2.95**
1	↑	BSX1	-0.11	**0.63**	-0.13	**0.73**	-0.02	**1.27**
	↔	LUV1	-0.08	**0.07**	-0.01	**0.03**	-0.04	**0.07**
	↓	KFY1	2.17	**3.39**	2.67	**2.81**	2.66	**3.02**
2	↑	EXC1	2.92	**3.44**	2.90	**3.08**	2.80	**3.05**
	↔	LUV2	2.46	**2.79**	2.64	**2.92**	2.62	**2.88**
	↓	KFY2	2.85	**6.52**	2.05	**3.80**	3.61	**3.89**
3	↑	AVNW1	1.22	**0.77**	1.29	**1.83**	1.26	**2.08**
	↔	PUK1	**0.38**	0.06	0.00	**0.76**	0.00	**0.64**
	↓	LUV3	6.12	**6.16**	4.63	**6.00**	4.70	**7.13**
4	↑	KFY3	2.94	**3.08**	2.67	**2.91**	2.80	**3.30**
	↔	EXC2	1.62	**5.14**	1.55	**2.08**	1.60	**2.49**
	↓	LUV4	2.95	**4.89**	2.96	**3.75**	2.80	**3.74**
5	↑	EXC3	**2.39**	2.35	2.14	**2.31**	2.38	**2.55**
	↔	PUK2	0.76	**2.04**	0.51	**2.40**	1.07	**3.16**
	↓	MGA1	3.80	**7.85**	3.15	**3.37**	2.80	**4.35**
6	↑	ED1	2.32	**2.58**	1.98	**2.75**	2.44	**2.84**
	↔	EXC4	3.96	**14.27**	3.72	**5.88**	3.47	**6.91**
	↓	PUK3	3.66	**4.53**	3.66	**3.95**	2.80	**3.64**
7	↑	BSX2	3.76	**4.33**	4.03	**4.15**	3.32	**4.33**
	↔	ED2	2.67	**2.82**	2.70	**2.71**	2.63	**2.75**
	↓	JBLU1	13.09	**14.98**	12.06	**12.63**	11.95	**13.43**
8	↑	MGA2	0.00	**0.09**	-3.39	**0.44**	-0.04	**0.68**
	↔	MGA3	1.34	**1.79**	0.51	**0.79**	0.48	**1.60**
	↓	ATRO1	11.51	**25.16**	16.08	**16.30**	11.31	**18.09**
9	↑	AVNW2	5.67	**12.02**	5.65	**8.60**	3.99	**10.88**
	↔	EXC5	0.56	**3.54**	0.96	**3.85**	0.87	**3.42**
	↓	AVNW3	2.20	**4.79**	2.14	**4.74**	2.10	**3.33**

Table 5: Average rankings of each algorithm according to the Friedman non-parametric test with the Holm post-hoc test over the mean hypervolume. Statistical significance at 0.05 percentage level is observed in downtrends where PSO^{SP} outperforms both PSO and GA.

Trend	Algorithm	Ranking	p-value	Holm
Uptrend	GA (control)	1.8	–	–
	PSO^{SP}	1.8	0.9999	0.05
	PSO	2.4	0.1797	0.025
Sideways	GA (control)	1.5	–	–
	PSO^{SP}	2.0	0.2636	0.05
	PSO	2.5	0.0253	0.025
Downtrend	PSO^{SP} (control)	1.0	–	–
	GA	**2.0**	**0.0253**	**0.05**
	PSO	**3.0**	**7.7442E-6**	**0.025**

front will provide a larger diversity of solutions for the user to select from. This measure of diversity can perhaps be included in the evaluation of the multiobjective algorithms presented here in a future work.

6 CONCLUSION

In this paper, we extend the work in [9] and proposed a GA and two PSO variants to tackle market timing as a multiobjective optimization problem. The number of financial metrics is increased from one to five, and we consider potential for loss in Portfolio Risk and VaR, confidence in solution in Transaction Count and Solution alongside the original profit metric AROR. We adopted a dominance based approach to multiobjective optimization, and modify the original algorithms with the addition of a global archive to maintain a Pareto set of solutions. We also adapted the selection procedure in GA, and both and neighborhood best selection in the two PSO variants. Finally, modified the pruning procedure of the original PSO^S algorithm to not remove components of a solution permanently, and instead ignore their impact when their weight falls below a predefined threshold. This allows for the possibility of reactivation later in the algorithm's run via a velocity update. we name this variant PSO^{SP}. Results show that PSO^{SP} has a statistically significant advantage over GA and PSO when it came to downtrend movements based on mean hypervolume. When comparing the maximum AROR value observed here versus in [9], we can note that the value observed here is considerably higher: 25.16 to 16.08 respectively.

