Artificial Intelligence Review
https://doi.org/10.1007/s10462-022-10307-0

Algorithmic trading with directional changes

Adesola Adegboye[1] · **Michael Kampouridis**[2] · **Fernando Otero**[1]

Abstract
Directional changes (DC) is a recent technique that summarises physical time data (e.g. daily closing prices, hourly data) into events, offering traders a unique perspective of the market to create novel trading strategies. This paper proposes the use of a genetic algorithm (GA) to optimize the recommendations of multiple DC-based trading strategies. Each trading strategy uses a novel framework that combines classification and regression techniques to predict when a trend will reverse. We evaluate the performance of the proposed multiple DC-strategy GA algorithm against nine benchmarks: five single DC-based trading strategies, three technical analysis indicators, as well as buy-and-hold, which is a popular financial benchmark. We perform experiments using 200 monthly physical time datasets from 20 foreign exchange markets—these datasets were created from snapshots of 10 min intervals. Experimental results show that our proposed algorithm is able to statistically significantly outperform all DC and non-DC benchmarks in terms of both return and risk, and establish multi-threshold DCs as an effective algorithmic trading technique.

Keywords Genetic algorithms · Directional changes · Algorithmic trading · Financial forecasting

1 Introduction

Predicting foreign exchange (Forex) rate is an important step in understanding the relationship among global currencies to evaluate the benefits and risks attached to cross-border trading. The prediction of Forex rate was relatively straightforward up to the early 1970s (Chang and Huang 2014). It was mostly determined by the balance of payments of countries and their level of importation and exportation of goods and services. In 1973, the floating

✉ Michael Kampouridis
mkampo@essex.ac.uk

Adesola Adegboye
adesolaadegboye@gmail.com

Fernando Otero
F.E.B.Otero@kent.ac.uk

[1] School of Computing, University of Kent, Kent, UK

[2] School of Computer Science and Electronic Engineering, University of Essex, Essex, UK

Published online: 07 November 2022

exchange rate was adopted by the world major currencies and in recent times, Forex trading is primarily done electronically (Cheung and Chinn 2001). These changes in currency policy and trading location opened the market to more participants, which led to an increase in market activities. The higher number of participants coupled with local and international supply–demand factors, such as economic, political and psychological, makes Forex forecasting a challenging task (Spero and Hart 2009; Nassirtoussi et al 2011; Frieden 2014; Bilgin et al 2020; Pascual-Ezama et al 2014; Petropoulos et al 2017). These challenges include: (i) short-term pronounced price fluctuation; (ii) a high trading volume of over 6.6 trillion USD per day as of 2019 (Wooldridge 2019) in a market open 24 h/day from Sunday 20:15 GMT to Friday 22:00 GMT across the globe (Sobol and Szmelter 2020); (iii) low profit margin in comparison to fixed income trading (Petropoulos et al 2017); and (iv) noisy and chaotic signals, making separation of uninteresting features from trends difficult (Abu-Mostafa and Atiya 1996; Kamruzzaman et al 2003).

Majority of approaches to predict Forex rates use historic market data captured using a physical time scale (Brabazon et al 2020). A drawback of using a physical time scale is that it makes the flow of physical time discontinuous, exposing market participants to some degree of risks due to ignorance of market activities between discrete time points. An alternative approach is to utilise intrinsic time scale which summarises data by capturing significant activities in the market. In this work, we use directional changes (DCs)—a form of intrinsic time scale to summarise significant market movement. DCs presents an alternative way of sampling data. Instead of taking snapshots of historical data in constant intervals, snapshots are taken when there is a change in price by a predetermined threshold θ. The threshold value is decided in advance by a trader according to their belief of what a significant price change is, either upwards or downwards. Price summaries are thus divided into alternate upward and downward trends. Each of these trends consists of a DC event, which is usually followed by an overshoot (OS) event. Using different threshold values allows the detection of different events and, as a consequence, the creation of different trend summaries. Therefore, the DC framework focuses on the size of a price change as time varies, while under physical time, the time interval is fixed (e.g. daily closing prices). This concept provides traders with new perspectives for price movements analysis and allows them to focus on key price movements, blurring out other price details which could be considered irrelevant. Furthermore, DCs have enabled researchers to discover new regularities in markets, which could have been ignored by the interval-based summaries (Glattfelder et al 2011). Therefore, these new regularities give rise to opportunities for traders and open a whole novel area for research.

As a result, an increasing number of works have been using the DC concept for trading purposes (e.g., Aloud 2020, 2021). Furthermore, Gypteau et al (2015) proposed a genetic programming (GP) based multi-threshold DC (MTDC) strategy, where the terminal nodes of the GP trees were composed of the trading actions (buy/sell/hold) recommended by each DC threshold, and the inner nodes were logical operators for combining the above recommendations. Bakhach et al (2016) proposed a classification algorithm that uses information from event series sampled using smaller threshold to forecasts DC events in DC summary sampled with a larger threshold. Ye et al (2017) proposed a mathematical equation for anticipating the magnitude of OS events. Their result shows that trading based on DC trend (DCT) reversal forecasting techniques yields profitable positive returns at comparatively low risk. Alkhamees and Fasli (2017a) highlighted a problem with summarising price movements based on single fixed threshold over a long physical time period. They argued that if a threshold of 0.01% is used in summarising events and over time significant events level drops to 0.009%, the new types of events will not be captured. Based on this finding, they recommended to trade with event summary sample taken over shorter physical time frame and to recalibrate threshold

size at intervals to identify the most significant event. They proposed to generate event series daily with dynamically adjusted thresholds size according to current and previous day price movement. Comparison results showed that trading on event series generated in shorter time period with dynamic threshold was more profitable than trading on event series generated using fixed threshold over longer periods. A similar conclusion was reached by Alkhamees and Fasli (2017b) having explored the same idea of generating event series using dynamically adjusted thresholds in data stream. Salman et al (2022) proposed several new DC-based trading strategies and optimised their recommendations using a genetic algorithm (GA). Lastly, Long et al (2022) was the first to combine different DC indicators under a GP algorithm.

