Expert Systems With Applications 73 (2017) 145–160

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Evolving trading strategies using directional changes

Michael Kampouridis*, Fernando E.B. Otero

School of Computing, University of Kent, UK

ARTICLE INFO

Article history:
Received 10 October 2016
Revised 2 December 2016
Accepted 23 December 2016
Available online 27 December 2016

Keywords:
Directional changes
Financial forecasting
Algorithmic trading
Genetic algorithm

ABSTRACT

The majority of forecasting methods use a physical time scale for studying price fluctuations of financial markets, making the flow of physical time discontinuous. Therefore, using a physical time scale may expose companies to risks, due to ignorance of some significant activities. In this paper, an alternative and original approach is explored to capture important activities in the market. The main idea is to use an event-based time scale based on a new way of summarising data, called Directional Changes. Combined with a genetic algorithm, the proposed approach aims to find a trading strategy that maximises profitability in foreign exchange markets. In order to evaluate its efficiency and robustness, we run rigorous experiments on 255 datasets from six different currency pairs, consisting of intra-day data from the foreign exchange spot market. The results from these experiments indicate that our proposed approach is able to generate new and profitable trading strategies, significantly outperforming other traditional types of trading strategies, such as technical analysis and buy and hold.

1. Introduction

The global financial system, recently rocked by the financial crisis, is open 24 hours a day, 7 days a week and can be defined as a complex network of interacting agents (e.g., corporations, retail traders). With an average daily turnover of 3–4 trillion USD (International Monetary Fund, 2009) and price changes nearly every second, its activity varies at different times of a day and reacts on the announcement of political or economic news.

The majority of traditional methods to observe such price fluctuations in financial time series are based on physical time change. For example, what researchers and practitioners tend to do is to use snapshots of the market, taken at fixed intervals; they first decide how often to sample the data, and then they take snapshots at the chosen frequency. Therefore, these snapshots create an interval-based summary—e.g. daily closing prices or minute-by-ummaries. However, important price movements (and thus potential profit) might be lost due to the creation of these artificial price summaries. For example, if we are using daily closing price summaries we would not be able to observe the 6 May 2010 Flash Crash, which was a United States trillion-dollar stock market crash that lasted for approximately 36 minutes.[1]

Directional Changes (DC) is based on the idea that an event-based system can capture significant points in price movements that the traditional physical time methods cannot. Hence, instead of looking the market from an interval-based perspective, DC record the key events in the market (e.g., changes in the stock price by a pre-specified percentage) and summarise the data based on these events, moving away from a physical-time view to an event-based-time view. Under this new paradigm, a threshold θ is defined, usually expressed by a percentage of the price. The market is then fragmented and summarised into upward and downward trends. Different thresholds produce different price summaries. Thus, the directional changes paradigm focuses on the size of price change, while time is the varying factor; whereas in the physical-time paradigm, time was fixed (e.g. daily closing prices). This new concept provides traders with new perspectives to price movements, and allows them to focus on those key points than an important event took place, blurring out other price details which could be considered irrelevant, or even noise.

The first works to use the concept of directional changes were proposed in Olsen et al. (1997) and Glattfelder, Dupuis, and Olsen (2011). In these works, new empirical scaling laws in foreign exchange data series were discovered. These scaling laws aimed to establish mathematical relationships among price moves, duration and frequency. Then, directional changes and the scaling laws from the above works were used to develop new trading models in Dupuis and Olsen (2012). However, those models were not used for any financial forecasting purposes and were only used to derive statistics from potential trading. Furthermore, Aloud, Tsang, Olsen, and Dupuis (2012) demonstrated the effectiveness of

* Corresponding author.
E-mail addresses: M.Kampouridis@kent.ac.uk (M. Kampouridis), F.E.B.Otero@kent.ac.uk (F.E.B. Otero).

[1] http://blogs.wsj.com/marketbeat/2010/05/11/nasdaq-heres-our-timeline-of-the-flash-crash/ Last access: 12 September 2016.

http://dx.doi.org/10.1016/j.eswa.2016.12.032
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

directional changes in capturing periodic market activities. In addition, Gypteau, Otero, and Kampouridis (2015) presented an approach to forecasting the daily closing price of financial markets by combing directional changes and genetic programming. Lastly, Tsang, Tao, Serguieva, and Ma (2016) introduced new trading indicators for profiling markets under directional changes. As we can observe, the majority of the above works have focused on theoretical aspects of directional changes—e.g. establishing mathematical relationships and developing new indicators. Only Dupuis and Olsen (2012) and Gypteau et al. (2015) attempted to generate trading strategies based on the DC concept. However, Dupuis and Olsen (2012) did not take advantage of the combined knowledge that can exist by using multiple DC thresholds to generate different event-based series; in addition, Gypteau et al. (2015) only tested their approach on 4 datasets on daily closing prices.

In this paper our aim is to offer a more complete analysis on the directional changes paradigm from a financial forecasting perspective. We run extensive tests on intraday data from six currency pairs from the foreign exchange (FX) market: yearly *tick data* from GBP/JPY, and yearly *10 minute interval data* from EUR/GBP, EUR/USD, EUR/JPY, GBP/CHF, and GBP/USD. In total, *we run tests on 255 different datasets*. In terms of DC, we present two novel types of trading strategies: a single-threshold DC strategy, and a multi-threshold DC strategy. The former uses a single threshold to generate event-based series. The multi-threshold strategy combines different thresholds, where each threshold generates a different event-based series; then information from each series is aggregated to form a more informed trading strategy. In addition, we use a genetic algorithm (GA), which is a bio-inspired algorithm mimicking an evolutionary process, to optimise the parameters of the multi-threshold strategy. Such evolutionary algorithms have extensively been used in financial forecasting problems and have shown to be extremely effective (Evans, Pappas, & Xhafa, 2013; Kampouridis & Otero, 2015; Kwon & Moon, 2007; Mani, 1996). The GA-generated trading strategies are then compared against the single-threshold and the multi-threshold DC strategies. We test the GA-generated trading strategies with other financial benchmarks, such as *buy and hold* and *technical analysis* strategies. Overall, our goals in this work can be summarised as follows: (i) demonstrate that the paradigm of DC returns profitable strategies, (ii) provide evidence that the strategies optimised by the GA are more profitable than using standard DC strategies, and (iii) demonstrate that our GA generated strategies outperform classical physical-time based strategies, namely technical analysis and buy and hold.

Lastly, it should be acknowledged that directional changes has similarities to the concept of the zigzag indicator (Azzini, da Costa Pereira, & Tettamanzi, 2010), which also focuses on an event-based approach, and to the concept of perceptually important point (PIP) identification (Chen & Chen, 2016), which preserves the salient points in a time series to reduce the number of data points. However, as we've mentioned above, the recent research on the DC field has led to the development on many new concepts, such as the scaling laws and new trading indicators that only exist under DC price summaries. Thus, in order to take advantage of all these new developments, we are focusing on the DC paradigm.

The rest of this paper is organised as follows: Section 2 presents background information in the fields of financial forecasting, directional changes, and genetic algorithms. Then, Section 3 presents the proposed DC-derived trading strategies, and Section 4 discusses how we used the genetic algorithm to optimise the parameters of these strategies. In addition, Section 5 presents the experimental setup, and Section 6 presents and discusses the results. Finally, Section 7 concludes the paper and discusses directions for future work.

2. Background

Financial forecasting is a vital area in computational finance (Tsang & Martinez-Jaramillo, 2004). The end goal of financial forecasting is deriving a trading strategy, which makes a recommendation whether to buy, hold or sell. There are numerous works that attempt to return profitable trading strategies—several examples can be found in Binner, Kendall, and Chen (2004); Chen (2002); Hu et al. (2015); Jaisinghani (2016).

There are several different strategies for the purposes of trading in a financial market. A very common investment strategy, albeit a passive one, is *buy and hold*, and commonly acts as benchmark for trading algorithms. The principle of this strategy is based on the view that in the long run financial markets give a good rate of return to investors. Thus, in this strategy an investor buys an asset and holds it for a long time, without being concerned about short-term price movements. Then at the end of a given period, s/he sells the stock and potentially makes profit based on the price difference.

In contrary to buy and hold, there is also the belief that it is profitable to take advantage of short-term price movements, as long as one can anticipate/predict them. Technical analysis is such a technique, and is discussed in Section 2.1. Then, we present background information on directional changes, a new way of summarising financial data that can lead to new types of trading strategies. This takes place in Section 2.2. Finally, Section 2.3 gives an overview of genetic algorithms, which is the technique used in this paper for optimising the use of directional changes parameters.

2.1. Technical analysis

Technical analysis is a methodology for financial forecasting. This method assumes that patterns exist in historical price data and that these patterns will repeat themselves. Consequently, it is worth identifying these patterns, so that we can exploit them in the future and make profit. Several works exist that are using technical analysis—recent examples can be found in Cervelló-Royo, Guijarro, and Michniuk (2015); Chourmouziadis and Chatzoglou (2016).

As part of technical analysis, several indicators are used. These technical analysis indicators are formulas that measure different aspects of a given financial dataset, such as trend, volatility and momentum. Below in Eqs. (1)–(6) we present 6 popular indicators that can be found in the literature (Allen & Karjalainen, 1999; Austin, Bates, Dempster, Leemans, & Williams, 2004; Martinez-Jaramillo, 2007). Given a price time series $[P(t), t \geq 0]$, and a period of length L, these indicators are defined as follows:

Moving Average (MA)

$$\text{MA(L,t)} = \frac{P(t) - \frac{1}{L}\sum_{i=1}^{L} P(t-i)}{\frac{1}{L}\sum_{i=1}^{L} P(t-i)} \tag{1}$$

MA allows traders to observe any changes in the trend of the prices of a stock. Typically, when a short-term *MA* (e.g., 12 days) goes above a long-term *MA* (e.g., 60 days), this indicates upward momentum. On the other hand, when a short-term *MA* goes below a long-term one, this indicates downward momentum.

Trade Break Out (TBR)

$$\text{TBR(L,t)} = \frac{P(t) - max\{P(t-1), \ldots, P(t-L)\}}{max\{P(t-1), \ldots, P(t-L)\}} \tag{2}$$

In order to understand this indicator better, we first need to explain two terms: *support* and *resistance*. Support is the point where the price stops going down any further, whereas resistance is the point where the price does not go up any further. Technical analysis suggests that downward price trends tend to reverse at support points, whereas upward trends tend to reverse at resistance points. However, when these points are breached (breakout), perhaps because of some new information regarding the market, it is likely that the price will continue in the same direction. Hence, traders tend to observe these breakouts and when a stock goes above its point of resistance, they buy; when on the other hand the stock price goes below its point of support, traders sell.

Filter (FLR)

$$\text{FLR(L,t)} = \frac{P(t) - min\{P(t-1), \ldots, P(t-L)\}}{min\{P(t-1), \ldots, P(t-L)\}} \tag{3}$$

This indicator is used to indicate buy or sell actions, depending on whether the price movement goes in the opposite direction by a predefined percentage. For instance, if the price reverses from a downward trend and rises by a specific percentage from the low price that it was previously, then the trader would perform a 'buy' action.

Volatility (Vol)

$$\text{Vol(L,t)} = \frac{\sigma(P(t), \ldots, P(t-L+1))}{\frac{1}{L}\sum_{i=1}^{L} P(t-i)} \tag{4}$$

A period of an increasing volatility could indicate a reversal in the trend or strong downward trends. This would thus give an indication to a trader that s/he should be cautious. On the contrary, when there is a period of decreasing volatility, this indicates upward trends and traders should buy.

