Optimising Directional Changes trading strategies with different algorithms

Jurgen Palsma
School of Computing
University of Kent, Canterbury, U.K.
jurgen.palsma@gmail.com

Adesola Adegboye
School of Computing
University of Kent, Medway, U.K.
atna3@kent.ac.uk

***Abstract*—Directional Changes (DC), a novel approach for sampling market data, allows the extraction of trends in financial time series by converting series from a time based format to an event-driven format. This paradigm has been shown to give some predictability in financial prediction, and has been used to generate profitable trading strategies on the FOREX market. In the past, a genetic algorithm was used to optimise the parameters of DC-based trading strategy. The goal of this work is to explore whereas different machine learning algorithms can be used to improve the results on the aforementioned optimisation task. For this purpose, we explore two algorithms, namely Particle Swarm Optimization and Shuffled Frog Leaping Algorithm. After comparing the performance of these two algorithms on 36 different datasets from 4 different currency pairs, we find that they statistically improve the profitability of the DC-based trading strategies.**

***Index Terms*—Directional Changes, Genetic Algorithm, Particle Swarm Optimisation, Shuffled Frog Leaping Algorithm**

I. Introduction

A new challenge has emerged from the increase in volume and velocity of financial data in the field of computational finance. Although this eruption of data was first a great source of wealth and opportunity, it quickly became a problem due to its magnitude, luring traders into hectic trading strategies, making the market data noisy and chaotic and limiting its predictability.

As a result, an approach named Directional Changes (DC), has recently been gaining attention to extract trends in financial data by converting time based data series into event-based data series. This approach allows to only focus on 'significant' events, i.e., events of a certain magnitude which interests the trader.

A particular implementation of a DC-based trading paradigm, proposed by [1], has been shown to generate profitable and risk-averse trading strategies on the foreign exchange market. In fact, their work was shown to outperform traditional technical analysis based trading strategies. The authors achieved their results by optimising the recommendations of multiple DC-based trading strategies through a Genetic Algorithm (GA).

The goal of this work is to explore whether further improvements in the trading strategy's profitability can take place if different algorithms are used in the place of the GA. In this paper we will be exploring two different techniques, namely Particle Swarm Optimisation and the Shuffled Frog Leaping Algorithm.

The rest of this paper is organised as follows: we present background information on DC in section II, and explain in detail the strategy proposed by [1]. We will then present in section III our proposed alternative optimisation algorithms. We will lay out our experimental setup including our data, how we tuned our algorithms, and their final configurations in section IV, present and analyse our results in section IV, and finish by concluding on our work in section VI.

II. Background

In this section, we present the background knowledge necessary to understand the setting of our experiments: we first describe Directional Changes, and then how they have been used in [1] to generate trading strategies. Lastly, we present a review of the DC literature.

A. Directional Changes

The directional change (DC) approach is an alternative approach for summarising market price movements. A DC event is identified by a change in the price of a given financial instrument. This change is defined by a threshold value, which was in advance decided by the trader. Such an event can be either an upturn or a downturn event. After the confirmation of a DC event, an overshoot (OS) event follows. This OS event finishes once an opposite DC event takes place. The combination of a downturn event and a downward overshoot event represents a downward trend and, the combination of an upturn event and an upturn overshoot event represents an upturn trend. In other words, a downward trend is a period between a downturn event and the next upturn event and an upturn trend is a period between an upturn event and the next downturn event.

Figure 1 presents an example of how a physical-time price curve is transformed to an event-based system and dissected into DC and OS events. As we can observe in figure 1, two different thresholds are used, and each threshold generates a different event series. Thus, each threshold produces a unique series of events. The idea behind the different thresholds is that each trader might consider different thresholds (price percentage changes) as significant. A smaller threshold creates a

Algorithm 1 Pseudocode for generating directional changes events (source: [2]).

Require: Initialise variables (event is Upturn event, $p^h = p^l = p(t_0), \Delta x_{dc}(Fixed) \geq 0, t_0^{dc} = t_1^{dc} = t_0^{os} = t_1^{os} = t_0$)

if event is Upturn Event **then**
  if $p(t) \leq p^h \times (1 - \Delta x_{dc})$ **then**
    $event \leftarrow DownturnEvent$
    $p^l \leftarrow p(t)$
    $t_1^{dc} \leftarrow t$ // End time for a Downturn Event
    $t_0^{os} \leftarrow t+1$ // Start time for a Downward Overshoot Event
  else
    if $p^h < p(t)$ **then**
      $p^h \leftarrow p(t)$
      $t_0^{dc} \leftarrow t$ // Start time for Downturn Event
      $t_1^{os} \leftarrow t-1$ // End time for an Upward Overshoot Event
else
  if $p(t) \leq p^l \times (1 + \Delta x_{dc})$ **then**
    $event \leftarrow UpturnEvent$
    $p^h \leftarrow p(t)$
    $t_1^{dc} \leftarrow t$ // End time for a Upturn Event
    $t_0^{os} \leftarrow t+1$ // Start time for an Upward Overshoot Event
  else
    if $p^l > p(t)$ **then**
      $p^l \leftarrow p(t)$
      $t_0^{dc} \leftarrow t$ // Start time for Upturn Event
      $t_1^{os} \leftarrow t-1$ // End time for an Downward Overshoot Event

higher number of directional changes, while a higher threshold produces fewer directional changes.