A particular branch of DC trading research has been to identify the reversal point of a trend. Kampouridis and Otero (2017) and Kampouridis et al (2017) attempted to do this by estimating the length of the DCT. To achieve this, they calculated the average length of DCTs for each dataset in the training set, and then used this value to predict when a trend would end in the test set. Adegboye et al (2017) extended Kampouridis and Otero (2017) by using a symbolic regression GP (SRGP) algorithm to evolve equations that calculated the average DC–OS event length ratio which was thus used to predict the duration of a trend. They identified both linear and non-linear relationships between DC and OS events, which they embedded into a trading strategy and yielded higher returns. Adegboye and Kampouridis (2021) further extended the above work, by observing that trends in DC datasets did not consistently have both DC and OS events. They observed that it was possible to have as little as 14.77% of DC events having a corresponding OS event. Although the number of OS events in a DC summary is threshold dependent, the maximum number of DC–OS event pair observed was 52.46%. To address this issue, they proposed a DCT reversal forecasting algorithm that combined classification with symbolic regression. A tailored classifier distinguished between DCTs composed of OS and DC events and others having only DC event. They then used a tailored SRGP to estimate OS event length of DCTs classified to have an OS event in their training set. Their results showed their approach significantly improved DCT reversal forecasting. The model was embedded in a single threshold-based strategy and tested on 1000 DC datasets created from a 10-min physical time-series from 20 major Forex markets. The trading strategy outperformed other DC and technical analysis-based strategies including buy-and-hold (BandH). Similar results were obtained in Adegboye et al (2021), which applied the above classification technique to a number of different DC algorithms.

While the above classification and regression GPs were novel and effective algorithms, they only used a single DC threshold, i.e., each trading strategy was based on a single DC summary. Therefore, the strategy is constrained by the information provided by that specific DC threshold—this is a major limitation. Besides, it is not easy to know which DC threshold results return a more informative DC summary. To overcome the above drawback, in this paper we propose using a MTDC trading algorithm. As a result, many thresholds will be used simultaneously, and thus at each point in time, there will be multiple buy–sell–hold recommendations. To overcome the conflicting recommendations, we will use a GA to optimise the weights of each DC threshold. The rationale for combining predictions stems from the fact that different kinds of events drive price volatility and a threshold is capable of summarising only one type of such events. Using multiple thresholds will enable us summarise concurrent events, thus, increasing the total number of DC events over the profiling period, consequently providing more opportunities to trade profitably.

We will run experiments on 200 datasets from 20 different Forex currency pairs. The proposed MTDC strategy will be compared against a total of nine benchmarks: five single-threshold DC (STDC) strategies; three technical analysis indicators under physical time; and a BandH strategy. The rest of this paper is organised as follows. Section 2 presents a brief

overview of the concept DCs, and Sect. 3 presents our proposed GA-based multi-threshold trading strategy. Section 4 presents the experimental setup, and Sect. 5 presents and discusses the results. Lastly, Sect. 6 concludes this article and discusses future work.

2 DC background

A DC event is identified by price changes defined by a user-specified threshold value. DC events are divided into upturn and downturn events. Once a DC event is confirmed, the price series usually continue moving towards the same direction (upwards or downwards, depending on what the current DCT is), and they form an OS event. An OS event finishes once a DC event in the opposite direction is confirmed. A DCT, upward or downward, consists of the combination of a DC and an OS events. Different thresholds generate different event series. Smaller thresholds create higher number of DC events than larger thresholds, which produce fewer events.

Let us now look at Fig. 1, where we present how we can summarise a physical-time price series into DC and OS events. In this example, we summarise price movements with two different thresholds, namely $\theta = 0.01\%$ (lines in red) and $\theta = 0.018\%$ (lines in blue). Price changes below θ are not considered a significant event. Price changes above θ are considered significant events, and divide the market into uptrends and downtrends. Solid lines represent DC events, and dashed lines represent OS events. For example, under $\theta = 0.01\%$, between Points A and B we have a downturn DC event followed by a downward OS event from Point B to C; when a trend reversal occurs, an upturn DC event starts from Point C to D. Lastly, between Point D and E it is an upward OS event. The price point where a DCT begins or ends is called *DC extreme point (DCE)*; under $\theta = 0.01\%$, Points A, C, and E are DCE points.

Under $\theta = 0.018\%$ (lines in blue), we obtain a different set of events: from A to B$'$: a downturn event; from B$'$ to C: a downward OS; from C to E: an upturn DC event; lastly, from Point E to E$'$ we have an upward OS trend.

Note that we can only confirm a DC event in hindsight, i.e., after there has been a price change of θ. For instance, under $\theta = 0.01\%$ we would not know we are in an upward trend until we have reached Point D. This point is called a *DC Confirmation point (DCC)*. Before Point D, one would consider that the market has been in a downward trend since Point A. Similarly, we would not know the trend has reversed from upward to downward until we have reached the DCC Point F. It is therefore crucial to be ablel to accurately predict when a trend reversal will take place. Algorithm 1 presents the pseudocode for the transformation of physical time series to event-based (DC) series.