Momentum (Mom)

$$\text{Mom(L,t)} = P(t) - P(t-L) \tag{5}$$

When *Mom* is positive, this indicates an upward trend. If *Mom* starts decreasing, this could be an indication that there is going to be a reverse in the previously upwards trend, and hence the traders should be cautious. Finally, when *Mom* is negative, this indicates a downwards trend.

Momentum Moving Average (MomMA)

$$\text{MomMA(L,t)} = \frac{1}{L}\sum_{i=1}^{L} Mom(L, t-i) \tag{6}$$

Finally, from *Mom* we can also calculate its *MA*, as shown in the above equation, which allows us to obtain summaries of the *Momentum* movements.

While the above indicators can offer valuable information, normally a trader would use many of these indicators together, thus combine their recommendations. It is very common in the literature to use machine learning algorithms to combine technical information indicators (Chiang, Enke, Wu, & Wang, 2016; Kim & Enke, 2016). In this work, we use EDDIE (Kampouridis & Tsang, 2010; 2012) as a baseline algorithm. EDDIE is a genetic programming (GP) (Koza, 1992) financial forecasting algorithm, which combines the different technical analysis indicators together, in order to form predictions. The advantage of combining technical analysis indicators is that their combined knowledge can lead to better predictions. We should also mention that EDDIE has been used over the years for different types of financial problems, such as stock price movement prediction (Kampouridis & Otero, 2015), arbitrage opportunities detection (Tsang, Markose, & Er, 2005), and market crash detection (Garcia-Almanza, Alexandrova-Kabadjova, & Martinez-Jaramillo, 2013). EDDIE has thus extensively and effectively utilised technical analysis for its predictions, and for this reason we have chosen to use it as a benchmark of an algorithm using technical analysis.

As EDDIE is a GP algorithm, its trading strategies are represented as trees. A sample tree of EDDIE is presented in Fig. 1. As we can see, if the *20 days MA* is less than or equal to *6.4*, then the user is advised to buy; otherwise, the user is advised to consult another tree, which is located in the third branch ("else-branch") of the tree. This tree checks if the 50 days *MomMA* is greater than 5.57; if it is, it advises to not-buy, otherwise to buy. Of course this is simply a sample tree; so additional/different technical analysis indicators could be used in other trees.

In summary, what we have presented so far—namely buy and hold, technical analysis and EDDIE—all use information derived from data that is based on physical-time. As we have explained, in this paper we propose using event-based price summaries for creating the trading strategies, based on the concept of directional changes.

2.2. 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.

Fig. 2 presents an example of how a physical-time price curve is transformed to the so-called *intrinsic time* (Glattfelder et al., 2011) and dissected into DC and OS events. As we can observe, 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 higher number of directional changes, while a higher thresholds 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. So an 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

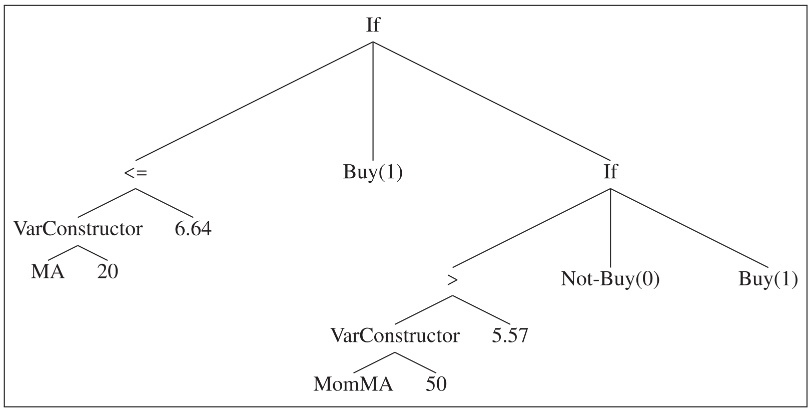

Fig. 1. Sample tree generated by EDDIE representing a trading strategy using technical indicators.

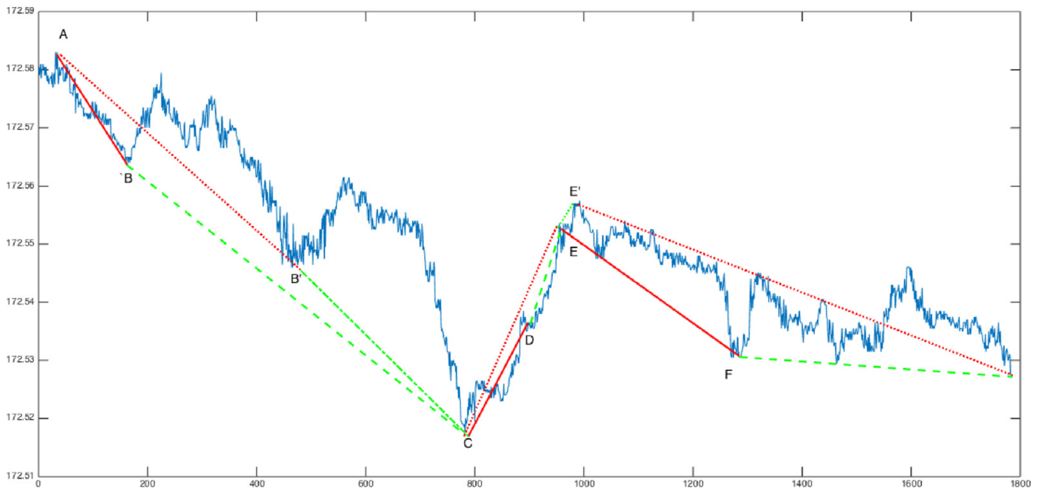

Fig. 2. Directional changes for tick data for the GBP/JPY currency pair. The solid and dashed lines denote a set of events defined by a threshold $\theta = 0.01\%$, while the dotted and dot-dashed lines refer to events defined by a threshold $\theta = 0.018\%$. The solid and the dotted lines indicate 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).

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 those key points that 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 (Glattfelder et al., 2011). 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 Glattfelder et al. (2011) 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 Fig. 3, 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 Glattfelder et al. (2011) experimented with. This thus leads 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.

In this work, we will present new trading strategies based on the concept of directional changes. As part of the implementation of this trading strategy we will be using the scaling law presented above; we have also introduced several parameters into the system, which we present in detail in Section 3.

Lastly, since a user/trader has to decide on which thresholds to use for generating DC events, a problem that arises is what are appropriate thresholds and how much weight we should give to the information provided by each threshold. We are thus faced with an optimisation problem, where one has to look into the space of different thresholds and focus on the most promising ones. One of the popular optimisation techniques is genetic algorithms, discussed next.

Algorithm 1 Pseudocode for generating directional changes events (source: Aloud et al. (2012)).

```
Require: Initialise variables (event is Upturn event, $p^h = p^l = p(t_0)$, $\Delta x_{dc}(Fixed) \geq 0$, $t_0^{dc} = t_1^{dc} = t_0^{os} = t_1^{os} = t_0$)
1:  if event is Upturn Event then
2:      if $p(t) \leq p^h \times (1 - \Delta x_{dc})$ then
3:          $event \leftarrow DownturnEvent$
4:          $p^l \leftarrow p(t)$
5:          $t_1^{dc} \leftarrow t$ // End time for a Downturn Event
6:          $t_0^{os} \leftarrow t + 1$ // Start time for a Downward Overshoot Event
7:      else
8:          if $p^h < p(t)$ then
9:              $p^h \leftarrow p(t)$
10:             $t_0^{dc} \leftarrow t$ // Start time for Downturn Event
11:             $t_1^{os} \leftarrow t - 1$ // End time for an Upward Overshoot Event
12:         end if
13:     end if
14: else
15:     if $p(t) \leq p^l \times (1 + \Delta x_{dc})$ then
16:         $event \leftarrow UpturnEvent$
17:         $p^h \leftarrow p(t)$
18:         $t_1^{dc} \leftarrow t$ // End time for a Upturn Event
19:         $t_0^{os} \leftarrow t + 1$ // Start time for an Upward Overshoot Event
20:     else
21:         if $p^l > p(t)$ then
22:             $p^l \leftarrow p(t)$
23:             $t_0^{dc} \leftarrow t$ // Start time for Upnturn Event
24:             $t_1^{os} \leftarrow t - 1$ // End time for an Downward Overshoot Event
25:         end if
26:     end if
27: end if
```

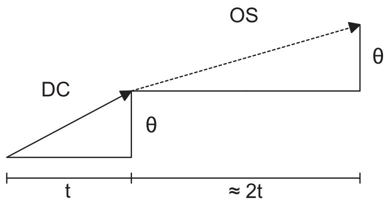

Fig. 3. An example of a scaling law presented in Glattfelder et al. (2011), which shows that (1) a DC event (solid line) of threshold θ is followed by an OS event (dotted line) of also threshold θ, and (2) the OS event lasts about the double amount of time that it took for the DC event to take place.

2.3. Genetic algorithms

Genetic algorithms (GAs) are bio-inspired algorithms that mimic an evolutionary process to find good solutions to optimisation problems (Godlberg, 1989; Michalewicz, 2002; Mitchell, 1996). GAs have several elements that allow them to perform a robust global search: (a) they work with a population of candidate solutions (individuals), rather than a single candidate solution; (b) individuals of the population are evaluated according to a fitness function, which measures the quality of the candidate solution represented by an individual—the higher their quality, the more likely that their *genetic material* will be carried forward to the next population; (c) the solution space is explored using genetic operators, which generate new offspring individuals from individuals of the current population using a stochastic selection procedure based on fitness.

Algorithm 2 presents a high-level pseudocode of a GA. The algorithm starts by creating a population of p candidate solutions,

Algorithm 2 High-level pseudocode of a genetic algorithm.

```
GA($p$, $Fitness$, $pc$, $pm$)
    $p$: population size
    $Fitness$: determines the quality of solutions
    $pc$: crossover rate
    $pm$: mutation rate
1: Initialise population: $P \leftarrow$ Generate $p$ individuals (candidate solutions) at random
2: Evaluate: for each $i$ in $P$, calculate Fitness(i)
3: while termination condition not satisfied do
4:     $P_g \leftarrow$ Create new population for generation $g$
       (a) Elitism: copy the $r$ best individuals from $P$ to $P_g$
       (b) Select: probabilistically select $(p - r)$ individuals of $P$ to add to $P_g$ and
           • Perform crossover between a pair of selected individuals according to $pc$
           • Perform mutation on a selected individual according to $pm$
5:     Update: $P \leftarrow P_g$
6:     Evaluate: for each $i$ in $P$, calculate Fitness(i)
7: end while
8: Return the individual with the highest fitness from $P$
```

where p is referred to as population size. These are evaluated by a fitness function in order to determine their performance in solving the problem at hand. On each iteration (*while* loop), a new population is then generated by probabilistically selecting the fitter individuals from the current population. Some of the selected individuals undergo crossover and mutation, which introduce modification to explore the search space; other are carried forward without modifications. The new population replaces the old and the new individuals are evaluated. This process is repeated until a maximum number of generations is reached or the (near-)optimal solution is found, acting as a termination condition. Through this evolutionary process, GAs perform a robust global search in the space of candidate solutions, less likely to get trapped into local minima.

Representation. Individuals in GAs are usually represented as a string of symbols, either binary or numeric values—the representation is dependant on the problem at hand. Fig. 4 shows an illustration of a real-valued representation. Each position in the string is referred to as a gene and it represents a variable to be optimised. At the start of a GA, the population is initialised with random individuals: each gene is initialised with a random value in the domain of the variable.