Looking at the events generated by a threshold of $\theta = 0.01\%$ (events connected via solid and dashed lines), we can observe that any price change less than this threshold is not considered a trend. On the other hand, when the price changes above that threshold, then the market is divided accordingly, to uptrends and downtrends. DC events are in solid lines, and OS events are in dashed lines. For example, a downturn DC event starts at Point A and lasts until Point B, when the downturn OS events starts. The downturn OS lasts until Point C, when there is a reverse in the trend, and an uptrend starts, which lasts until Point D. From Point D to E we are in an upturn OS event, and so on.

As we mentioned, different thresholds generate different event series. Looking at $\theta = 0.018\%$ (events connected via dotted and dot-dashed lines), we can observe that the events generated are different: a downward trend starts from A and lasts until B$'$, and the downward OS is from Point B$'$ until C. Then, from Point C until Point E there is an upward DC trend, and from E to E$'$ there's an upward OS trend. Algorithm 1 presents the high-level pseudocode for generating directional changes events.

It is important to note here that the confirmation of a change of a trend can only be confirmed retrospectively, i.e. only after the price has changed by the pre-specified DC threshold value θ. For example, under $\theta = 0.01\%$ we can only confirm that we are in a upward trend from Point D onwards. Point D is thus called a *confirmation point*. Before Point D, the directional change had not been confirmed (i.e. the market price had not changed by the pre-specified threshold value), thus a trader summarising the data by the DC paradigm would continue believing we are in a downward trend, which started from Point A. Similarly, a trader using $\theta = 0.01\%$ would continue considering being in a upward trend from Point D until the price has reversed by $\theta = 0.01\%$, which only takes place at the next confirmation point, i.e., Point F. So what becomes important here is to be able to anticipate the change of the trend as early as possible, i.e. before Points C and E have been reached. In addition, since different thresholds generate different event series, we hypothesise that the combined information from these series would lead to profitable trading strategies.

The advantage of this new way of summarising data is that it provides traders with new perspectives to price movements, and allows them to focus on key points where an important event took place, blurring out other price details which could be considered irrelevant or even noise. Furthermore, DC have enabled researchers to discover new regularities in markets, which cannot be captured by the interval-based summaries [3]. Therefore, these new regularities give rise to new opportunities for traders, and also open a whole new area for research.

One of the most interesting regularities that was discovered in [3] was the observation that a DC of threshold θ is on average followed by an OS event of the same threshold θ. At the same time, it was observed that if on average a DC takes t amount of physical time to complete, the OS event will take an amount of $2t$. This observation is summarised in Figure 2, and *was only made under DC-based price summaries*, and not under phsycical-time summaries. Furthermore, this astonishing observation was made on all of the 13 different currency exchange rates that the authors of [3] experimented with. This thus lead us to further hypothesise that such statistical properties could lead to profitable strategies, if appropriately exploited, mainly because such properties are not well-known to traders yet. Therefore, the DC area is a rich research area that could potentially lead to significant discoveries.

B. Evolving trading strategies with directional changes

The trading strategies focused on in our work and for which we want to improve optimisation are proposed in [1]. They propose a multiple DC threshold trading strategy, where each threshold would advise one weighed trading action and a decision would be suggested from a "voting" session between each weighed trading action, with the argument that multiple thresholds capture different event magnitudes and allow for improved predictions.

The idea behind this trading strategy of using multiple DC thresholds is that different thresholds provide different perspective of the data under observation. Smaller threshold