3 Methodology

Our proposed method, which we refer to as multi-threshold DC (MTDC), is a new algorithm for trading under the DC paradigm. This novel algorithm will overcome the limitations from Adegboye and Kampouridis (2021) and Adegboye et al (2021), which were constrained in using a single threshold to generate DC summaries. MTDC allows for multiple DC thresholds and DC summaries to be used. This allows for multiple view of the data. Thus combines the use of different threshold values in an attempt to take advantage of the different characteristics of smaller and larger event kinds that cause price movement. Furthermore, since there are multiple thresholds, there can be multiple recommendations (buy/sell/hold) at any point in

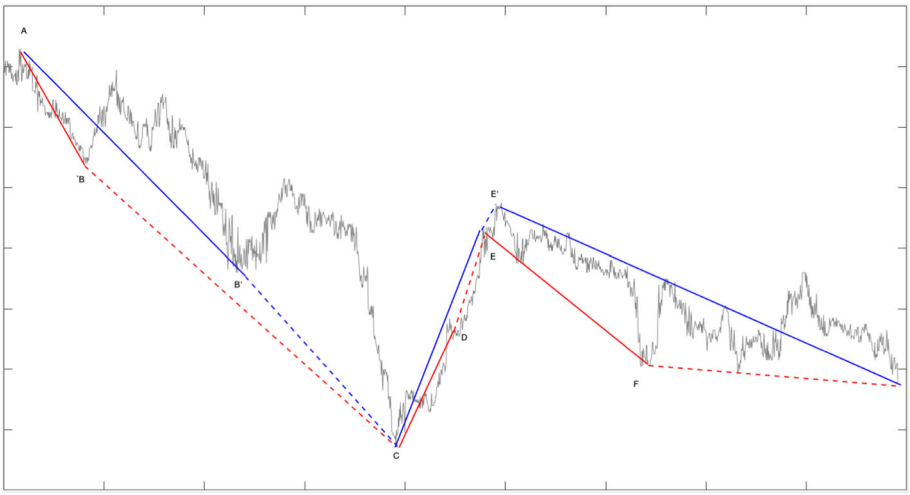

Fig. 1 Directional changes for the GBP/JPY FX currency pair. The red lines represent events created by a threshold $\theta = 0.01\%$, and the blue lines events created by a threshold $\theta = 0.018\%$. DC events are denoted by solid lines, and OS events by dashed lines. Under $\theta = 0.01\%$, we summarise data as follows: Downturn DC event: Point $A \mapsto B$; Downward OS event: Point $B \mapsto C$; Upturn DC event: Point $C \mapsto D$; Upward OS event: Point $D \mapsto E$; Downturn DC event: Point $E \mapsto F$. Under $\theta = 0.018\%$, we summarise data as follows: Downturn DC event: Point $A \mapsto B'$; Downward OS event: Point $B' \mapsto C$; Upturn DC event: Point $C \mapsto E$; Upward OS event: Point $E \mapsto E'$. DC extreme points (DCE): Points A, C, E, and E'. DC confirmation points (DCC): Points B, B', D, E, and F

time. To resolve the conflicting recommendations, we use a GA to decide how much weight we should assign to each DC threshold.

Our trading algorithm can be broken into two main parts: a STDC algorithm and MTDC, which essentially optimises the recommendations from the multiple STDC algorithms. STDC was first presented in Adegboye and Kampouridis (2021), and its main components are summarised in Sect. 3.1. Afterwards, in Sect. 3.2, we discuss the main contribution of this article, namely the MTDC algorithm.

3.1 Singe threshold-based DC strategy

The aim of the single threshold-based DC strategy (STDC) is to predict when the current trend will reverse (trend reversal point) and subsequently use this information during trading. The trend reversal point can be predicted via regression algorithms, where the relationship between the DC and OS lengths is estimated. However, early work in Adegboye et al (2017) showed that simply regressing the above relationship has a major drawback: the resulted function f that describes this relationship does not take into account that many DC events are not followed by an OS event, as they can often be followed by another DC event of the opposite direction. Thus any regression algorithm will learn a DC–OS length relationship by using inaccurate data. To overcome this issue, we first use a classification step, which predicts whether a DC event is followed by an OS event. Introducing this classification step allows us to only perform regression on data that contain consecutive DC and OS events, thus creating more accurate regression models, which itself enables us to predict the end of a trend. This information is then used by a trading strategy.

Algorithm 1 Pseudocode for generating directional changes events given threshold θ.

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

if event is Upturn Event **then**
  if $p(t) \leq p^h \times (1-\theta)$ **then**
    $event \leftarrow Downturn\ Event$
    $P^l \leftarrow p(t)$ //Price at end time for a Downturn Event
    $t_1^{dc} \leftarrow t$ //End time for a Downturn Event
    $t_0^{os} \leftarrow t+1$ //Start time for a Downward Overshoot Event
  else
    if $p^h < p(t)$ **then**
      $p^h \leftarrow p(t)$ //Price at start of Downturn event
      $t_0^{dc} \leftarrow t$ //Start time for Downturn event
      $t_1^{os} \leftarrow t-1$ //End time for an Upturn Overshoot Event
    end if
  end if
else
  if $p(t) \geq p^l \times (1+\theta)$ **then**
    $event \leftarrow Upturn\ Event$
    $P^h \leftarrow p(t)$ //Price at end time for upturn event
    $t_1^{dc} \leftarrow t$ //End time for an Upturn Event
    $t_0^{os} \leftarrow t+1$ //Start time for an Upturn Overshoot Event
  else
    if $p^l > p(t)$ **then**
      $p^l \leftarrow p(t)$ //Price at start time for upturn event
      $t_0^{dc} \leftarrow t$ //Start time for an Upturn Event
      $t_1^{os} \leftarrow t-1$ //End time for a Downturn Overshoot Event
    end if
  end if
end if

Thus, STDC has three main steps: a classification step; a regression step; and a trading step. The process flowchart is illustrated in Fig. 2. Next we briefly present each step. For a more detailed description, the reader is referred to Adegboye and Kampouridis (2021).