Genetic Operators. Genetic operators manipulate the genetic material of individuals (strings of symbols) to generate new offspring individuals, mimicking a mechanism of inheritance. For example, crossover create two new offspring solutions from two parent individuals by swapping portions of genetic material (or genes) from each parent. To illustrate, consider the *uniform crossover* operator in Fig. 4. This operator combines genes sampled uniformly from two parents. In addition to crossover, GAs usually employ a mutation operator, which produces a new offspring individual from a single parent. In *uniform mutation*, small changes are introduced to a parent individual by choosing and modifying each gene at random. Both uniform crossover and uniform mutation are controlled by two probabilities rates: the first one is used to decided whether the selected individual will undergo crossover/mutation (pc and pm in Algorithm 2, respectively) or not; the second rate is used to decide if a gene is swapped/mutated or not. Fig. 4 shows an illustration of both uniform crossover and uniform mutation operators.

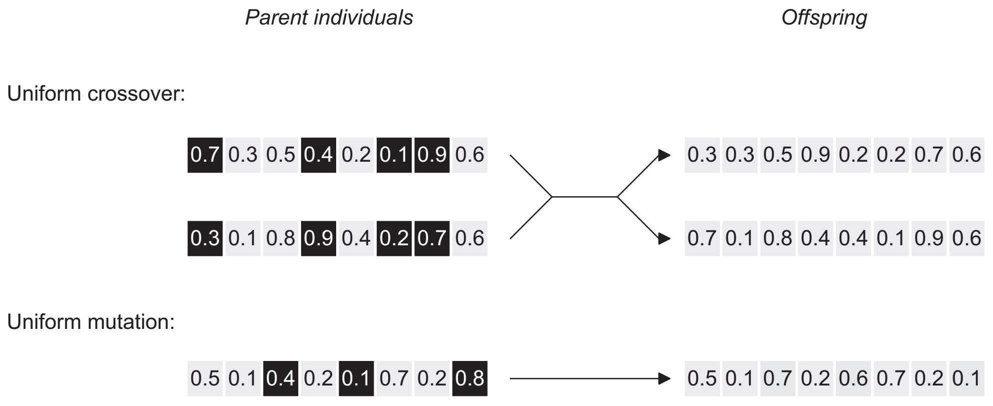

Fig. 4. Illustration of uniform crossover and uniform mutation operators. Individuals are represented as a string of real values. The dark positions (genes) in parent individuals are the ones that will be swapped/mutated to generate offspring individuals.

Elitism and Selection. Elitism is the process of allowing the best individuals (in terms of fitness) of the current population to be carried over to the next without modification. This guarantees that the solution fitness will not decrease from one generation to the next. The remaining individuals are subject to a probabilistic selection for inclusion in the next generation population. One of the most popular selection strategies is the *tournament selection*. In tournament selection, k individuals are selected at random, where k denotes the tournament size. The more fit individual of the selected subset is then selected for inclusion in the next population.

2.4. Summary

In this section we started by discussing two popular trading techniques, namely buy and hold and technical analysis. Both of these two methods will form our financial benchmarks during our experimental phase. In addition, we presented in detail the concept of directional changes, which our trading strategies are going to be based on. Lastly, we discussed what genetic algorithms are, as they are going to be our optimisation engine. In the next section, we present two new types of trading strategies, which are derived by directional changes.

3. DC-derived trading strategies

In this section, we will present how we can use the DC paradigm to derive two different types of trading strategies. The first one is going to be based on a single DC threshold, and is going to be presented in Section 3.1. The second strategy is going to be based on multiple DC thresholds and is going to be presented in Section 3.2.

3.1. Single-threshold DC trading strategy

As we have already discussed in Section 2.2, a physical-time price curve can be transformed to intrinsic time, where it's dissected into DC and OS events. When a DC event is confirmed (either upturn or downturn), the OS period starts. It is worth reiterating that we can only know the market has changed direction in hindsight; we only detect a DC event only after the DC confirmation point has been observed. After the DC confirmation has taken place, we are during an OS period, which lasts until there is a reverse in the direction, which is again only confirmed once we have reached the next confirmation point. Therefore, if one can act during the OS period and before the reverse of the trend, then this can lead to a profitable trading strategy. The idea is that if we are in an downtrend, we buy at the last point (ideally) of the OS event, which is the lowest observed value. When the trend then reverses and we are in an uptrend, we can then sell at a much higher price and make profit. The same principle applies for uptrend: sell as close as possible to the end of the OS event. To sum up, when there is a downtrend, we buy; when there is an uptrend, we sell.

In order to make the above clearer, let us refer back to Fig. 2. As we can observe, after the confirmation of the downward trend in Point B, a period of OS starts, which lasts until Point D, which is the next DC confirmation point, confirming that we are now in a upward trend. It is thus important to take a buy action during the OS event and ideally before the reverse of the trend, which as we can see takes place at Point C. The closer to the end of the OS event we can trade (i.e., the closer to Point C), the higher the profit margin a trader can make.

Thus, it is crucial to be able to anticipate the reverse in the trend and be able to act before that. In order to tackle this, we use the scaling laws we discussed earlier in Section 2.2. As explained, these laws states that when on average a DC event takes t amount of physical time to complete, the OS event takes an amount of $2t$. Because this is an approximation and it can rely on the underlying dataset, we wanted to have our own calculations for the datasets we are dealing with. Thus, what we did was to calculate the average time of each OS event for every period we would use as a training set. We hence created two variables, expressed as the average ratio of the OS event length over the DC event length. These two variables are r_u and r_d, where r_u is the average ratio of the upwards OS event, and r_d is the average ratio of the downwards OS event. Our calculations showed that these two variables had indeed values around 2 (ranging between 1.8–2.0), which confirms the scaling law findings. More importantly, this allowed us to have tailored values for r_u and r_d, for each training set we use.

After obtaining these ratios, we were able to anticipate the end of a trend (approximately) and as a result make trading decisions once an OS event had reached the average ratio of r_u or r_d. Of course, in reality things are not that simple. The r_u and r_d ratios are just average approximations, so many times the OS event might last longer or shorter than anticipated. In an attempt to address this issue, we have created two user-specified parameters, namely b_1 and b_2, which define a range of time within the OS period, where trading is allowed. For instance, if a trader expects the OS event to last for 2 hours, then we can define an *action* range of $[b_1, b_2] = [0.90, 1.0]$, which effectively means we are going to trade at the last 10% of the 2 hours duration, i.e. in the last 12 minutes. By introducing b_1 and b_2, we are essentially attempting to anticipate the approximation errors that might have been created during the calculation of r_u and r_d. Eq. (7) presents the formulas for these starting and ending for upward and downward OS periods:

$$\begin{aligned} t_0^U &= (t_1^{dc} - t_0^{dc}) \times r_u \times b_1 \\ t_1^U &= (t_1^{dc} - t_0^{dc}) \times r_u \times b_2 \\ t_0^D &= (t_1^{dc} - t_0^{dc}) \times r_d \times b_1 \\ t_1^D &= (t_1^{dc} - t_0^{dc}) \times r_d \times b_2, \end{aligned} \tag{7}$$

where t_0^U, t_1^U are the start and end times for upwards overshoot period, respectively, and t_0^D, t_1^D are the start and end times for downwards overshoot period, respectively. In addition, t_0^{dc} and t_1^{dc} are the start and the end times of the current DC event, after the confirmation of the event has taken place at time t_1^{dc}. Their difference $t_1^{dc} - t_0^{dc}$ returns the length of the current DC event. Also, r_u and r_d are the average ratios of the upwards and downwards OS period lengths, respectively, over the current DC period. Lastly, b_1 and b_2 are the two parameters defining the action range within the OS periods, as explained above.

Although b_1 and b_2 define a window for trading, a problem that exists with high-frequency data—particularly tick data—is that there can still be hundreds of points to trade, even if that trading window is very narrow. This could be problematic, because trading at multiple price levels will not return the highest profit. What is more effective is to sell (buy) at a price as expensive (cheap) as possible. To achieve this, we introduced another variable b_3, which prevents traders from doing expensive trades. To ensure this, we only allow the system to sell at the most expensive (peak) price P_{peak} and buy at the cheapest recorded price (trough) P_{trough}, or in prices in close range. This range is determined by the value of b_3. Therefore a trader would sell when the price is equal to $P_{peak} \times b_3$, or buy when the price is equal to $P_{trough} \times (1 - b_3)$. Essentially, b_3 is a value within the range of [0, 1] and defines the range of prices close to P_{peak} and P_{tough} that the system will perform an action.

Furthermore, there is a user-specified parameter Q, which controls the trading quantity. Lastly, it should be mentioned that our system allows short selling. However, in order to avoid excess short selling, which can lead to significant losses, we have introduced a stop loss mechanism that is called *short selling allowance*. This allowance is a percentage of our budget and allows short selling activities up to this pre-specified percentage. This percentage is decided during parameter tuning.

3.2. Multi-threshold DC trading strategy

This strategy builds on the previous one, as it still uses Eq. (7) and the b_3 variable. But instead of only using a single threshold, it combines information by multiple thresholds. As we discussed in Section 2.2, a DC event is identified by a change in the price by a given threshold value. The use of different DC thresholds provides a different view of the data: smaller thresholds allow the detection of more events and, as a result, actions can be taken promptly; larger thresholds detect fewer events, but provide the opportunity of taking actions when bigger price variations are observed. This proposed trading strategy combines the use of different threshold values in an attempt to take advantage of the different characteristics of smaller and larger thresholds.

From the single-threshold strategy we know that under a specific threshold we should buy towards the end of a downtrend and sell towards the end of an uptrend (i.e. towards the end of the respective OS events). Since now we are dealing with multiple thresholds, each threshold summarises the data in a unique way. For example, at one point in time the trading strategy under one threshold could be recommending a buy action, while under a different threshold recommend a sell action. As we have already argued, the advantage of having the multiple thresholds is that we have multiple recommended actions per data point. In order to decide which action to follow, a majority vote takes place.

Table 1
DC strategy parameters.

Parameter	Description
r_u	Average ratio of upwards OS event over the upwards DC event length
r_d	Average ratio of downwards OS event over the downwards DC event length
Q_{trade}	Quantity to trade
$N_\downarrow$	Number of thresholds recommending to buy
$N_\uparrow$	Number of thresholds recommending to sell
N_θ	Total number of thresholds used in the experiments
Q	Quantity for trading
b_1	Start of trading period during an OS event
b_2	End of trading period during an OS event
b_3	Range of prices close to the trading price P_{trade} that a trade can be perfomed

In order to allow for a majority vote, we associate each DC treshold to an equal weight of 1 (vote). Therefore, $W_1 = W_2 = W3 = \ldots = W_{N_\theta} = 1,$ where N_θ is the total number of thresholds used. As a result, at any point in time the trading strategy is able to make a buy/sell/hold recommendation based on the combined recommendations of all thresholds. As we already know, each threshold produces DC events; thus each threshold is able to make this buy/sell/hold recommendation. Since we have N_θ thresholds, this means that at any point in time we receive N_θ recommendations. In order to decide which recommendation to follow, we sum the weights of the thresholds: if the sum of the weights for all thresholds recommending a buy (sell) action is greater than the sum of the weights for all thresholds recommending a sell (buy) action, then the strategy's action will be to buy (sell). The hold action is a special case of both buy and sell and it happens when we are outside the price range recommended by b_3, or when there is not enough quantity to act, see Algorithm 4 lines 8, 11, and 26.