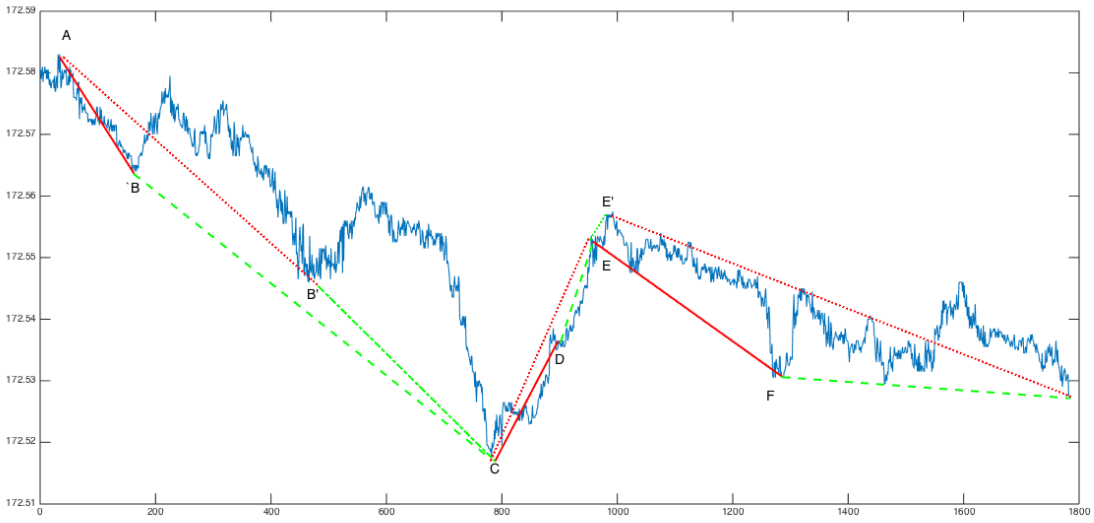

Fig. 1. Directional changes for tick data for the GBP/JPY currency pair. The solid and dashed lines denote a set of events defined by a threshold $\theta = 0.01\%$, while the dotted and dot-dashed lines refer to events defined by a threshold $\theta = 0.018\%$. The solid and the dotted lines indicate the DC events, and the dashed and dot-dashed indicate the OS events. Under $\theta = 0.01\%$, the data is summarised as follows: Point A $\mapsto$ B (Downward directional change), Point B $\mapsto$ C (Downward overshoot event), Point C $\mapsto$ D (Upward directional change), Point D $\mapsto$ E (Upward overshoot event), Point E $\mapsto$ F (Downward directional change). Under $\theta = 0.018\%$, the data is summarised as follows: Point A $\mapsto$ B' (Downward directional change), Point B' $\mapsto$ C (Downward overshoot event), Point C $\mapsto$ E (Upward directional change), Point E $\mapsto$ E' (Upward overshoot event).

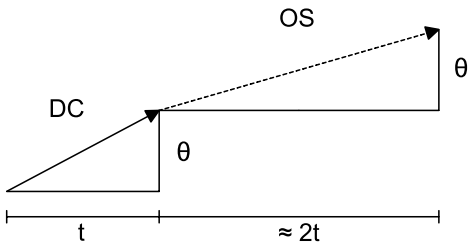

Fig. 2. An example of a scaling law presented in [3], which shows that (1) a DC event (solid line) of threshold θ is followed by an OS event (dotted line) of also threshold θ, and (2) the OS event lasts about the double amount of time that it took for the DC event to take place.

sizes are used in detecting more events, and this allows traders to react more promptly to price movement. However, this might not be an optimal strategy because of the transaction costs associated with trading actions. On the other hand, with a larger threshold, fewer events are detected, providing opportunities of taking action when price change is more sizeable. Selecting a threshold that is too large can lead to inaction or opportunity loss. Thus, this trading strategy combined the use of different threshold values in an attempt to take advantage of the different characteristics of smaller and larger thresholds. At any point in time, each threshold could be recommending an action: buy, hold, or sell. As there are multiple thresholds, each threshold might recommend a different action. In order to decide which action to take, a weight system was used, where each DC threshold was assigned a weight. Then a majority vote would take place, where the recommendation with the highest sum of weights, wins. For example, if we have 5 DC thresholds, and the first three recommend buy and the remaining two sell, we would then sum up the weight values of the first three thresholds, and compare it to the sum of the weights of the last two thresholds; depending on which sum is higher, we would follow the respective action. The weights are not fixed, but are evolved by the genetic algorithm. More information about this trading strategy can be found in [1].

C. Review of DC literature

[4] carried out the first work that used a DC dataset as an alternative to a physical time dataset. [5] extended the work of [3] and introduced four additional scaling laws, these new scaling laws were successfully applied to investigate the impact of different strategies on trading activities in high-frequency FOREX market. The catalogue of DC indicators was further extended by [6], who introduced 5 additional laws to the ones already discovered in [5], [3]. [7] reported the Scale of Market Quakes (SMQ). SMQ is a way of sizing the impact of economic or political development and other major breaking news on price movement within the FOREX market. Their goal was to set the foundation for creating a metric that can be used to measure price change vis-a-vis major world events.

[8] developed agents that model traders' behaviour in FOREX market. Their work focused on establishing stylised facts regarding how traders react and adapt to changes in FOREX market. Their agents used strategies known as ZI-DC0

developed by combining DC approach with trend following and contrary trading technical indicators. [9] proposed a new trading strategy called ZI-DC1 as an improvement to the study in [8]. Comparison results between ZI-DC0 and ZI-DC1 showed that ZI-DC1 was more profitable. [10] introduced an automated trading strategy (DCT2) that can perceive changes in market conditions and adapt dynamically to remain profitable.