We propose the following opportunities of future research. Although hypervolume provides some insight into the performance of the multiobjective algorithms, it does not provide information regarding the diversity of solutions in the returned Pareto sets. Considering this metric when evaluating Pareto fronts could be beneficial, as fronts with high diversity present the user with a better variety of compromises between the optimized objectives to select from. We also observed from the results that one algorithm performed particularly well under a particular trend type. This work could be enhanced by increasing the number of multiobjective algorithms tackling market timing and use meta-learning to discover metaheuristics that have a statistically significant advantage for each trend type. This ensemble of metaheuristics could then be used to build more adept market timing strategies. A final proposition for improvement would be in regards to Pareto front management. We currently do not have any constraints about which regions of the Pareto front the solutions come from. This means that the Pareto front may have its population around certain regions, limiting the choice provided to the end user in terms of compromise between the different objectives optimized. By ensuring that the Pareto front is more uniformly populated across its various objectives, we ensure that the end user has a better choice of compromises between the different objectives. This applies to both the GA and PSO algorithms, and can be done by incorporating some of the aspects of front management from algorithms such as NSGA-II [37] and NSGA-III [38].

REFERENCES

[1] S. Patterson, *Dark Pools: The Rise of A.I. Trading Machines and the Looming Threat to Wall Street.* Random House Business Books, 2013. [Online]. Available: https://books.google.co.uk/books?id=qaxaTcS9gBQC

[2] P. J. Kaufman, *Trading Systems and Methods*, 5th ed. John Wiley & Sons, Inc, 2013.

[3] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43. [Online]. Available: http://ieeexplore.ieee.org/document/494215/

[4] A. Soler-Dominguez, A. A. Juan, and R. Kizys, "A Survey on Financial Applications of Metaheuristics," *ACM Computing Surveys*, vol. 50, no. 1, pp. 1–23, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=3054133{%}5Cnhttp://dl.acm.org/citation.cfm?doid=3058791.3054133

[5] Y. Hu, K. Liu, X. Zhang, L. Su, E. W. T. Ngai, and M. Liu, "Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review," *Applied Soft Computing*, vol. 36, pp. 534–551, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2015.07.008

[6] F. Allen and R. Karjalainen, "Using genetic algorithms to find technical trading rules," *Journal of Financial Economics*, vol. 51, no. 2, pp. 245–271, 1999.

[7] A. C. Briza and P. C. Naval Jr., "Stock trading system based on the multi-objective particle swarm optimization of technical indicators on end-of-day market data," *Applied Soft Computing*, vol. 11, no. 1, pp. 1191–1201, 2011. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1568494610000621

[8] I. Mohamed and F. E. Otero, "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.* Springer International Publishing, 2018, pp. 435–436.

[9] I. Mohamed. and F. E. B. Otero., "Using population-based metaheuristics and trend representative testing to compose strategies for market timing," in *Proceedings of the 11th International Joint Conference on Computational Intelligence - Volume 1: ECTA, (IJCCI 2019)*, INSTICC. SciTePress, 2019, pp. 59–69.

[10] D. de la Fuente, A. Garrido, J. Laviada, and A. Gómez, "Genetic algorithms to optimise the time to make stock market investment," in *Genetic and Evolutionary Computation Conference*, 2006, pp. 1857–1858. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1143997.1144298

[11] H. Subramanian, S. Ramamoorthy, P. Stone, and B. Kuipers, "Designing Safe, Profitable Automated Stock Trading Agents Using Evolutionary Algorithms," in *Genetic and Evolutionary Computation Conference*, vol. 2, 2006, p. 1777. [Online]. Available: http://eprints.pascal-network.org/archive/00004834/

[12] M. Kampouridis and F. E. Otero, "Evolving trading strategies using directional changes," *Expert Systems with Applications*, vol. 73, pp. 145–160, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2016.12.032

[13] Y. Kim, W. Ahn, K. J. Oh, and D. Enke, "An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms," *Applied Soft Computing*, vol. 55, pp. 127–140, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2017.02.006

[14] S. Chakravarty and P. K. Dash, "A PSO based integrated functional link net and interval type-2 fuzzy logic system for predicting stock market indices," *Applied Soft Computing*, vol. 12, no. 2, pp. 931–941, 2012.