In addition, the multi-threshold trading strategy is able to make recommendations on the trading quantity Q_{trade}. The decision for this quantity is a dynamic decision, taken by the number of DC thresholds that are advising to sell (buy) at a certain point in time: if many thresholds are advising to sell (buy), then the algorithm sells (buys) a higher quantity of the given currency pair. Eqs. (8a) and (8b) present the relevant formulas, for buy and sell, respectively:

$$Q_{trade} = \left(1 + \frac{N_\downarrow}{N_\theta}\right) \times Q \tag{8a}$$

$$Q_{trade} = \left(1 + \frac{N_\uparrow}{N_\theta}\right) \times Q \tag{8b}$$

where Q_{trade} is the quantity to trade, $N_\downarrow$ and $N_\uparrow$ are the number of thresholds recommending to buy and sell, respectively, N_θ is the total number of thresholds used in our experiments, and Q is the user-specified quantity already presented in Section 3.1. As we can see, by taking into account the recommendations given by the DC thresholds, we are giving more or less weight to the Q quantity, resulting to a new quantity Q_{trade}.

This concludes the presentation of the two proposed DC strategies and their respective parameters. For the convenience of the reader, we have summarised and listed these parameters in Table 1. We have also summarised the trading strategy processes into pseudocodes: Algorithm 3 presents an overview of how the return of a trading strategy is calculated. In addition, Algorithm 4 presents how the buy and sell actions take place. While these algorithms are for the multi-threshold strategy, they can also be applied to the single-threshold strategy, where there is only a single weight (for the single threshold) of $W_1 = 1$ and $Q_{trade} = Q$.

Algorithm 3 Pseudocode for calculating the return of a trading strategy.

```
Require: Initialise variables ($cash = budget, Q_{trade} = 0, current = 0$)
Require: $b_1, b_2, b_3$, Q and weight values $W_1 = W_2 = \ldots = W_{N_\theta} = 1$ for each threshold
 1: for $i = 0; i < dataset\_length; i++$ do
 2:     Initialise variables weights for buy and sell: $W_B = W_S = 0$, number of upturn and downturns: $N_\uparrow = N_\downarrow = 0$
 3:     $current \leftarrow current + 1$
 4:     if $P_C > P_{peak}$ then //$P_C$ is the current price and $P_{peak}$ is the highest so-far price.
 5:         $P_{peak} \leftarrow P_C$
 6:     else if $P_C < P_{trough}$ then
 7:         $P_{trough} \leftarrow P_C$
 8:     end if
 9:     for $j = 0; j < N_\theta; j++$ do
10:         Calculate $t_0^U, t_1^U, t_0^D, t_1^D$ as explainedin Equation 7
11:         if event is Downturn Event then
12:             $W_B \leftarrow W_B + W_j$
13:             if current within range of $[t_0^D, t_1^D]$ then
14:                 $N_\downarrow \leftarrow N_\downarrow + 1$
15:             else
16:                 $N_\downarrow \leftarrow N_\downarrow - 1$
17:             end if
18:         else
19:             $W_S \leftarrow W_S + W_j$
20:             if current within range of $[t_0^U, t_1^U]$ then
21:                 $N_\uparrow \leftarrow N_\uparrow + 1$
22:             else
23:                 $N_\uparrow \leftarrow N_\uparrow - 1$
24:             end if
25:         end if
26:     end for
27:     if $W_S > W_B$ then
28:         Perform the sell action for a given quantity [seeAlgorithm 4]
29:     else if $W_S < W_B$ then
30:         Perform the buy action for a given quantity [seeAlgorithm 4]
31:     end if
32: end for
33: $Wealth \leftarrow cash + Q_{trade} \times P_C$
34: $Return \leftarrow 100 \times \frac{wealth}{budget}$
```

Algorithm 4 Pseudocode for performing the buy and sell actions.

```
 1: if $W_S > W_B$ then
 2:     if $N_\uparrow > 0$ and $P_C \geq b_3 \times P_{peak}$ then
 3:         $Q_{trade} \leftarrow (1 + \frac{N_\uparrow}{N_\theta}) \times Q$
 4:         if $Q_{trade} > 0$ or ($Q_{trade} \leq 0$ and $|Q_{trade}| \times P_C \leq shortSellingAllowance \times budget$) then
 5:             $Cash \leftarrow Cash + Q_{trade} \times P_C$
 6:             $PFL \leftarrow PFL - Q_{trade}$ // PFL stands for Portfolio, i.e. the amount/quantity of the currency pair we are currently holding
 7:         else
 8:             Hold
 9:         end if
10:     else
11:         Hold
12:     end if
13: else if $W_S < W_B$ then
14:     if $N_\downarrow > 0$ and $P_C \leq P_{trough} + (P_{trough} \times (1 - b_3))$ then
15:         $Q_{trade} \leftarrow (1 + \frac{N_\downarrow}{N_\theta}) \times Q$
16:         if $cash > Q_{trade} \times P_C$ then
17:             $Cash \leftarrow Cash - Q_{trade} \times P_C$
18:             $PFL \leftarrow PFL + Q_{trade}$
19:         else
20:             // Buy only as much as you can afford
21:             $Q'_{trade}$ is the new quantity to be traded, up to the amount we can afford
22:             $Cash \leftarrow Cash - Q'_{trade} \times P_C$
23:             $PFL \leftarrow PFL + Q'$
24:         end if
25:     else
26:         Hold
27:     end if
28: end if
```

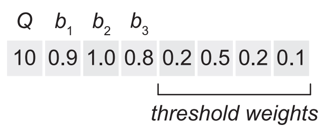

Fig. 5. An example of a 8-gene GA chromosome. The first four genes are : Q, b_1, b_2 and b_3, respectively. The remaining four genes are the weights for the DC thresholds: W_1, W_2, W_3, and W_4.

4. Optimising multi-threshold strategies via a genetic algorithm

In the previous section, we presented two novel trading strategies based on the DC paradigm: a single-threshold strategy and a multi-threshold strategy that builds on top of the single-threshold. While the multi-threshold strategy has the advantage of combining recommendations from different thresholds, a problem that exists is that we do not know how much weight we should give to each threshold. Simply assigning an equal weight of 1 to all of the thresholds might be a naive approach. Some thresholds might be more useful than others, hence we should give them more weight. Thus, we use a genetic algorithm (GA) to evolve *real values* for the weight of each DC threshold. In addition, we also evolve some other DC parameters that are crucial to the success of the trading strategy. All these are discussed next, in Section 4.1, where the GA representation is presented. Then, Section 4.2 presents the GA operators and Section 4.3 presents the fitness function.

4.1. Representation

Each chromosome consists of $4 + N_\theta$ genes, where N_θ is the number of different threshold values of the multi-threshold strategy. The number 4 denotes that in addition to the thresholds, there are also 4 additional parameters to be optimised: Q (first gene), b_1 (second gene), b_2 (third gene), and b_3 (fourth gene). Q, b_1, b_2 and b_3 refer to the DC-related parameters presented in Section 3.1. Each remaining gene in the chromosome (positions 5 to $[4 + N_\theta]$) represents a weight associated to a given threshold. Thus, after first deciding the DC threshold values (through parameter tuning) and generating the DC events per threshold, each GA gene is assigned the same initial weight. Therefore, $W_1 = W_2 = W3 = \ldots = W_{N_\theta} = \frac{1}{N_\theta}$. The GA then evolves the weight for each threshold (in addition to the 4 parameter values in positions 1–4).

As a result, at any point in time a GA individual is able to make a buy/sell/hold recommendation based on the combined recommendations of all thresholds by using the majority vote mechanism we presented in Section 3.2. An example of an 8-gene GA chromosome is presented in Fig. 5.

Based on this example, the GA recommends buying/selling a quantity of Q equal to 10, and only acting in the period [0.9, 1.0] of the estimated duration of the OS event (i.e., in the last 10% of the length of the OS event). In addition, the fourth gene recommends to only consider prices that are within a 20% range (the value of $b3$ is 0.8, so $1.0 - 0.8 = 0.20$ or 20%) of the highest (lowest) recorded price P_{peak} (P_{trough}). In addition, to decide the trading action, we would check the recommendation of each individual threshold. For this example, let us assume that the first threshold recommends buy, the second threshold recommends sell, the third threshold recommends buy, and the fourth threshold recommends hold. We would then sum up the weights of the thresholds, according to each action. Therefore, the weight for buying W_B is equal to $W_1 + W_3 = 0.2 + 0.2 = 0.4$, and the weight for selling W_S is equal to $W_2 = 0.5$.[2] Since $W_S > W_B$, the GA's recommendation would be to sell.

4.2. Operators

The following three operators are being used during the evolutionary process: elitism, uniform crossover and uniform mutation—as detailed in Section 2.3.

In elitism, the best-performing individual (in terms of fitness) is copied to the next generation. In uniform crossover, two parents are selected via a tournament selection. In this type of crossover, the genes between the two parents are swapped with a fixed probability of 0.5. In addition, we ensure that the value of the third gene is always greater than the value of the second gene, i.e. b_2 always has to be greater than b_1. Lastly, for the uniform mutation operator a single parent is selected, again by tournament selection. With a probability of 0.5, each gene of the chromosome is mutated, and a different value is obtained. It should be clarified here that for the first gene (quantity Q), the mutated value can be any integer up to a pre-specified maximum quantity value; whereas for the remaining genes (i.e., b_1, b_2, b_3 and all weights W), the mutated values are real numbers between 0 and 1, where $b_2 > b_1$.

4.3. Fitness function

Several different metrics have been used in the literature as fitness function in algorithmic trading problems. Some examples are: wealth, profit, return, Sharpe ratio, information ratio (Brabazon & O'Neill, 2006; Bradley, Brabazon, & O'Neill, 2009). In this paper, we set our fitness equal to the total return minus the maximum drawdown, presented in Eq. (9):

$$ff = Return - \alpha \times MDD$$
$$MDD = \frac{P_{trough} - P_{peak}}{P_{peak}}, \tag{9}$$

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.

5. Experimental setup

This section is divided into three parts: Section 5.1, where we present the data we use for our experiments, Section 5.2, where we present the experimental setup, and lastly, Section 5.3, which presents the experimental parameters.

5.1. Data

We use two different types of datasets: (i) tick data and (ii) intra-day data at 10 minute intervals.[3] In the first case of tick data, we use a year's FX spot tick data on a daily basis from the currency pair of GBP/JPY (British Pound and Japanese Yen), for the period June 2013 to May 2014. Thus, we use a daily rolling window, where a single day is used for training the algorithm, and the consecutive day is used for testing the returned model. Exception to this rule was when there is a weekend, which is not taken into account. The number of tick data can vary significantly from day to day, and even more from month to month. Nevertheless, each day has a very high number of observations, giving more than enough training data for the GA to learn and produce a profitable model. As we can observe from Fig. 6, where the minimum and maximum number of daily tick data on a given month are presented, a day could have anything between approximately 70,000 transactions (minimum value of April 2014) to above 900,000 transactions (maximum value of June 2013). It should be noted that this high number of data per day should not considered to be a problem for the DC algorithm, i.e. that the algorithm is dealing with too much data to handle; on the contrary, this is one of the strengths of the algorithm, since it will only be focusing on the important events, thus filtering out all 'noise' from the data.

In addition, we use 10 minute interval high frequency data for the following currency pairs: EUR/GBP (Euro and British Pound), EUR/USD (Euro and US dollar), EUR/JPY (Euro and Japanese Yen), GBP/CHF (British Pound and Swiss Franc), and GBP/USD (British Pound and US dollar). The period is again June 2013 to May 2014. Since the amount of the 10-minute data is significantly less than the tick data (e.g. for the whole of June 2013 for EUR/GBP there's around 3000 entries for the whole month), we test our algorithms 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.

5.2. Algorithmic experimental setup

In order to demonstrate the efficiency of our evolutionary event-based DC approach, we will be comparing it with several other benchmarks. This section presents in detail the different algorithms that we use to benchmark our approach. It should be stated that the objective function (i.e., the function that all trading strategies are optimising) for all of these algorithms is Eq. (9), which was presented earlier in Section 4.3.