[11] developed a neuro-fuzzy logic based trading strategy that captures volatility using DCs within a pre-specified threshold. The system predicted the future price of an asset based on the current price and the immediate past three consecutive observations in the market. Their model outclassed the physical-time scale trading strategies they compared with, in terms of profitable returns. [12] transformed a DC forecasting task into a classification problem. Their goal was to establish the predictive power in directional changes approach. To do this end, they created three new directional changes indicators inspired from technical indicators which they used for forecasting price value at OS extreme point. [13] was the first work to use a genetic programming algorithm to generate DC-based trading strategies. Results showed that the new algorithm had the potential to outperform its competitors.

[1] (presented in Section II-B), and subsequently [14], [15] used DC for trading purposes and optimised the multiple DC thresholds' recommendations using a genetic algorithm. The GA allowed the authors to yield promising results when experimenting on tick and 10-minute data from 5 currency pairs in the FOREX market in a time period of 10 months from August 2013 to May 2014. According to their experiments, the multi-threshold strategy outperformed traditional benchmarking techniques such as buy and hold and proposed a genetic programming FOREX trading strategy.

These promising results motivated us to apply different optimisation techniques to the multi-threshold strategy. As the GA was never tested against other algorithms for the given optimisation task, we are interested in investigating whether other algorithms can improve the profitability of the DC-based trading strategies. As mentioned earlier, we will be using particle swarm optimization algorithm and continuous shuffled frog leaping algorithm, which have been shown by [16] to outperform genetic algorithms in some optimization problems.

III. Methodology

In this section, we will present the two algorithms we have used to optimise the multiple DC threshold strategy developed by [1], presented in the previous section. We first present our approach to the particle swarm optimisation (PSO) algorithm, in subsection III-A and then present the shuffled frog leaping algorithm in subsection III-B. Finally, we present the fitness function which allows us to evaluate the performance of a candidate solution to the optimisation problem.

A. Particle swarm optimization

Particle swarm optimisation (PSO) is a nature inspired metaheuristic search algorithm that was introduced by [17]. It shares some similarities with a GA as it optimises and transforms a set of candidate solutions. However, instead of mutating and evolving the individuals as in a GA, the PSO algorithm optimises individuals based on a concept of velocity which guides the search at each iteration in the search space. The individuals move through the search space until their convergence, i.e. when their change in velocity reaches a certain change threshold.

An individual (candidate solution) in the PSO is called a particle and is composed of a vector of n attributes, similar to a GA individual, which represents the set of parameters for our search problem. In our particular search problem, an individual is represented as a set of possible trading parameters for the trading strategy presented in subsection II-B. These parameters are among other the amount to trade at each action, the value for each DC threshold, and the weight accorded to each threshold. To optimise these parameters, a particle "searches" through the search space with a certain velocity, which represents the search direction of each parameter towards an optimal fitness value.

Each parameter in a particle has its own velocity v_{ij}, which is defined, for the j-th parameter of the i-th particle, at an iteration $t + 1$, in equation 1. The computation of a particle's velocity is a weighted sum of three variables, its inertia, its memory, and its neighbourhood. These variables are represented in equation 1 with:

- The inertia influence $v_{ij}(t)$ representing the parameter's previous position influence on the search direction, weighed by the inertia weight w_s.
- The memory influence $h_{ij} - x_{ij}(t)$ representing the parameter's previous best (historical) position h_{ij} relative to the parameter's previous position $x_{ij}(t)$, weighed by the historical weight w_h
- The neighbourhood influence $g_{ij} - x_{ij}(t)$ representing the parameter's neighbour's best position g_{ij} relative to the parameter's previous position $x_{ij}(t)$, weighed by the neighbourhood weight w_g. The particle's neighbourhood is a subdivision of the swarm which allows to focus either on exploitation or exploration depending on the neighbourhood' size.

$$v_{ij}(t+1) = w_s.v_{ij}(t)+w_h.(h_{ij}-x_{ij}(t))+w_g.(g_{ij}-x_{ij}(t)) \quad (1)$$

We also propose some enhancements on top of the canonical aspect of the PSO to maximise our algorithm's performance. First, we introduce an early stopping criterion to minimise computation costs by minimising fitness evaluations. If k particles reach a common best fitness value, we assume convergence at a global optimum and stop the algorithm. We also submit our particles to a technique named clamping which prevents particles from exponentially increasing in velocity by defining a maximum velocity a particle can have. Our final enhancement allows us to deal with outlying individuals with excessively low fitness (which violate fitness constraints), by "resetting" their velocity by setting their inertia and memory

weight to 0 for one iteration. This allows the particle to ignore its low-fitness inducing memory weight and makes it more prone to explore the parameter space in direction of its neighbours.

At each iteration of the algorithm, until we reach convergence, we "move" each particle by applying its velocity to its coordinates (parameter values) in the search space, represented by the equation 2.

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij} \tag{2}$$

For the sake of clarity, we present the PSO algorithm in a high-level pseudocode in algorithm 2.