3.1.1 The classification step

As there are numerous classification algorithms that can be used for the classification task, STDC uses Auto-WEKA (Thornton et al 2013), an automated machine learning (AutoML) framework.[1] Using Auto-WEKA allows for a tailored classification algorithm and tailored hyperparameters for each dataset. The model classifies a DCT as either composed of DC and OS events (αDC) or only DC event (βDC). If a trend is classified as βDC, then the trading action will be taken at the DCC point (more about this in Sect. 3.1.3). Conversely, if a trend is classified as αDC, the trend is expected to reverse at the end of a sum of the DC event length, known at the DCC point and the OS event length, estimated with a regression model, which is presented in the next section (Sect. 3.1.2).

Lastly, the attributes used for classification are DC-related features, namely:

- *DC event price*, which is the price difference between the upturn/downturn point and the DCC point.
- *DC event time*, which is the time difference between the upturn/downturn point and the DCC point.
- *Speed*, which is the speed at which price change from the start of a trend to the DCC point.
- *Previous DC event price*, which is the price at the previous confirmation point.

[1] Auto-WEKA is a framework with 39 classification algorithms uses high-dimensional stochastic optimisation to fully automate the creation and tuning of tailored classification models.

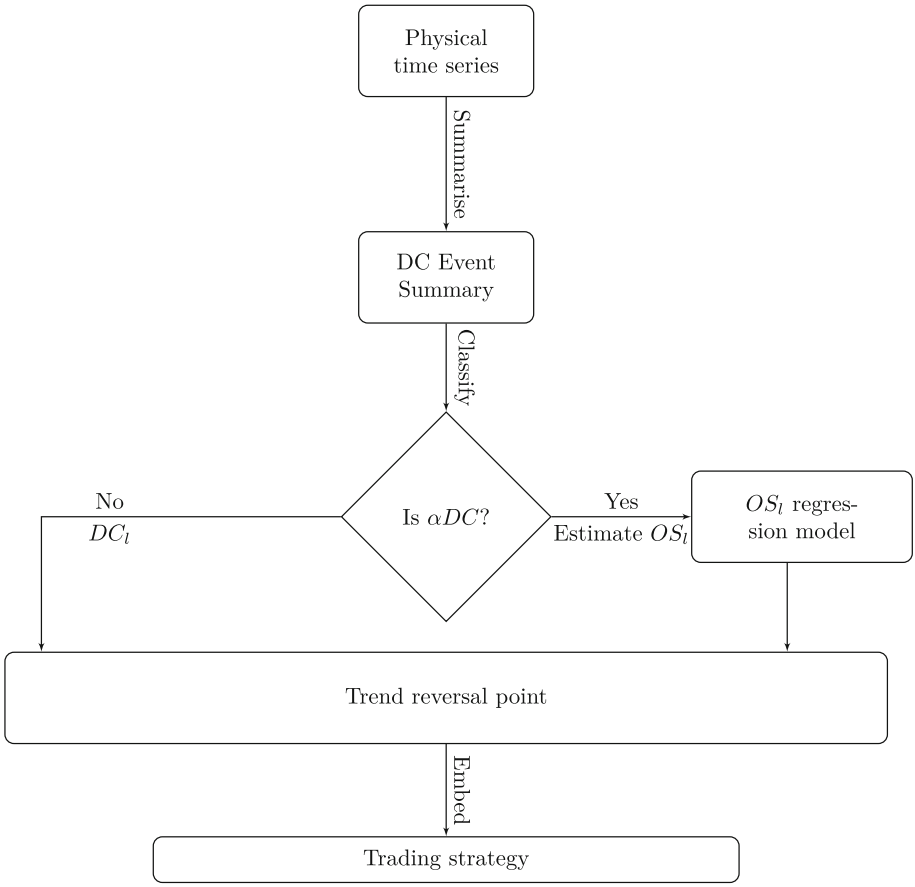

Fig. 2 Predicting trend reversal in DC. A DC trend classified to compose of only DC event is expected to reverse at DCC, while DC trend classified to compose of DC and OS events is expected to reverse at estimated DCE, which is the sum of DC event length at DCC and the OS event length predicted by the SRGP. Once the trend reversal point has been determined, we embed it into a trading strategy and perform trading

- *Previous OS*, which is a boolean variable that indicates whether the immediately previous DCT has an OS event.
- *Flash event*, which is a boolean variable indicating whether the DC event start and end times are equal.

3.1.2 The regression step

Once the classification step is complete, STDC aims to learn the relationship between the DC and OS lengths for each dataset. This is actually a symbolic regression step, which aims to find both the shape of the solution/equation, as well as the value of its parameters. The target solution can be expressed by the generic equation 1, which essentially tells us that the length of an OS event is a function of the length of the DC event.

$$OS_l = f(DC_l), \tag{1}$$

where OS_l is the length of an OS event, DC_l is the length of a DC event.

To find the form of the equation, Algorithm 2, a tree-based SRGP algorithm[2] (SRGP) proposed by Adegboye et al (2017) is used. SRGP is able to evolve scale variant linear and non-linear equations that best express the relationship between DC and OS events lengths in a DC event summary.

Algorithm 2 Pseudocode for evolving SRGP i.e., equation to estimate OS event length after the DC event length is known.