Our proposed algorithm is going to be benchmarked against 3 different other types of trading strategies: (i) single-threshold and multi-threshold directional changes, (ii) buy and hold, and (iii) technical analysis. In addition, there are 4 parameters for all DC configurations, which depending on the experimental setup, we optimise or not. These 4 parameters are: Q, b_1, b_2, and b_3. Please refer back to Section 3.1 for a detailed presentation of these parameters. Therefore, by taking the above parameters into account, we have the following different configurations, which will constitute our different algorithmic experimental setups:

Standard directional changes

The purpose here is to present the results of the DC paradigm, under a single-threshold and a multi-threshold framework. The values of thresholds were decided during the parameter tuning process.

[2] As explained in the previous section, the hold action is an exceptional case that is considered as an alternative to buy and sell actions; see Algorithm 4, Lines 8, 11 and 26 for detail.

[3] All data was purchased by OlsenData: http://www.olsendata.com

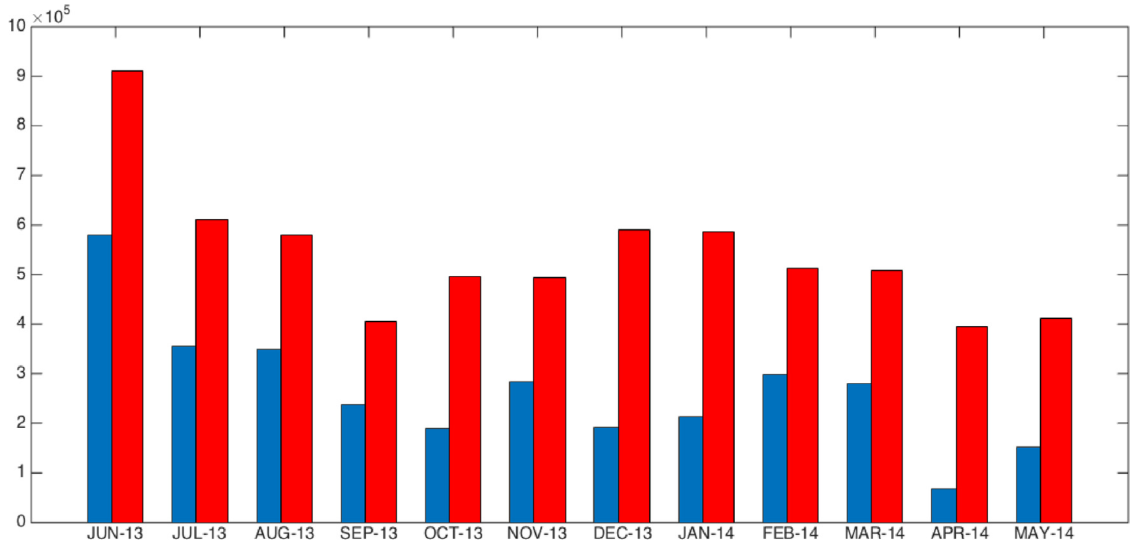

Fig. 6. Minimum and maximum daily tick data (transactions) per month for the GBP-JPY currency pair, for the period June 2013 to May 2014.

1. *Single-threshold, with no evolution (SDC)*
   This is the trading strategy presented in Section 3.1. The 4 parameters [Q, b_1, b_2, b_3] have not been optimised and have fixed values: $Q = 1, b_1 = 0, b_2 = 1, b_3 = 1$, which essentially allow trading of a single quantity throughout the length of the OS event for any given price, no matter how expensive or cheap it is. Of course, this is not the optimal setup for these values and this is why in the next setup we evolve these parameters to obtain better values. However, we consider it important to report results from this setup of SDC to demonstrate that it is crucial to optimise the four parameters [Q, b_1, b_2, b_3]. Lastly, in order to decide which (single) threshold to use, we experiment with several different thresholds (one at a time) and report the performance of the best threshold.
2. *Single-threshold, with evolution on the 4 parameters (SDC_{EVO})*
   As above. The difference is that now we use a standard GA to evolve the values of the 4 numeric parameters [Q, b_1, b_2, b_3]. The idea behind this setup was to evolve the 4 parameters for a single threshold, so that algorithmic performance is optimised for that specific threshold.
3. *Multi-threshold, with no evolution (MDC)*
   This is the trading strategy presented in Section 3.2. The 4 parameters [Q, b_1, b_2, b_3] have not been optimised and have fixed values: $Q = 1, b_1 = 0, b_2 = 1, b_3 = 1$.
4. *Multiple-threshold, with evolution on the 4 parameters (MDC_{EVO})*
   As above. The difference is that now we use a standard GA to evolve the values of the 4 numeric parameters [Q, b_1, b_2, b_3].
   Buy and hold
   Buy and hold is a common benchmark for trading algorithms. Under this strategy, one would buy at a certain point in time and not act (hold) for a long period. Thus, traders are not concerned with short-term price movements, as they expect that in the long term the value of their portfolio will increase.
5. *Buy and Hold (BH)*
   Buy at the beginning of the trading period in August 2013[4], sell at the end of the period, in May 2014.
   Technical analysis
6. *EDDIE*
   The EDDIE algorithm, which uses technical analysis indicators to evolve decision trees that make suggestions for buying, selling, or holding.
   Proposed algorithm
7. *DC+GA*
   Our proposed algorithm, which evolves both the DC threshold weights [$W_1, W_2, \ldots, W_{N_\theta}$] and the 4 DC parameters [Q, b_1, b_2, b_3].

As we can observe, $DC + GA$ will be benchmarked against 6 different types of traging strategies. Next, we present the parameter tuning process that we undertook.

5.3. Experimental parameters

In order to decide the values for the parameters for the algorithms, we undertook a parameter tuning process by using the I/F-Race package (Lopez-Ibanez, Dubois-Lacoste, Stutzle, & Birattari, 2011). It should be noted that buy and hold is a simple process with no parameters that require tuning. I/F-Race implements the iterated racing procedure, which is an extension of the Iterated F-race process. Its main purpose is to automatically configure optimisation algorithms by finding the most appropriate settings, given a set of instances of an optimisation problem. It builds upon the race package by Birattari, Yuan, Balaprakash, and Stutzle (2009).

In order to avoid biased results, we used the first two months of our data (June and July 2013) for each currency pair (both tick and 10-minute data) for tuning purposes. Thus, I/F-Race was applied to the data of June and July 2013. The remaining ten months (August 2013–May 2014) were used only with the tuned parameters, after I/F-Race was complete. At the end of the tuning process, we picked the best parameters returned by I/F-Race. These parameters constitute the experimental parameters for our algorithms. These parameters are presented in Table 2. The buy and hold setup did not have any parameters, so it is not present in Table 2.

As we can observe, we will be using 5 different thresholds. These thresholds are: 0.01%, 0.013%, 0.015%, 0.018%, and 0.02%. In addition, it should be mentioned that when we use an evolutionary algorithm (SDC_{EVO}, MDC_{EVO}, EDDIE, and DC+GA), the experiments are run 50 times on each dataset and the results presented correspond to the average value over the 50 executions; SDC and MDC are run just once per dataset, since they represent deterministic strategies. Similarly, the buy and hold strategy BH is run one time per dataset, as it also represents a deterministic strategy.

[4] The first two months (June and July 2013) were used for parameter tuning, and the remaining ten months were used for our experiments. More details about this in Section 5.3.

Table 2
Experimental parameters determined using I/F-Race.

Parameter	SDC / MDC	SDC_{EVO} / MDC_{EVO} / DC+GA	EDDIE
Population	N/A	1000	500
Generations	N/A	35	30
Tournament size	N/A	7	2
Crossover probability	N/A	0.90	0.90
Mutation probability	N/A	0.10	0.10
Number of thresholds	5	5	N/A
Short selling allowance	0.25	0.25	0.25
MDD weight	0.20	0.20	0.20

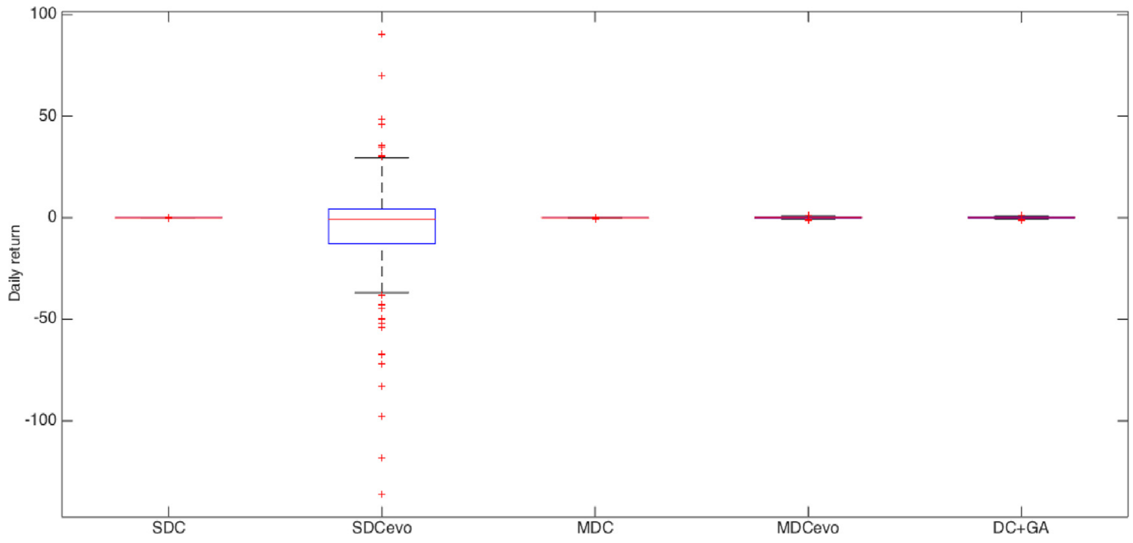

Fig. 7. Day-to-day average return for the period August 2013 to May 2014 for SDC, SDC_{EVO}, MDC, MDC_{EVO} and DC+GA. Results shown in % values.

6. Results

This section presents results for all DC algorithms, and the two physical-time financial benchmarks of BH (buy and hold) and EDDIE (technical analysis), for the currency pair of GBP/JPY under tick data, and for the remaining five currency pairs (EUR/GBP, EUR/JPY, EUR/USD, GBP/CHF, GBP/USD) under the 10-minute interval data. Experiments took place for the 10 month period of August 2013–May 2014. As explained in Section 5, we used a daily rolling window for GBP/JPY, where every day was used for training the algorithms, and the following day was used for testing. The above setup resulted in 205 different datasets, i.e. each algorithm was tested at 205 different unseen datasets for the tick data of GBP/JPY. In addition, for each of the remaining five currency pairs we undertook experiments for each month during the 10 month period August 2013–May 2014. Therefore, this returned 50 different datasets for the 10 minute interval currency pairs. Therefore, *our experiments were conducted over a total of 255 different datasets*.

To increase comprehensibility, we divide this section in the following way: Section 6.1 presents results for the tick data dataset (GBP/JPY), Section 6.2 presents results for the 10 minute interval datasets, Section 6.3 presents the computational time results for the algorithms, and Section 6.4 discusses the results. In addition, Sections 6.1 and 6.2 are futher divided into two Sections 6.1.1, 6.1.2, and 6.2.1, 6.2.2, respectively. Sections 6.1.1 and 6.2.1 present a comparison among the DC algorithms only (i.e. SDC, SDC_{EVO}, MDC, MDC_{EVO}, and DC+GA), in order to identify the best DC setup; Sections 6.1.2 and 6.2.2 present results among the best DC algorithm and the two physical-time financial benchmarks (i.e. BH and EDDIE).