Algorithm 2 Particle Swarm Optimization

With: *fitness function:* $f()$

Initialise population: $P \leftarrow$ *with* N *random particles*
for *each particle* x_i *in* P **do** *calculate* $f(x_i)$
while *convergence not reached* **do**
    for *each particle* x_i *in* P **do**
        for *each attribute* x_{ij} *in* x_i **do**
            $x_{ij} = x_{ij} + v_{ij}$
            $v_{ij} = w_s.v_{ij} + w_h.(h_{ij} - x_{ij}) + w_g.(g_{ij} - x_{ij})$
    for *each particle* x_i *in* P **do** *calculate*$f(x_i)$

B. Continuous shuffled frog leaping algorithm

The continuous shuffled frog algorithm (CSFLA) is based on the shuffled frog leaping algorithm introduced by [18], improved by [19], which adapted the original shuffled frog leaping algorithm to be applied to continuous search spaces.

The individuals in the CSFLA, named frogs, are represented as a vector of n attributes which are in our case the parameters of the multi-threshold trading strategy. In our particular search problem, an individual is represented as a set of possible trading parameters for the trading strategy presented in subsection II-B. These parameters are among other the amount to trade at each action, the value for each DC threshold, and the weight accorded to each threshold.

In order to optimise each set of parameters contained in an individual, the CSFLA starts by initialising a population of N randomly generated frogs. Then, the algorithm iteratively evolves the population by evolving niches of candidates to minimise fitness evaluations (which is crucial in our case, as our fitness evaluations are costly by nature, being trading simulations over a large dataset). Thus, the algorithm repeats the following process for a total of G_m iterations (generations):

First, we perform an initial fitness evaluation and order the frogs in descending fitness value order. We then divide our population into M memeplexes, which are sub groups of individuals which allow the exploration of local optima (niches). The division into memeplexes is done with the following process: we assign the first individual in our list of sorted frogs to the first memeplex, the second individual to the second memeplex, the M-th individual to the M-th memeplex, the $M+1$-th individual to the first memeplex, and so on, until each frog is assigned to a memeplex.

To evolve our individuals into niches, we divide our memeplex into "submemplexes". The division is done according to a probabilistic distribution which assures that individuals with higher fitness have a higher chance of being selected. We present the probability of selecting the i-th individual x_i into a submemeplex in equation 3:

$$p(x_i) = \frac{2(N+1-i)}{N(N+1)} \tag{3}$$

This division into submemeplexes allows to focus on exploitation of niches and ignoring the lower scoring individuals to minimise fitness function calculations and encourage convergence towards an optimum in the memeplex. To reduce computational costs, we evolve, in each submemplex, only the individual with the lowest fitness x_w. The lowest individual evolves according to the position in the search space of best performing frog x_b in the submemeplex, with the following 3-step procedure:

- First, we attempt a learning procedure where x_w "learns" from x_b with equation 4, with r a random number between 0 and 1.

$$x_w(t+1) = x_w(t) + r(x_b(t) - x_w(t)) \tag{4}$$

  Considering our fitness function f and if $f(x_w(t+1)) > f(x_w(t))$ then we assume the worst frog has positively learned and we assign $x_w(t)$ to its new value $x_w(t+1)$.
- If $f(x_w(t+1)) < f(x_w(t))$, the worst frog has not improved in terms of fitness, and thus we try to learn from the best individual x_s in the entire swarm with equation 5:

$$x_w(t+1) = x_w(t) + r(x_s(t) - x_w(t)) \tag{5}$$

  As in the previous step, considering the fitness function f and if $f(x_w(t+1)) > f(x_w(t))$ then we assume the worst frog has positively learned and we assign $x_w(t)$ to its new value $x_w(t+1)$.
- Otherwise, we assign $x_w(t+1)$ to a randomly generated position, with a and b some arbitrarily set maximum and minimum boundaries, in equation 6.

$$x_w(t+1) = a + r(b-a) \tag{6}$$

We repeat this evolution process in our sub-memeplex for a certain number G_s of sub-generations, and then return our individuals to our memeplex, and repeat our process for a certain number of G_m generations. Once the generations have past, the algorithm yields a list of candidates sorted by fitness. In our case, we pick the highest performing individual as our candidate solution for the optimisation problem. A high-level pseudocode representation of the CSFLA is presented in algorithm 3.

Algorithm 3 Continuous Shuffled Frog Leaping Algorithm

With: *fitness function:* $f()$, $g_m = 0$, $g_s = 0$

Initialise population: $P \leftarrow$ *with* N *random particles*
for *each frog* x_i *in* P **do** *calculate* $f(x_i)$
Sort P *by decreasing fitness value*
while $g_m < G_m$ **do**
    Generate M *memeplexes*
    for *each memeplex* **do**
        Generate n *sub-memeplexes*
        for *each attribute* x_{ij} *in* x_i **do**
            while $g_s < G_s$ **do**
                Move the worst frog of the memeplex
                According to equations $4, 5, 6$
                $g_s = g_s + 1$
            Insert all frogs back in P
    for *each particle* x_i *in* P **do** *calculate*$f(x_i)$
    $g_m = g_m + 1$