[15] C. F. Liu, C. Y. Yeh, and S. J. Lee, "Application of type-2 neuro-fuzzy modeling in stock price prediction," *Applied Soft Computing*, vol. 12, no. 4, pp. 1348–1358, 2012.

[16] S.-M. Chen and P.-Y. Kao, "TAIEX forecasting based on fuzzy time series, particle swarm optimization techniques and support vector machines," *Information Sciences*, vol. 247, pp. 62–71, 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0020025513004283

[17] P. Ladyzynski and P. Grzegorzewski, "Particle swarm intelligence tunning of fuzzy geometric protoforms for price patterns recognition and stock trading," *Expert Systems with Applications*, vol. 40, no. 7, pp. 2391–2397, 2013.

[18] F. Wang, P. L. Yu, and D. W. Cheung, "Combining technical trading rules using particle swarm optimization," *Expert Systems with Applications*, vol. 41, no. 6, pp. 3016–3026, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2013.10.032

[19] A. Bera, D. Sychel, and B. Sacharski, "Improved Particle Swarm Optimization method for investment strategies parameters computing," *Journal of Theoretical and Applied Computer Science*, vol. 8, no. 4, pp. 45–55, 2014.

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

[21] A. Karathanasopoulos, C. Dunis, and S. Khalil, "Modelling, forecasting and trading with a new sliding window approach: the crack spread example," *Quantitative Finance*, vol. 7688, no. September, pp. 1–12, 2016. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/14697688.2016.1211796

[22] J. Moore and R. Chapman, "Application of particle swarm to multiobjective optimization," Department of Computer Science and Software Engineering, Auburn University, Tech. Rep., 1999.

[23] X. Hu and R. Eberhart, "Multiobjective Optimization using Dynamic Neighborhood Particle Swarm Optimization." in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1677–1681.

[24] R. E. X. Hu and Y. Shi, "Particle Swarm with Extended Memory for Multiobjective Optimization." in *Proceedings of the IEEE swarm Intelligence Symposium*, 2003, pp. 193–197.

[25] E. L. C.A. Coello Coello and A. Aguirre, "Mopso: A proposal for multiple objective particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1051–1056.

[26] Z. M. M. M. L. Zhang, C. Liu and Y. Liang, "Solving multi objective optimization problems using particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 4, 2003, pp. 2400–2405.

[27] S. Mostaghim and J. Tiech, "Strategies for finding local guides in multi-objective particle swarm optimization (mopso)," in *Proceedings of the IEEE Swarm Intelligence Symposium*, 2003, pp. 26–33.

[28] Y. Zhang and S. Huang, "Multiobjective optimization using distance-based particle swarm optimization," in *Proceedings of the International Conference on Computational Intelligence, Robotics and Autonomous Systems*, 2003.

[29] G. Armano and M. R. Farmani, "Multiobjective clustering analysis using particle swarm optimization," *Expert Systems with Applications*, vol. 55, pp. 184 – 193, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741630032X

[30] J. Meza, H. Espitia, C. Montenegro, E. Giménez, and R. González-Crespo, "Movpso: Vortex multi-objective particle swarm optimization," *Applied Soft Computing*, vol. 52, pp. 1042 – 1057, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494616304859

[31] A. Pan, L. Wang, W. Guo, and Q. Wu, "A diversity enhanced multiobjective particle swarm optimization," *Information Sciences*, vol. 436-437, pp. 441 – 465, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025518300549

[32] X. Liu, Z. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 587–602, Aug 2019.

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

[34] C. M. Fonseca, L. Paquete, and M. López-Ibáñez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006).* Piscataway, NJ: IEEE Press, Jul. 2006, pp. 1157–1163.

[35] N. Beume, C. M. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold, "On the complexity of computing the hypervolume indicator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1075–1082, 2009.

[36] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, May 2010. [Online]. Available: https://doi.org/10.1016/j.ins.2009.12.010

[37] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[38] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, Aug 2014.