In addition, we would like to remind the reader that the goal of our experiments is threefold: (i) demonstrate that the paradigm of DC returns profitable strategies, (ii) provide evidence that the DC strategies optimised by the GA are more profitable than using standard DC strategies, and (iii) demonstrate that our GA generated strategies outperform typical physical-time based strategies, namely technical analysis and buy and hold. We demonstrate the fulfilment of (i) and (ii) in Sections 6.1.1 and 6.2.1. We demonstrate the fulfilment of (iii) in Sections 6.1.2 and 6.2.2.

Lastly, when an algorithm yields positive return, we will also be commenting on its MDD performance, as an indicator of downside risk.[5] In this way, we make a detailed analysis on both performance metrics (return, MDD), which offers a more holistic view on the results of the trading algorithms.

6.1. Tick data results

6.1.1. Comparison among the DC algorithms

Since each algorithm was tested on a daily basis (excluding weekends) over the 10-month period, we can calculate the daily return for each algorithm. Fig. 7 presents the box and whisker plot for each DC algorithm. As we can observe, SDC, MDC, and MDC_{EVO} and DC+GA show results with very low variance, as all of them are concentrated around the mean that seems to be a value near and above zero. On the other hand, SDC_{EVO} experiences high variance and many extreme values, both positive and negative.

These results are summarised in Table 3. What we can observe is that only MDC_{EVO} and DC+GA have positive mean daily return over the 10 month period. We can also observe that their mean returns are relatively close, as MDC_{EVO}'s return is 0.0677%

[5] We are analysing MDD performance only on strategies with positive returns, as a trader would not consider at all a trading algorithm that yields negative returns.

Table 3
Mean return results for each DC algorithm. Tick data for GBP/JPY. Results shown in % values.

	SDC	SDC_{EVO}	MDC	MDC_{EVO}	DC+GA
Mean	−0.0053	−5.6975	−0.0092	0.0677	0.0730
StandDev	0.0536	25.296	0.1069	0.3673	0.3942
Max	0.0963	90.170	0.1820	1.1684	1.2587
Min	−0.3873	−136.18	−0.7751	−1.1750	−1.3742

Table 4
Mean return results for EDDIE and DC+GA under GBP/JPY's tick data. BH's return (not included in the table, as it does not do daily trading) was −0.1164.

	EDDIE	DC+GA
Mean	−0.1918	0.0730
StandDev	0.3732	0.3943
Max	0.929	1.26
Min	−2.01	−1.37

and DC+GA's is 0.0730%. We should note here that while these return values are relatively low, the reader should keep in mind that trading takes place on a daily basis. Thus, an average daily mean return of the scale of 0.0730% will have a significant cumulative effect in the long run. This is further demonstrated in Section 6.1.2, when we present and discuss with equity curve for DC+GA.

To investigate whether there is a statistical significance between MDC_{EVO} and DC+GA, we ran the Kolmogorov–Smirnoff non-parametric test, with the null hypothesis being that the data from these two algorithms come from the same continuous distribution. The test showed that indeed they both come from the same distribution with a p-value of 0.9638, thus the difference in the mean values is not statistically sigificant. In addition, we look into the maximum drawdown (MDD) values for these two algorithms, to get insight on the downside risk of the trading strategies generated by each algorithm. The average value for MDC_{EVO} is 0.4156%, whereas DC+GA's value is slighly higher, at 0.4251%, showing that both algorithms' strategies have similar downside risk. We further explore the effect of this risk in the next section, when we present the average daily return and its fluctuations.

Since DC+GA returned higher mean return, we use this setup for the comparisons with the physical-time financial benchmarks.

6.1.2. Comparison with physical-time financial benchmarks

Table 4 presents the mean results for DC+GA and EDDIE. We should also note that BH yielded a return of −0.1164%. As we can observe from the table, EDDIE also has a negative mean daily return of −0.1918%. Therefore, DC+GA was the only algorithm among these three with a positive daily return. In addition, a two-sample Kolmogorov–Smirnov test at 5% significance level returned a p-value of 4.7194e-09, and thus showed that DC+GA significantly outperformed EDDIE.

To visualise DC+GA's results, we present the average (over the 50 runs) daily return for the period August 2013 to May 2014 in Fig. 8. As we can observe, the majority of the days experience a positive return. In fact, 58.5% of the tested datasets experienced a positive return (120 out of the total of 205 days). Furthermore, Fig. 9 presents the equity curve for DC+GA. Equity curve is a graphical representation of the change in value of a trading account over a time period. An equity curve with a consistently positive slope would generally indicate that the trading strategies of the account are profitable, while a negative slope would indicate that the account is losing money. As we can observe, the given equity starts from an initial budget of £500K and never drops below this threshold. It generally follows a positive slope, with the only exception of around February-March, where there was a decline. Nevertheless, the curve soon returns to its positive slope, demonstrating the long-run effectiveness of the trading strategy.

This concludes the results under tick data, which showed that DC+GA was ranked first among the other DC versions, and also outperformed the two physical-time financial benchmarks. Next, we present results under the 10 min interval data.

6.2. 10 minute interval data results

This section presents results for the 10-minute interval data for the five currency pairs: EUR/GBP, EUR/JPY, EUR/USD, GBP/CHF, and GBP/USD. We start again by presenting results for the DC algorithms only, in Section 6.2.1. After identifying the best DC setup, we move on to Section 6.2.2, where we compare this best DC setup with BH and EDDIE.

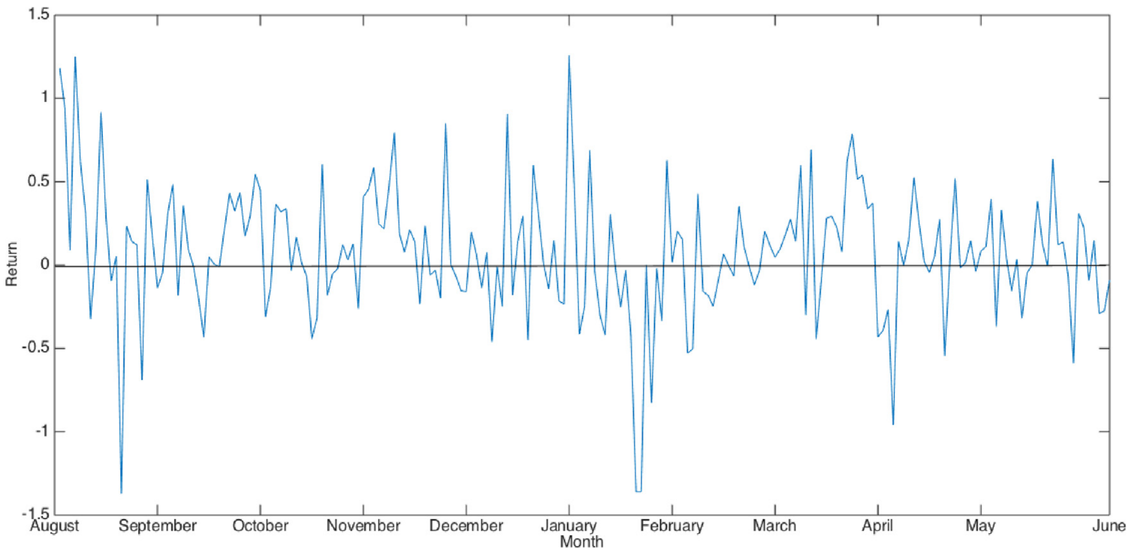

Fig. 8. Day-to-day average return for the DC+GA strategy for the period August 2013 to May 2014 for GBP/JPY's tick data. Results shown in % values.

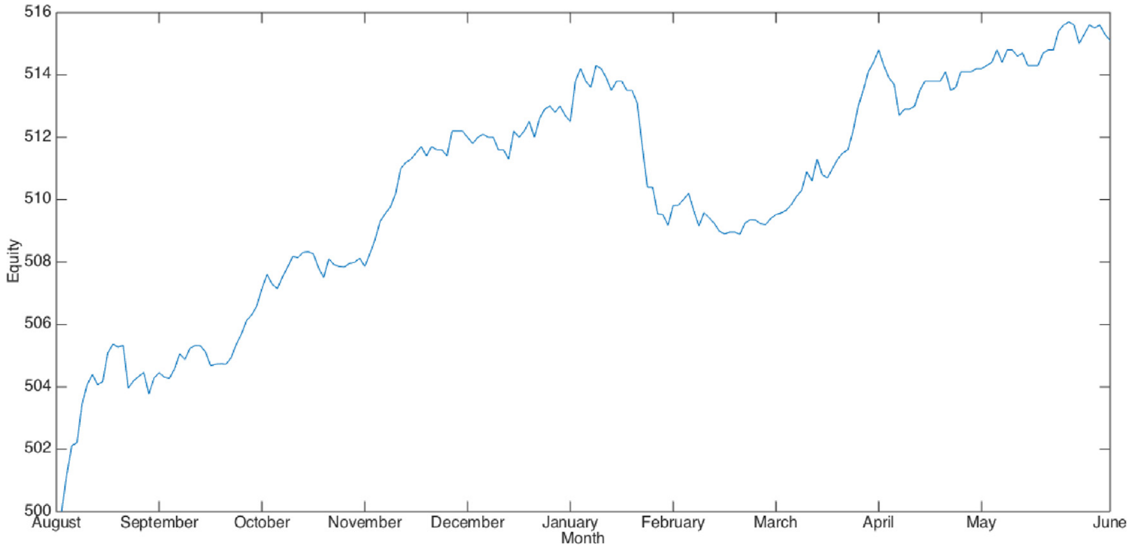

Fig. 9. Equity curve for the DC+GA strategy for the period August 2013 to May 2014 for GBP/JPY's tick data.

6.2.1. Comparison among the DC algorithms

Table 5 presents the return results for each month, for each DC algorithm, under the 10 minute interval data, for each currency pair. Overall, each DC algorithm appears to be able to show positive returns for each currency pair. SDC and SDC$_{EVO}$ are the two algorithms with the highest frequency of positive returns. Out of the 10 months tested per currency pair, SDC had 7 positive returns for EUR/GBP, 6 positive returns for EUR/JPY, 7 positive returns for EUR/USD, 5 positive returns for GBP/CHF, and 3 positive returns for GBP/USD. Similarly, SDC$_{EVO}$'s number of positive returns were 6, 5, 6, 6, and 4.

Table 6 summarises these results. In terms of currency pairs, it appears that EUR/GBP is the easiest to predict, as all algorithms showed non−negative returns. On the other hand, all other currency pairs had two to three currency pairs with negative returns. Overall, SDC and MDC experience again, as with the tick data, a negative mean return. On the other hand, SDC$_{EVO}$, MDC$_{EVO}$ and DC+GA experience positive mean returns, with values close to each other (0.01064%, 0.00875%, and 0.01046%). It thus appears that these three algorithms have similar performance. Once again, we would like to note that while these returns appear to be low, their cumulative effect can be much higher when trading over all 50 datasets available for the 10 minute interval data.

To further investigate the algorithms' performance, we applied Friedman's non-parametric statistical test to compare multiple algorithms. We present the results in Table 7. For each algorithm, the table shows the average rank according to the Friedman test (first column), and the adjusted p-value of the statistical test when that algorithm's average rank is compared to the average rank of the algorithm with the best rank (control algorithm) according to the Hommel post-hoc test (second column) (Demšar, 2006; García & Herrera, 2008). As we can observe from the Friedman test, there is no statistical significance between SDC$_{EVO}$ and DC+GA at the $\alpha = 0.05$ level. Also, there is no statistical significance between SDC$_{EVO}$ and MDC$_{EVO}$ at the $\alpha = 0.05$ level, but there is a statistical significance at the $\alpha = 0.10$ level.