C. Fitness function

Several different metrics have been used in the literature as fitness function in algorithmic trading. Some examples are: wealth, profit, return, Sharpe ratio, information ratio. In this paper, we set the fitness of an individual (proposed set of strategy parameters)equal to the total return minus the maximum drawdown of the outcome of trading using the individual over a certain dataset. We use this fitness function to evaluate our performance for consistency with the methods used in [1]. Equation 7 present the function:

$$\begin{aligned} ff &= Return - \alpha \times MDD \\ MDD &= \frac{P_{trough} - P_{peak}}{P_{peak}} \end{aligned} \quad , \tag{7}$$

where $Return$ is the return of the investment, MDD is the maximum drawdown, and α is a tuning parameter. Maximum drawdown is defined as the maximum cumulative loss since commencing trading with the system. It is used to penalise volatile trading strategies in terms of return. Its value is given as the percentage of $\frac{P_{trough}-P_{peak}}{P_{peak}}$, where P_{trough} the trough value of the price, and P_{peak} is the peak value of the price. Lastly, the tuning parameter α is used to define how much risk-averse the strategy is. The more risk-averse in terms of wishing to avoid a catastrophic loss, the higher the value of α.

IV. Experimental setup

In this section, we present how we set up our experiment: we describe the data we used, how we tuned our algorithms for our optimisation problem, and lay out the different algorithm configurations we used for testing.

A. Data

The data used in the experiment consists of 12 months of 10-minute interval data from the FOREX market, from June 2013 to May 2014, from 4 different currency pairs: Euro / US Dollar , Euro / British pound, British pound / Swiss franc, and British Pound / US dollar. The first 3 months of data were used to tune the algorithms, and the remaining months were used to evaluate the results of our algorithms on the optimisation problem. Our algorithms were tested in the following way: every month is split into its own dataset, with the first 70% of the data being the training set, and the remaining 30% being the testing set.

B. Algorithm tuning and parameters

We applied the same tuning methodology to both the PSO and the CSFLA: we used a horse-race parameter tuning process, by testing over 50 different PSO combinations and over 40 CSFLA combinations on our 3 months of test data. The candidate parameter sets that came out of the horse race tuning with the highest returns were then selected as the parameter sets we used on our test data. The selection took place by using the non-parametric Friedman statistical test, thus the selected configurations statistically outperformed the alternatives.

The final PSO configuration is presented in table I, the final CSFLA configuration in table II, and the GA parameters, set as equal to the ones set by [1] in their experiments, are presented in table III. Multiple DC thresholds were used in generating our DC series. These thresholds are the same as the ones select in [1] and the values are: 0.01%, 0.013%, 0.015%, 0.018%, and 0.02%. Tuning parameter α presented in equation 7 was set to 0.2. This value was determined using I/F-Race [20].

TABLE I
PSO CONFIGURATION

Parameter	Value
Swarm size	50
Maximum velocity	150
Inertia weight	0.85
Memory weight	0.45
Neighbourhood weight	0.05
Stopping criterion k	5
Minimum velocity threshold	0.00005
Maximum iterations	7

TABLE II
CSFLA CONFIGURATION

Parameter	Value
Number of frogs	200
Frogs per memeplex	20
Number of memeplexes	20
Maximum generations	3
Maximum sub-memeplex generations	40

In addition, we also benchmark our results against a genetic programming financial forecasting algorithm [21], [22], [23], [24], [25]. This algorithm combines different technical analysis indicators together, in order to form predictions. The reason to benchmark against technical analysis is because it is one of the most common trading techniques, and this allows us to measure our work's performance not only on a specific

TABLE III
GA CONFIGURATION

Parameter	Value
Population size	1000
Number of generations	35
Tournament size	4
X-over probability	0.90
Mutation probability	0.0025

optimisation problem but also as a competitive trading strategy alternative.

V. RESULTS AND ANALYSIS

In this section, we present the results of our experiments running the algorithms presented in section III with the setup presented in section IV.

Table IV presents the monthly mean results for each currency pair, over GA, CSFLA, PSO, and GP. We denote in bold the best mean return per row. As we can observe, GA was best 8 times, CSFLA 9 times, PSO 7 times, and GP 12 times. In terms of overall average performance, as we can see from Table V, GP was the best algorithm for EUR/GBP, GA the best for GBP/CHF, while PSO came first for both GBP/USD and EUR/USD. More importantly, PSO had the highest mean return of 0.010381.

These results were further supported by the non-parametric Friedman test, presented in Table VI. As we can observe, the PSO ranks first, followed by the CSFLA and GP that have the same ranking, and the GA being ranked last. Subsequent analysis on the Holm post-hoc test showed that the above results were not significant at the 5% level. However, this should not alarm us, because the fact remains that the PSO has ranked first in the majority of the currency pairs tested. This suggests that it is a more robust algorithm and is expected to produce better results when new data is used.