Require: Initialise variables (PopulationSize= 500; GenerationSize= 35; TournamenteSize = 3; CrossoverRate = 0.98; MutationRate = 0.02; MaximumDepth = 3; ElitismRatio = 0.1; Prune = True)

Initialise population: $P \leftarrow$ Generate PopulationSize individuals (Candidate programs) using ramped half-and-half
Evaluate: for each p in P, calculate Fitness with Equation 2
while termination condition not satisfied **do**
  $P_g \leftarrow$ Initialise new population for generation g
  Get elite individuals in P: $ER[1,\ldots,(\text{ ElitismRatio} \times \text{PopulationSize})]$
  Add elite individuals to P_g
  for $i = ER + 1$ to P_g **do**
    if RandomNumber < CrossoverRate **then**
      Select parent1: probabilistically select TournamenteSize individuals from P
      Select parent2: probabilistically select TournamenteSize individuals from P
      $P_{gi} : \leftarrow$ Perform crossover between parent1 and parent2
    end if
    if RandomNumber < MutationRate **then**
      $P_{gi} : \leftarrow$ Perform mutation on P_{gi}
    end if
  end for
  Update: $P \leftarrow P_g$
  Evaluate: for each p in P, calculate Fitness with Equation 2
end while
Return the individual (i.e., equation) with the highest fitness from P

To evolve the equation, SRGP is configured to use 2-arity functions {addition, subtraction, division, multiplication, power} and 1-arity functions {sine, cosine, power, logarithm, exponential} as the function set. The terminal nodes are composed of attribute that represented DC event length and ephemeral random constants (ERC).[3] SRGP is initialised using the ramped half-and-half method. During evolution the fittest 10% of the tree population are copied to the next generation and the rest are evolved with subtree crossover and mutation operators. The fitness of the trees in the population is measured using Eq. 2. It calculates the regression error ε between actual OS length (OS_l) and SRGP estimated OS length. After evolution, the tree with the least regression error in the final generation is selected as the regression model.

$$\varepsilon = \sqrt{\frac{\sum_{i=1}^{N}(OS_l - \hat{OS_l})^2}{N}}, \tag{2}$$

where N is the sample size, ε is the root mean squared error.

A concurrent step with the creation of the regression model is the selection of appropriate thresholds. A pool of 100 thresholds is created with real value numbers from 0.005 with a step size of 0.0025. The SRGP model described above is created for each threshold in the

[2] SRGP searches a space of mathematical expressions to create a symbolic regression model that best expresses the relationship between DC and OS event lengths in an event series.

[3] ERC is a set of randomly generated terminals that retain their values across the population at initialisation and during evolution.

pool. The regression errors of the 100 thresholds' best SRGPs are then measured. The SRGP with the least error with the associated threshold is selected for backtesting.[4]

For backtesting, Adegboye et al (2021) selected the best five thresholds and their SRGPs and traded with them independently. A characteristic of a single threshold-based trading strategy is its obliviousness to slight changes to price movement that are lower than the specified threshold even though the change could also be relevant. For example, if a trader considers a price change of 0.1 to be significant, price changes of 0.0999 are ignored. This inherent limitation of single threshold-based strategy can be addressed by combining information from multiple thresholds (Fernald et al 2021). In Sect. 3.2, we present a multi-threshold trading strategy that combined and optimised information from multiple thresholds using GA (Holland 1992), a well-known technique for solving optimisation problems.

3.1.3 The trading step

As explained earlier, the classification step predicts if a trend is composed of DC and OS events (αDC) or only a DC event (βDC). In the former case, the trend reversal point is predicted to be the end of the OS event as predicted by the SRGP algorithm, while in the former case it is the DCC point.

In order to decide how to trade, we differentiate between opening (sell the base currency and buy the quoted currency) and closing a position (buy the base currency and sell the quoted currency). In order to open a position, there are two requirements: (i) there is not an already open position, and (ii) the return from opening the position would be positive, after accounting for transaction costs. If the above requirements hold, we open a position at the extreme point of an upward DCT. Similarly, to close a position, there are two requirements: (i) there is an existing open position, and (ii) the return from closing the position would be positive after accounting for transaction costs. If these conditions hold, we close the position at the extreme point of a downward DCT.

The extreme point in both of the above cases can be either at a αDC or βDC, depending on the prediction of the classification model. When the above requirements are not met, no trading takes place. All transactions take place by using the entire capital. Transaction cost is 0.025% per transaction.

3.2 Multi-threshold DC strategies using a genetic algorithm

3.2.1 Overview

This strategy builds on the single-threshold strategy by combining market trends and predictions from multiple thresholds. As discussed in Sect. 2, a DC event is identified by a change in the price by a given threshold value. Each DC threshold summarises the data in a unique way: 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.

[4] This idea of dynamically selecting thresholds has also been explored by Alkhamees and Fasli (2017a, b), where a dynamic DC threshold was calculated on a daily basis. In our framework, we do not do this on a daily basis, but instead we simultaneously consider 100 different thresholds and select the one that returns the lowest regression error.

Thus, at one point in time the trading strategy under one threshold could be recommending a buy action, while a different threshold recommending a sell action. In addition, even if all strategies are recommending the same trading action, there might not be consensus on where the trend reversal point is, as each DC summary uses its own SRGP algorithm and thus has different predicted reversal points.

To deal with the above issues, we assign a weight for each DC threshold. Thus, if there are N_θ thresholds, there will be N_θ DCs summaries and as a result N_θ recommendations. Each threshold makes the two following recommendations: (i) what action to take, and (ii) where is the trend reversal point, i.e. when to take the recommended action.

What action to take. A majority vote is performed, based on the thresholds' weights: the weights of the same actions (e.g., buy, buy, …) are summed up and the action with the largest weight is followed. For example, if $N_\theta = 5$ and the sum of weights for the buy actions is 0.65, while the sum of weights for the sell action is 0.35, the action to be taken will be buy. It is worth noting here that the deciding factor is the sum of weights, rather than the number of thresholds recommending an action.

Overall, there are three possible actions that can be recommended by each threshold: buy, sell, and hold. If the action to be taken is a buy, we buy all available base currency in exchange for the quoted currency. If the action to be taken is a sell, we sell all available base currency in exchange for the quoted currency. Therefore, there is no situation where we have both base currency and quoted currency in our portfolio. With regards to a hold action, this can happen in three specific situations: (1) when the action is "sell" and there isn't enough base currency available to sell, (2) when the action is "buy" and there isn't enough base currency to buy and, (3) when the return is negative after deducting transaction costs.