Lastly, we compare the MDD results over the three algorithms that yielded positive mean return (i.e., SDC$_{EVO}$, MDC$_{EVO}$, and DC+GA). DC+GA and MDC$_{EVO}$ have the lowest mean MDD values, 0.03789% and 0.03308%, respectively; on the other hand, SDC$_{EVO}$'s mean MDD value is higher, at 0.05251%. Overall, all algorithms showed very low MDD values. One interesting observation that can be made is that while SDC$_{EVO}$ returned the highest mean return, as we saw in Table 6, it also returned more volatile trading strategies. This is mainly because of the much higher MDD value for EUR/JPY in Table 8 (0.22863% for SDC$_{EVO}$, against 0.12386% and 0.14274% for MDC$_{EVO}$ and DC+GA, respectively). Nevertheless, since SDC$_{EVO}$ ranked first in terms of mean return, we are going to be using it with the comparisons with the physical-time financial benchmarks.

6.2.2. Comparison with physical-time financial benchmarks

Table 9 presents the mean return for EDDIE and SDC$_{EVO}$ under the 10-minute interval datasets (for completeness, we also present the month-by-month return results in the Appendix in Table A.11). We should also note that BH's average return was 0.01274%. As we can observe, EDDIE has again a negative mean return of −0.00873%; it is also worth noting that for all five currency pairs EDDIE's mean return is negative. On the other hand, SDC$_{EVO}$ has a positive return for three currency pairs: EUR/GBP, EUR/JPY, and EUR/USD. Overall, SDC$_{EVO}$'s mean return is 0.01064%. A two-sample Kolmogorov-Smirnov test for EDDIE and SDC$_{EVO}$ returned a p-value of 0.0560, showing that there is a statistical significance between these two algorithms at the 10% significance level. However, the fact that EDDIE returned a negative mean return means that it would not be attractive to an investor as a trading algorithm. This leads us to argue that SDC$_{EVO}$ outperforms EDDIE, while it returns a similar average return with BH.

6.3. Computational times

Table 10 presents the average computational times per run for all algorithms. SDC, MDC, and BH are deterministic algorithms and are thus very fast in executing (around 1 second). All other algorithms have their execution times varying between 10 seconds and 55 seconds. Thus, all algorithms have relatively fast execution times. As we can see, DC+GA ranks third in terms of computational cost, but we believe that this slower execution time is justified by the improvements in the algorithm's mean return performance. Besides, all these are very minor differences, especially after taking into account that the current forecasting application is an offline problem. Lastly, evolutionary algorithms can be easily parallelised since each individual (trading strategy) builds and evaluates a candidate solution independently from all other individuals in the population. Therefore, a large speed up could be obtained by running a parallel version of any evolutionary DC version, as it

Table 5
Monthly return results for each DC algorithm. 10-minute interval data. Results presented per currency pair, in % values.

		SDC	SDC_{EVO}	MDC	MDC_{EVO}	DC+GA
EUR/GBP	August	0.000004	−0.005459	0.000012	−0.007698	−0.009455
	September	−0.000014	−0.004532	−0.000032	−0.003943	−0.002992
	October	0.000007	0.002058	0.000011	−0.000974	−0.001070
	November	0.000002	0.002844	0.000007	0.001022	0.002826
	December	0.000001	0.013137	0.000001	0.007134	0.006939
	January	0.000005	−0.000073	0.000010	0.004246	0.004940
	February	0.000004	0.000962	0.000005	−0.000139	0.000471
	March	−0.000003	−0.002265	−0.000009	0.000006	−0.001710
	April	0.000001	0.000304	0.000002	0.002933	0.003820
	May	−0.000001	0.000892	−0.000001	0.000797	0.000344
EUR/JPY	August	−0.000483	0.155999	−0.000490	0.000000	0.000000
	September	−0.000433	0.050100	−0.001048	0.001103	−0.022498
	October	0.000209	0.060285	−0.000711	0.003731	−0.072743
	November	−0.000822	0.000000	−0.001747	−0.005446	−0.001633
	December	−0.003502	−0.421926	−0.008150	−0.084816	−0.049468
	January	0.000647	0.771926	0.001547	0.584600	0.598602
	February	0.000320	−0.006850	0.000485	−0.004669	−0.004390
	March	0.000395	−0.118628	0.000842	−0.106838	−0.085178
	April	0.000375	0.049121	0.000679	0.015529	0.057428
	May	0.000460	0.040874	0.001183	0.062574	0.064694
EUR/USD	August	−0.000010	0.001232	−0.000022	0.000000	0.000000
	September	0.000003	0.001880	0.000004	−0.000063	−0.000062
	October	0.000016	−0.007077	0.000030	−0.005307	−0.003792
	November	−0.000001	−0.000073	−0.000002	−0.001359	−0.001942
	December	0.000002	−0.005356	0.000007	−0.007442	−0.008586
	January	0.000024	0.014046	0.000052	0.024849	0.018826
	February	0.000010	−0.000601	0.000029	−0.000359	−0.004705
	March	0.000001	0.003289	0.000003	0.001864	0.004594
	April	−0.000001	−0.006851	−0.000009	−0.014233	−0.016055
	May	0.000003	0.000756	0.000008	0.000176	0.000430
GBP/CHF	August	0.000007	−0.004584	0.000009	−0.000334	0.000061
	September	0.000000	−0.014025	0.000000	−0.019435	−0.026564
	October	0.000002	0.002891	0.000001	0.004546	0.007189
	November	0.000004	−0.002861	−0.000008	−0.000798	−0.000490
	December	−0.000006	0.000477	−0.000010	−0.001521	−0.002420
	January	−0.000011	0.000968	−0.000035	−0.000030	−0.000033
	February	0.000012	0.009052	0.000024	0.015172	0.015056
	March	−0.000015	−0.027585	−0.000026	−0.032727	−0.034124
	April	−0.000001	0.000696	−0.000006	−0.000063	−0.000009
	May	0.000002	0.005771	0.000004	−0.001288	0.002493
GBP/USD	August	−0.000004	0.000249	0.000000	−0.000863	−0.000716
	September	−0.000031	−0.014953	−0.000043	0.005087	0.003959
	October	0.000001	0.028377	−0.000002	0.035794	0.052110
	November	−0.000001	0.000013	−0.000009	0.000000	0.000000
	December	0.000001	−0.030266	0.000001	−0.035563	−0.040706
	January	−0.000009	−0.001725	−0.000024	0.000107	0.001240
	February	−0.000004	−0.012477	−0.000005	−0.004516	−0.004187
	March	−0.000015	−0.002918	−0.000026	−0.012024	−0.010595
	April	0.000000	−0.005678	0.000001	−0.002531	−0.003123
	May	0.000000	0.010379	−0.000001	0.021339	0.028387

Table 6
Mean return results for each DC algorithm. 10-minute interval data. Results shown in % values.

	SDC	SDC_{EVO}	MDC	MDC_{EVO}	DC+GA
EUR/GBP	0.00000	0.00079	0.00000	0.00034	0.00063
EUR/JPY	−0.00028	0.05809	−0.00074	0.04658	0.05387
EUR/USD	0.00000	0.00012	0.00001	−0.00019	−0.00125
GBP/CHF	0.00000	−0.00292	0.00000	−0.00365	−0.00388
GBP/USD	−0.00001	−0.00290	−0.00001	0.00068	0.00293
Mean	−0.00006	0.01064	−0.00015	0.00875	0.01046

Table 7
Statistical test results according to the non-parametric Friedman test with the Hommel's post-hoc test. 10-min interval data. Significant differences at the $\alpha = 0.1$ level are shown in boldface.

Algorithm	Average rank	Adjusted p_{Homm}
SDC_{EVO} (c)	1.86	–
DC+GA	1.91	0.80258
MDC_{EVO}	**2.23**	**0.06431**

has actually been shown in Brookhouse, Otero, and Kampouridis (2014), where speed ups of up to 21 times were observed.

6.4. Discussion

From the above results, we can reach the following conclusions.

DC has the potential of returning profitable trading strategies. The single and multi-threshold DC strategies (SDC and MDC) were able to return profitable strategies, as it is evident from the best results of Tables 3 and 6. As we can observe in these tables, all DC algorithms had the maximum return entry as a positive value, which indicates that there was at least one instance per algorithm that had yielded positive return. However, the SDC and MDC paradigm

Table 8
Mean MDD results for each DC algorithm. 10 minute interval data. Results shown in % values.

	SDCevo	MDCevo	DC+GA
EUR/GBP	0.00347	0.00418	0.00520
EUR/JPY	0.22863	0.12386	0.14274
EUR/USD	0.00798	0.01334	0.01402
GBP/CHF	0.01139	0.0114	0.01302
GBP/USD	0.01107	0.0126	0.01449
Mean	0.05251	0.03308	0.03789

Table 9
Mean return results for EDDIE and SDC_{EVO}. 10-minute interval data. BH's average return (not included in the table) was 0.01274%. Results shown in % values.

	EDDIE	SDC_{EVO}
EUR/GBP	−0.00141	0.00079
EUR/JPY	−0.01644	0.05809
EUR/USD	−0.00840	0.00012
GBP/CHF	−0.01114	−0.00292
GBP/USD	−0.00628	−0.00290
Mean	−0.00873	0.01064

Table 10
Mean computational times per run for SDC_{EVO}, MDC_{EVO}, EDDIE, and DC+GA. SDC, MDC, and BH are deterministic algorithms and only take 1 second to execute.

	SDC_{EVO}	MDC_{EVO}	EDDIE	DC+GA
Tick	18 secs	20 secs	55 secs	45 secs
10 min	10 secs	12 secs	25 secs	20 secs

could not consinstently return profitable strategies and thus their mean returns were negative. So it was evident to us that while DC is a promising method, it would benefit from optimising its parameters.

Optimising DC parameters and weights increases the mean return. Using a genetic algorithm to optimise the parameters SDC and MDC increased the mean return, leading to a positive mean return in 3 out of 4 cases. More specifically, under the tick data (Table 3) moving from MDC to MDC_{EVO} increased the mean daily return from −0.0092% to 0.0677%. In addition, under the 10 minute data (Table 6), moving from SDC to SDC_{EVO} led to an increase of mean return from −0.00006% to 0.01064%. Similarly, moving from MDC to MDC_{EVO} increased the mean return from −0.00015% to 0.00875%.

Furthermore, optimising the weights of MDC_{EVO} led to the development of the DC+GA algorithm. DC+GA further improved the mean return of MDC_{EVO} under the tick data, from 0.0677% to 0.0730%. Under the 10 minute data, DC+GA again improved the mean return from MDC_{EVO}'s 0.00875% to DC+GA's 0.01046%. However, DC+GA had slightly lower mean return from SDC_{EVO}'s 0.01046%. Nevertheless, statistical tests showed that the difference in DC+GA's performance and SDC_{EVO} were not statistically significant.

The above leads to us conclude that the introduction of both parameter and weight optimisation is beneficial to DC algorithms.