It is important to note that our original goal of improving the GA's performance as an optimiser for DC-based trading strategies has been achieved, since both the PSO and CSFLA have yielded higher average returns and also shown better ranking through the Friedman test.

VI. CONCLUSION

In conclusion, we have applied two algorithms, the particle swarm optimisation algorithm and the continuous shuffled frog leaping algorithm, to optimise parameters of DC-based trading strategies. We have shown that both algorithms offer improvements over a genetic algorithm, which was previously used as the optimiser for the given task. The PSO and CSFLA also returned comparable results to a technical analysis trading strategy, which was optimised by genetic programming.

These results are very encouraging because they demonstrate the potential of directional changes, as they can yield positive returns, and can outperform some technical analysis based strategies. Moreover, the results demonstrate that the performance of the DC-based trading strategies can be further improved by looking into how the strategies' parameters are optimised. In addition to their capacity to yield positive returns on historical data, we emphasise the potential of our strategies being not well know to traders yet, and coupled with atypical optimisation algorithms which increase the strategy's returns. This allows them to be innovative and potentially competitive in real-world trading. It would be interesting to apply the tested algorithms to other trading strategies to investigate their performance in different environments, and see whether they are best fitted for DC-based environments or other trading strategies. We could also try to improve the optimisation problem results by combining our tested algorithms or test them

TABLE IV
MEAN MONTHLY RESULTS FOR THE 4 CURRENCY PAIRS.

	GA	CSFLA	PSO	GP
EUR_GBP				
September	-0.0000148	-0.001801	**0.00017145**	0.00010843
October	0.0000762	-0.000637	**0.00059506**	0.00058537
November	**0.0000173**	-1.31E-04	-8.33E-05	-0.0000335
December	0.0000496	-0.000866	**1.14E-04**	0.0000068
January	**-6.89E-05**	-0.000096	-0.00130394	-0.00033099
February	-3.07E-05	9.10E-05	-0.0000485	**0.00068661**
March	0.0000594	-0.001033	0.0001673	**0.00017825**
April	-0.0000023	-0.000118	**0.00018897**	0.00012726
May	**2.01E-05**	0.000001	-0.00177085	-0.00146786
GBP_CHF				
September	0.0000399	0.000123	0.00063854	**0.0006417**
October	-0.0004399	**4.60E-03**	0.0000861	0.00017781
November	0.0001084	0.000204	**2.92E-04**	0.0000923
December	0.0004526	**0.000483**	-0.0003138	-0.00045079
January	**0.0001703**	-8.15E-04	8.21E-05	0.0000191
February	**1.51E-04**	0.000091	-0.00030491	-0.00026982
March	-0.000133	**0.000273**	-0.00010231	-0.00018848
April	-0.0000516	**0.001324**	0.000432	0.00070451
May	0.0000117	-0.003114	0.00016799	**0.00016947**
GBP_USD				
September	0.0000527	0.002506	0.00730391	**0.00833202**
October	**0.000196**	-0.001178	-0.00027488	-0.0001799
November	-0.0001843	**0.002049**	4.76E-04	-0.0000774
December	5.91E-05	-0.000061	0.00013813	**0.00015082**
January	**0.0000044**	-0.005245	-0.00051436	-0.00070105
February	-0.0000413	**0.000117**	-0.00177948	-0.00165598
March	-0.0000485	**0.002314**	-0.00026263	0.00020933
April	0.0000046	0.000877	0.00204839	**0.00226083**
May	0.0001046	-0.001608	0.00014476	**0.00035787**
EUR_USD				
September	0.0000081	0.000208	0.0001952	**0.00064548**
October	0.0006714	-0.023286	0.00039047	**0.00128115**
November	0.0000269	**1.20E-04**	-7.49E-05	-0.0000733
December	0.0000381	**0.000628**	-0.00036009	-0.00039472
January	-0.000506	-0.000433	**-0.00029313**	-0.00045077
February	-0.0000667	0.000332	0.00049563	**0.0006901**
March	-0.000012	0.000103	**0.00044259**	0.00017442
April	-0.0000877	0.000944	0.00053225	**0.00223177**
May	**-4.50E-06**	-0.000044	-0.0023122	-0.00317595

TABLE V
MEAN RETURNS OF EACH ALGORITHM ACROSS THE FOUR CURRENCY PAIRS ON 9 MONTHS OF DATA

	GA	CSFLA	PSO	GP
EUR/GBP	-0.00459	-0.00197	-0.00014	**0.00001**
GBP/CHF	**0.003170**	0.000978	0.000896	0.00004
GBP/USD	-0.000229	0.007279	**0.008697**	0.00001
EUR/USD	-0.021428	-0.000984	**0.000928**	-0.00002
Mean	-0.023077	0.005303	**0.010381**	0.00005

TABLE VI
FRIEDMAN RANKING FOR THE MEAN RETURNS OF THE PSO, CSFLA, GA, AND GP

Algorithm	Average Rank
PSO (c)	1.75
CSFLA	2.5
GP	2.5
GA	3.25

further by comparing their performance to other metaheuristic algorithms on the same problem. Lastly, we could also test our results on different datasets such as different currencies, different markets, or different time-periods.