When to act. As each DC summary (each derived by a different DC threshold) can predict a different trend reversal point, there is no consensus as to what point to take a buy/sell/hold action. To alleviate this, we act at the weighted average of the predicted reversal points of the recommended action. To better understand this, let us go back to the previous example of $N_\theta = 5$, where the sum of weights for buy was greater than the sum of weights for sell, and as a result the action to be taken is buy. Let us assume that it was only two thresholds recommending the buy action, with weights $w_1 = 0.3$ and $w_2 = 0.35$, respectively. Let us also assume that the first threshold predicts that the trend will reverse at point $t = 10$, and that the second threshold predicts point $t = 20$. Thus, the buy action will be taken according to Eq. 3:

$$W = \frac{\sum_{i=1}^{n} w_i t_i}{\sum_{i=1}^{n} w_i}, \tag{3}$$

where w_i is each weight value, and t_i is each predicted trend reversal point. Plugging in the above values would give $\frac{(0.3\times 10)+(0.35\times 20)}{(0.3+0.35)} \approx 15$. Thus the buy trading action will take place at point $t = 15$. By following the above method we take into account the threshold weights both in terms of what action to take and when to act.

The above is a brief introduction of the multi-threshold strategies. Apart from what has been discussed above, everything else is similar to the single-threshold strategy. Thus, the MTDC framework consists of four steps: (1) classification step; (2) symbolic regression step; (3) optimised strategy by a GA; and (4) trading step. Steps (1), (2), and (4) follow the same approach as in STDC described in Sect. 3.1. The differentiating step in MTDC is step (3), which is what we have discussed above and is necessary for dealing with the multiple trend reversal points that are returned by each individual threshold's process. Thus, after predicting the trend reversal point per threshold, we assign weights to represent each threshold. Figure 3 presents a high-level overview of the MTDC strategy.

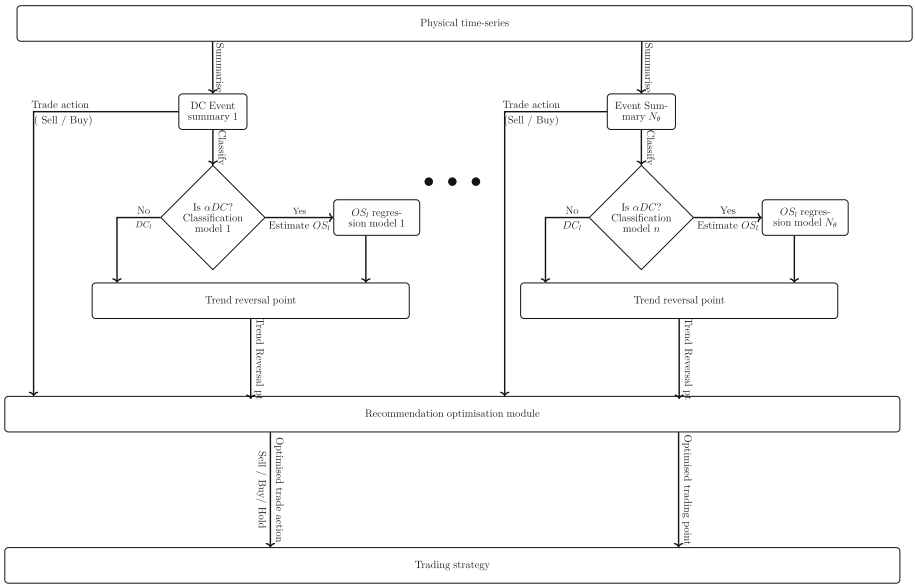

Fig. 3 Our proposed multi-threshold strategy framework. It uses a majority vote system to sum similar trade action recommendations from thresholds and it follows the action with the highest sum. The action is taken after calculating the weighted average of the forecasted trend reversal point of thresholds that recommended the winning action

The only part we have not discussed yet is how the weights are decided. This is an important step, because 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 GA to evolve real values for the weight of each DC threshold. We present the GA next.

3.2.2 Genetic algorithm

GAs are well-known evolutionary algorithms to find solutions to hard optimization problems (Goldberg 1989; Michalewicz 2002; Mitchell 1996). GAs use a population of individuals (candidate solutions) and subject them to an evolutionary process: individuals are evaluated according to how well they solve the problem and combined to generate individuals using genetic operators. In this process, individuals are selected based on their quality, where individuals with a higher quality have a higher chance to be selected and their genetic material to contribute to the creation of the next population.

Representation. Each GA chromosome consists of N_θ genes, where N_θ is the number of thresholds used in the multi-threshold strategy. Each gene is assigned a weight value during population initialisation. The weight is a measure of the importance of a threshold's recommendation in the trading decisions. The weights are real values where the maximum weight value is 1 and the minimum value is 0. We initialise the first gene in the first chromosome with the maximum weight value and initialise the rest of the genes with minimum weight value. We initialise the second gene in the second chromosome with the maximum weight value and initialise the rest of the genes with minimum weight value. We initialise the third

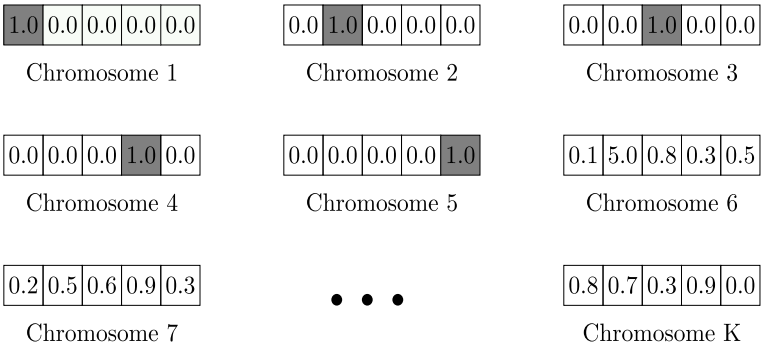

Fig. 4 Illustration of GA population initialisation for $N_\theta = 5$ thresholds

gene in the third chromosome with the maximum weight value and initialise the rest of the genes with minimum weight value. We repeat this initialisation of weights for the first N_θ chromosomes in our GA population. The idea is to ensure that in the worst case scenario, the trading result of our strategy is as good as the result of the best performing single threshold. The genes of the rest of the chromosome in our GA population are randomly assigned real values between the minimum and maximum weights inclusive. The pseudocode presented in Algorithm 3 summarises this procedure and Fig. 4 illustrates the initialisation step. The GA then evolves real value weights for each threshold over a number of generations. At the end of the evolution process our optimisation model is created.