The DC paradigm is able to outperform traditional physical-time financial benchmarks. The third and last conclusion we can reach is that the DC paradigm is able to outperform traditional physical-time financial benchmarks, such as buy and hold (BH) and technical analysis (EDDIE). In fact, EDDIE never managed to yield positive returns under the experiments in this work, even though in the past has been very successful in similar financial problems (Kampouridis & Otero, 2015; Kampouridis & Tsang, 2010; 2012). On the other hand, BH yielded a negative return under the tick data and a positive return under the 10 minute data. In addition, DC+GA significantly outperformed EDDIE under the tick data at the 5% level, and SDC_{EVO} significantly outperfomed EDDIE under the 10 minute data at the 10% level. However, SDC_{EVO} is heavily dependent on the single threshold we choose to use, so while sometimes it can perform very well (10 minute data), some other times it can perform extremely poorly (tick data). We made a similar observation for SDC_{EVO}'s mean MDD value under the EUR/JPY (Table 8), where it almost doubled the MDD value to its competitors MDC_{EVO} and DC+GA. For this reason, we believe that DC+GA is a better algorithm, as it is more robust, i.e., it is more consistent in terms of positive returns and lower MDD. Hence, for completeness we also run a Kolmogorov-Smirnov test between DC+GA and EDDIE, under the 10 minute data. The null hypothesis is that they both come from the same continuous distribution. The p-value of the test is 0.0560, which just misses rejecting the hypothesis at the 5% level, but does reject it at the 10% level.

So, DC+GA is able to significantly outperform EDDIE at the 5% level under tick data and at the 10% level under the 10 minute data. In addition, DC+GA returned higher mean return than BH under the tick data, and a similar mean return under the 10 minute data. We can thus conclude that DC+GA is able to perform at least as well as BH and outperforms EDDIE.

To summarise, the above three conclusions demonstrate that we have successfully met the three goals of this paper: (i) the DC paradigm returns profitable strategies, (ii) optimising DC strategies by a GA leads to an increase in profits, and (iii) our proposed algorithm, DC+GA, is able to outperform physical-time financial strategies, such as technical analysis, and buy and hold.

7. Conclusion

To conclude, this paper used a new way of summarising high-frequency foreign exchange data, and combined it with a genetic algorithm for optimising its parameters. We used our proposed framework to trade in six different FX markets, and showed that we are able to not only produce average profitable results, but also outperform benchmarks coming from two traditional physical-time approaches (technical analysis, buy and hold). We believe that these results constitute a very promising start, and that further research should take place towards this direction.

More specifically, at the moment, we have focused on the core theory of directional changes and we have derived strategies based on that theory. More work could take place in defining new indicators, derived from the concept of directional changes, in a similar manner that technical analysis indicators exist with physical time. In addition, in our current approach we allowed for the generation of multiple thresholds, and then let the GA combine the suggested action of each threshold. A potential improvement to this would be to leave the decision of the generation of thresholds completely to the optimisation algorithm. For example, the genetic algorithm could be further extended to not only generate thresholds at the beginning of each evolutionary process, but also mutate them, thus generate new ones, during the process. This is by far a more dynamic technique, which could lead to even better trading results. Lastly, we also plan to test our DC+GA algorithm on more data sets from the FX market and also from other type of markets, for instance the stock market.

Appendix A. Monthly return results for EDDIE and SDC_{EVO}

Table A1
Monthly return results for EDDIE and SDC_{EVO}. 10-minute interval data. Results presented per currency pair. Results shown in % values.

		EDDIE	SDC_{EVO}
EUR/GBP	August	0.004458	−0.005459
	September	0.005456	−0.004532
	October	−0.013644	0.002058
	November	0.006171	0.002844
	December	−0.011072	0.013137
	January	−0.010316	−0.000073
	February	−0.001960	0.000962
	March	0.007997	−0.002265
	April	0.000799	0.000304
	May	−0.002026	0.000892
EUR/JPY	August	−0.513460	0.155999
	September	−0.170211	0.050100
	October	−0.088246	0.060285
	November	0.101843	0.000000
	December	0.613728	−0.421926
	January	−0.548754	0.771926
	February	0.033461	−0.006850
	March	0.400234	−0.118628
	April	0.048707	0.049121
	May	−0.041675	0.040874
EUR/USD	August	−0.026099	0.001232
	September	−0.004848	0.001880
	October	−0.053083	−0.007077
	November	0.004349	−0.000073
	December	−0.0004962	−0.005356
	January	−0.008512	0.014046
	February	0.011698	−0.000601
	March	−0.000885	0.003289
	April	−0.006766	−0.006851
	May	0.000674	0.000756
GBP/CHF	August	−0.019175	−0.004584
	September	0.002983	−0.014025
	October	−0.036008	0.002891
	November	−0.030220	−0.002861
	December	−0.019675	0.000477
	January	0.007053	0.000968
	February	0.009592	0.009052
	March	−0.008943	−0.027585
	April	−0.002353	0.000696
	May	−0.014646	0.005771
GBP/USD	August	0.000832	0.000249
	September	−0.009259	−0.014953
	October	0.002542	0.028377
	November	0.001273	0.000013
	December	−0.020717	−0.030266
	January	0.007417	−0.001725
	February	−0.005826	−0.012477
	March	−0.029045	−0.002918
	April	−0.002988	−0.005678
	May	−0.007080	0.010379

References

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

Aloud, M., Tsang, E., Olsen, R., & Dupuis, A. (2012). A directional change events approach for studying financial time series. *Economics, 36*, (online).

Austin, M., Bates, G., Dempster, M., Leemans, V., & Williams, S. (2004). Adaptive systems for foreign exchange trading.. *Quantitative Finance, 4(4)*, 37–45.

Azzini, A., da Costa Pereira, C., & Tettamanzi, A. G. B. (2010). Modeling turning points in financial markets with soft computing techniques. In A. Brabazon, M. O'Neill, & D. G. Maringer (Eds.), *Natural computing in computational finance* (pp. 147–167)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Binner, J., Kendall, G., & Chen, S.-H. (Eds.) (2004). Applications of artificial intelligence in finance and economics (vol. 19). *Advances in Econometrics*. Elsevier.

Birattari, M., Yuan, Z., Balaprakash, P., & Stutzle, T. (2009). F-race and iterated f-race: An overview. Technical Report TR/IRIDIA/2009-018, IRIDIA, Université Libre de Bruxelles, Belgium.

Brabazon, A., & O'Neill, M. (2006). *Biologically inspired algorithms for financial modelling*. Springer.

Bradley, R., Brabazon, A., & O'Neill, M. (2009). In M. Giacobini, & A. Brabazon (Eds.), *Dynamic high frequency trading: a neuro-evolutionary approach* (pp. 233–242)). Springer.

Brookhouse, J., Otero, F., & Kampouridis, M. (2014). Working with OpenCL to speed up a genetic programming financial forecasting algorithm: Intial results. In *Proceedings of the companion publication of the 2014 annual conference on genetic and evolutionary computation (gecco)* (pp. 1117–1124). ACM.

Cervelló-Royo, R., Guijarro, F., & Michniuk, K. (2015). Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data. *Expert Systems with Applications, 4 (14)*, 5963–5975.

Chen, S.-H. (2002). *Genetic algorithms and genetic programming in computational finance*. Springer-Verlag New York, LLC.

Chen, T. L., & Chen, F. Y. (2016). An intelligent pattern recognition model for supporting investment decisions in stock market.. *Information Sciences, 346*, 261–274.

Chiang, W. C., Enke, D., Wu, T., & Wang, R. (2016). An adaptive stock index trading decision support system. *Expert Systems with Applications, 59*, 195–207.

Chourmouziadis, K., & Chatzoglou, P. D. (2016). An intelligent short term stock trading fuzzy system for assisting investors in portfolio management. *Expert Systems with Applications, 43*, 298–311.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Dupuis, A., & Olsen, R. (2012). High frequency finance: Using scaling laws to build trading models. In J. James (Ed.), *Handbook of exchange rates*. Wiley, NJ, USA.

Evans, C., Pappas, K., & Xhafa, F. (2013). Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation. *Mathematical and Computer Modelling, 58*(6), 1249–1266.

García, S., & Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research, 9*, 2677–2694.

Garcia-Almanza, A., Alexandrova-Kabadjova, B., & Martinez-Jaramillo, S. (2013). Bankruptcy prediction for banks: An artificial intelligence approach to improve understandability. In X.-S. Yang (Ed.), *Artificial intelligence, evolutionary computing and metaheuristics*. In *Studies in Computational Intelligence: 427* (pp. 633–656). Springer Berlin Heidelberg.

Glattfelder, J., Dupuis, A., & Olsen, R. (2011). Patterns in high-frequency FX data: Discovery of 12 empirical scaling laws. *Quantitative Finance, 11 (4)*, 599–614.

Godlberg, D. (1989). *Genetic algorithms in search optimisation and machine learning*. Addison-Wesley.

Gypteau, J., Otero, F. E., & Kampouridis, M. (2015). Generating directional change based trading strategies with genetic programming. In M. A. Mora, & G. Squillero (Eds.), *Applications of evolutionary computation: 18th european conference* (pp. 267–278). Springer.

Hu, Y., Liu, K., Zhang, X., Su, L., Ngai, E., & Liu, M. (2015). Application of evolutionary computation for rule discovery in stock algorithmic trading. *Applied Soft Computing, 36*(C), 534–551. doi:10.1016/j.asoc.2015.07.008.

International Monetary Fund (2009). Global financial stability report. https://www.imf.org/external/pubs/ft/gfsr/2009/01/pdf/text.pdf.

Jaisinghani, D. (2016). An empirical test of calendar anomalies for the indian securities markets. *South Asian Journal of Global Business Research, 5 (1)*, 53–84.

Kampouridis, M., & Otero, F. E. B. (2015). Heuristic procedures for improving the predictability of a genetic programming financial forecasting algorithm. *Soft Computing*, 1–16.

Kampouridis, M., & Tsang, E. (2010). EDDIE for investment opportunities forecasting: Extending the search space of the GP. In *Proceedings of the IEEE world congress on computational intelligence* (pp. 2019–2026). Barcelona, Spain

Kampouridis, M., & Tsang, E. (2012). Investment opportunities forecasting: Extending the grammar of a GP-based tool. *International Journal of Computational Intelligence Systems, 5*(3), 530–541.

Kim, Y., & Enke, D. (2016). Developing a rule change trading system for the futures market using rough set analysis. *Expert Systems with Applications, 59*, 165–173.

Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

Kwon, Y.-K., & Moon, B.-R. (2007). A hybrid neurogenetic approach for stock forecasting. *IEEE Transactions on Neural Networks, 18*(3), 851–864. doi:10.1109/TNN.2007.891629.

Lopez-Ibanez, M., Dubois-Lacoste, J., Stutzle, T., & Birattari, M. (2011). The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium.

Mani, S. (1996). Financial forecasting using genetic algorithms. *Applied Artificial Intelligence, 10*, 543–566.

Martinez-Jaramillo, S. (2007). *Artificial Financial Markets: An agent-based Approach to Reproduce Stylized Facts and to study the Red Queen Effect*. CFFEA, University of Essex Ph.D. thesis..

Michalewicz, Z. (2002). *Genetic algorithms + data structures = evolution programs*. Springer.

Mitchell, M. (1996). *An introduction to genetic algorithms*. MIT Press, Cambridge, MA.

Olsen, R. B., Muller, U. A., Dacorogna, M. M., Pictet, O. V., Dave, R. R., & Guillaume, D. M. (1997). From the bird's eye to the microschope: A survey of new stylized facts of the intra-day foreign exchange markets. *Finance and Stochastics, 1 (2)*, 95–129.

Tsang, E., Markose, S., & Er, H. (2005). Chance discovery in stock index option and future arbitrage. *New Mathematics and Natural Computation, World Scientific, 1(3)*, 435–447.

Tsang, E., & Martinez-Jaramillo, S. (2004). Computational finance. *IEEE Computational Intelligence Society Newsletter*, 3–8.

Tsang, E., Tao, R., Serguieva, A., & Ma, S. (2016). Profiling high-frequency equity price movements in directional changes. *Quantitative Finance, 2016*, (online).