REFERENCES

[1] M. Kampouridis and F. E. Otero, "Evolving trading strategies using directional changes," *Expert Systems with Applications*, vol. 73, pp. 145–160, 05 2017.

[2] M. Aloud, E. Tsang, R. Olsen, and A. Dupuis, "A directional change events approach for studying financial time series," *Economics*, vol. 36, p. (online), 2012.

[3] J. Glattfelder, A. Dupuis, and R. Olsen, "Patterns in high-frequency fx data: discovery of 12 empirical scaling laws," *Quantitative Finance*, vol. 11, no. 4, pp. 599–614, 2011.

[4] D. M. Guillaume, M. M. Dacorogna, R. R. Davé, U. A. Müller, R. B. Olsen, and O. V. Pictet, "From the bird's eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets," *Finance and stochastics*, vol. 1, no. 2, pp. 95–129, 1997.

[5] M. Aloud and M. Fasli, "The impact of strategies on the stylized facts in the fx market," 2013.

[6] M. E. Aloud, "Time series analysis indicators under directional changes: The case of saudi stock market," *International Journal of Economics and Financial Issues*, vol. 6, no. 1, 2016.

[7] T. Bisig, A. Dupuis, V. Impagliazzo, and R. Olsen, "The scale of market quakes," *Quantitative Finance*, vol. 12, no. 4, pp. 501–508, 2012.

[8] M. Aloud, E. Tsang, and R. Olsen, "Modeling the fx market tradersâĂŹ behavior: An agent-based approach," *Banking, Finance, and Accounting: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*, p. 350, 2014.

[9] M. Aloud, "Directional-change event trading strategy: Profit-maximizing learning strategy," *The Seventh International Conference on Advanced Cognitive Technology and Applications*, 2016.

[10] M. E. Aloud, "Profitability of directional change based trading strategies: The case of saudi stock market," *International Journal of Economics and Financial Issues*, vol. 6, no. 1, 2016.

[11] A. Kablan and W. L. Ng, "Intraday high-frequency fx trading with adaptive neuro-fuzzy inference systems," *International Journal of Financial Markets and Derivatives*, vol. 2, no. 1-2, pp. 68–87, 2011.

[12] A. Bakhach, E. P. Tsang, and W. L. Ng, "Forecasting directional changes in financial markets," 2015.

[13] J. Gypteau, F. E. Otero, and M. Kampouridis, "Generating directional change based trading strategies with genetic programming," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2015, pp. 267–278.

[14] A. Adegboye, M. Kampouridis, and C. G. Johnson, "Regression genetic programming for estimating trend end in foreign exchange market," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017.

[15] M. Kampouridis, A. Adegboye, and C. Johnson, "Evolving directional changes trading strategies with a new event-based indicator," in *Simulated Evolution and Learning*, Y. Shi, K. C. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, and Y. Jin, Eds. Springer International Publishing, 2017.

[16] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, vol. 19, no. 1, pp. 43 – 53, 2005.

[17] G. Venter and J. Sobieszczanski-Sobieski, "Particle swarm optimization," *AIAA Journal*, vol. 41, no. 8, pp. 1583–1589, 08 2003.

[18] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *Journal of Water Resources Planning and Management*, vol. 129, no. 3, pp. 210–225, 2003.

[19] Z. Zhen, D. Wang, and Y. Liu, "Improved shuffled frog leaping algorithm for continuous optimization problem," in *2009 IEEE Congress on Evolutionary Computation*, 05 2009, pp. 2992–2995.

[20] M. LÃşpez-IbÃąÃśez, J. Dubois-Lacoste, L. P. CÃąceres, M. Birattari, and T. StÃijtzle, "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, vol. 3, pp. 43 – 58, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214716015300270

[21] M. Kampouridis and E. Tsang, "Eddie for investment opportunities forecasting: Extending the search space of the gp," in *IEEE Congress on Evolutionary Computation*, 07 2010, pp. 1–8.

[22] ——, "Investment opportunities forecasting: Extending the grammar of a gp-based tool," *International Journal of Computational Intelligence Systems*, vol. 5, no. 3, pp. 530–541, 2012.

[23] M. Kampouridis, A. Alsheddy, and E. Tsang, "On the investigation of hyper-heuristics on a financial forecasting problem," *Annals of Mathematics and Artificial Intelligence*, vol. 68 (4), pp. 225–246, 2013.

[24] F. E. B. Otero and M. Kampouridis, *A Comparative Study on the Use of Classification Algorithms in Financial Forecasting*. Springer Berlin Heidelberg, 2014, pp. 276–287.

[25] M. Kampouridis and F. E. Otero, "Heuristic procedures for improving the predictability of a genetic programming financial forecasting algorithm," *Soft Computing*, pp. 1–16, 2015.