Algorithm 3 Pseudocode for initialising chromosome weight in GA population

```
for i = 0; i < numberOfThresholds; i++ do
    for j = 0; j < chromosomeInPopulation[i]; j++ do
        if index i is threshold position in chromosome then
            w_i ← 1.0
        else
            w_i ← 0.0
        end if
    end for
end for
for i = numberOfThresholds; i < chromosomeInPopulation; i++ do
    for j = 0; j < chromosomeInPopulation[i]; j++ do
        w_i ← RandomNumberFunction(0.0, 1.0)
    end for
end for
```

Genetic operators. We use three operators namely elitism, uniform crossover and uniform mutation. For elitism, we copy the chromosome with the best fitness value into the next generation. For uniform crossover and uniform mutation, individuals from the population are selected into a mating pool. From the pool, through tournament selection, individuals that best favour the optimisation goal are selected as parents of individuals for the next generation. In this work we select as parent, individual in the pool with highest fitness. In uniform crossover both parents contribute their genes where each gene has a fixed probability of 0.5 of being swapped. In uniform mutation operation, the selected parent's gene have a fixed probability of 0.5 of being swapped as well. Figures 5 and 6 illustrate uniform crossover and uniform mutation, respectively.

GA model evaluation. We measure the quality of our GA individual using Sharpe ratio presented in Eq. 6 where R is the return, Q is the quantity traded , TC is the transaction cost

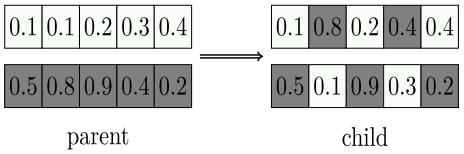

Fig. 5 A sample uniform crossover operation by our GA. Either of the children is randomly selected for the next generation

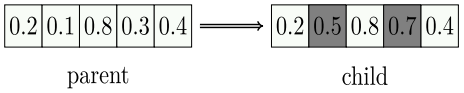

Fig. 6 A sample uniform mutation operation by the GA

discounted from a transaction, $FXrate$ is the FX rate of the relevant currency pair RFR the risk-free rate which we assume to be zero for Forex trading and σ_R the standard deviation of the return over the trading period. We choose Sharpe ratio because it is an aggregate metric of risk-adjusted return, as it measures how well the return compensates an investor for the risk of following the trades strategy.

$$TC = Q * \frac{0.025}{100}, \tag{4}$$

$$R = (Q - TC) * FXrate, \tag{5}$$

$$Sharpe\ Ratio = \frac{R - RFR}{\sigma_R}. \tag{6}$$

Algorithm 4 summarises the procedure of optimising trading actions and trend reversal points. Algorithms 5 and 6 summarise the trading rules applied at optimised trend reversal point at optimal trade action buy and sell respectively.

4 Experimental setup

4.1 Data

We use 10-min interval high frequency data from March 2016 to February 2017 for 16 currency pairs and from June 2013 to May 2014 for 4 currency pairs.[5] These pairs are presented in Table 1.

We consider each month as a separate physical-time dataset. In the tuning phase, we use 40 physical-time datasets (i.e., 20 currency pairs × first 2 months of our physical-time data) to create 200 DC datasets (40 physical-time datasets × 5 thresholds). In the non-tuning phase, 200 datasets (i.e., 20 currency pairs × last 10 months of our physical-time data) are used in creating 1000 DC datasets (i.e., 5[6] thresholds × 20 currency pairs × remaining 10 months of our physical time datasets). In both tuning and non-tuning phases, the DC datasets are split in 70:30 ratio for training and testing respectively.

[5] Data was purchased by OANDA (http://www.oanda.com). Thus the datasets analysed during the current study are not publicly available, but are available from the corresponding author on reasonable request.

[6] We have experimented in aprevious work (Kampouridis and Otero 2017) with the number of thresholds and found that five thresholds is optimal. Higher number of thresholds do not appear to make significant impact on trading performance, while at the same time they significantly affect the computational cost of running the algorithm.

Algorithm 4 Pseudocode for Multi-threshold Optimisation

Require: Initialise base_quantity = budget, quote_quantity = 0.0
Require: current_price = 0.0, LastUpPrice = 0.0
Require: Initialise weight values W_1, W_2, W_3 ... W_{N_θ} according to Algorithm 3
Require: Get forecast model F_1, F_2, F_3 ... F_{N_θ} for each threshold
for i = 0; i < $dataset\_length$; i++ **do**
  Initialise forecast and action dictionary: Dict = empty
  Initialise weights for buy and sell: $W_B = W_s = 0$
  Initialise buy and sell trend reversal list: $List_B = List_S$ = empty
  for j = 0; j < N_θ; j++ **do**
    Initialise trend reversal point: TRP = 0.0
    if event is upturn && DCC_point **then**
      $TRP \leftarrow F_j$
      $Insert\ TRP\ into\ List_S$
      $W_S \leftarrow W_S + W_j$
    else if event is downward trend && DCC_point **then**
      $TRP \leftarrow F_j$
      $Insert\ TRP\ into\ List_B$
      $W_B \leftarrow W_B + W_j$
    end if
  end for
  if $W_S > W_B$ **then**
    $TRP_{optimal_i} \leftarrow optimise\ List_S$ according to Equation 3
    $Insert\ TRP_{optimal_i}\ and\ Sell\ into\ Dict\ at\ position\ i$
  else
    $TRP_{optimal_i} \leftarrow optimise\ List_B$ according to Equation 3
    $Insert\ TRP_{optimal_i}\ and\ Buy\ into\ Dict\ at\ position\ i$
  end if
  if Dict[i] is not empty **then**
    if Dict[i] == $Sell$ **then**
      $current\_price \leftarrow dataset\_length_{i[ask]}$
      Trade with Sell Rule [See Algorithm 5]
    else if Dict[i] == Buy **then**
      $current\_price \leftarrow dataset\_length_{i[bid]}$
      Trade with Buy Rule [See Algorithm 6]
    end if
  end if
end for
$Wealth \leftarrow base\_quantity - budget$
$Return \leftarrow 100 \times \frac{Wealth}{budget}$

Algorithm 5 Trading rules used for selling the base currency

Require: Sell rule
if $base\_quantity > 0$ **then**
  $base\_quantity \leftarrow base\_quantity$ - $transaction\_Cost$
  $quote\_quantity \leftarrow base\_quantity \times current\_price$
  $base\_quantity \leftarrow 0.0$
  $LastUpPrice \leftarrow current\_price$
else Hold
end if

Algorithm 6 Trading rules used for buying the base currency

Require: Buy rule
if $quote\_quantity > 0$ && current_price < LastUpPrice **then**
  $quote\_quantity \leftarrow quote\_quantity$ - $transaction\_Cost$
  $base\_quantity \leftarrow \frac{quote\_quantity}{current\_price}$
  $quote\_quantity \leftarrow 0.0$
else Hold
end if

4.2 Parameter tuning

For the classification step, the only parameter of Auto-WEKA that required tuning was its execution time. This is because Auto-WEKA requires to be given enough time to search its algorithms and hyperparameter space for a classification model that is best in predicting our two class labels (αDC, βDC). We experimented with different runtime configurations namely 15 min, 30 min, 45 min 60 min, 75 min. We chose a runtime of 60 min based on average $f\text{-}measure$, which we observed to diminish at a runtime of 75 min. Depending on

Table 1 FX currency pairs used in our experiments

Currency pairs March 2016 to February 2017	
AUD/JPY	Australian $/Japan. Yen
AUD/NZD	Australian $/N. Zeal. $
AUD/USD	Australian $/US $
CAD/JPY	Canadian $/Japan. Yen
EUR/AUD	Euro/Australian $
EUR/GBP	Euro/British Pound
EUR/CAD	Euro/Canadian $
EUR/CSK	Euro/Czech Krona
EUR/NOK	Euro/Norwegian Krona
GBP/AUD	British Pound/Australian $
NZD/USD	New Zealand $/US $
USD/CAD	US $/Canadian $
USD/NOK	US $/Norwegian Krona
USD/JPY	US $/Japan. Yen
USD/SGD	US $/Singaporean Dollar
USD/ZAR	US $/South African Rand
June 2013 to May 2014	
EUR/USD	Euro/US $
EUR/JPY	Euro/Japan. Yen
GBP/CHF	British Pound/Swiss Franc
GBP/USD	British Pound/US $

Table 2 Regression GP experimental parameters for detecting DC–OS relationship, determined using I/F-Race

Parameter	
Population	500
Generation	37
Tournament size	3
Crossover probability	0.98
Mutation probability	0.02
Maximum depth	3
Elitism	0.10

the number of CPU cores available, it is possible to execute Auto-WEKA in serial or parallel mode. For our experiment we executed Auto-WEKA in serial mode, using 1 CPU core.

With regards to the regression step, we tuned the GP parameters using the I/F-Race package (López-Ibánez et al 2011). I/F-Race package is based on an iterated racing procedure, which is an extension of the Iterated F-race procedure. It implements racing methods for the selection of the best configuration for an optimisation algorithm by empirically selecting the most appropriate settings from a set of instances of an optimisation problem. Table 2 presents the GP configuration to evolve the five symbolic regression models for estimating the OS event length.

Table 3 GA experimental parameters for multi-threshold trading strategy determined using I/F-Race

Parameter	
Population size	500
Generation size	50
Tournament size	7
Crossover probability	0.90
Mutation probability	0.10
Elitism	0.1

With regards to the optimisation part of the DC weights, we again used the I/F-Race package to determine the optimal GA parameters. Table 3 presents the value of the tuned parameters.

4.3 Trading experimental setup

4.3.1 DC-related benchmarks

As a reminder, the aim of this study is to demonstrate that by optimising recommendations from multiple thresholds using ML techniques we can improve profitability and risk, statistically outperforming single threshold-based strategies. To do this, we compare the trading performance from a trading strategy that draws recommendations under a five-threshold DC setup, against five individual strategies, where each strategy draws recommendations from a single DC threshold setup. Each single threshold DC strategy will from now on be denoted as STDC; since we are testing five individual thresholds, we thus have STDC1 (single threshold DC trading strategy 1), STDC2, STDC3, STDC4, and STDC5. It is worth re-iterating that the five individual thresholds can be different per dataset, as they are dynamically chosen. The MTDC strategy will be referred to as MTDC and consists of the same five individual thresholds in a given dataset.

4.3.2 Financial (non-DC) benchmarks

Technical analysis trading strategy Technical analysis is a very popular approach in trading. It uses technical indicators, for insight into when to make trading decisions. We experiment with three trading strategies that utilise the relative strength index (RSI) indicator, the exponential moving average indicator (EMA), and the moving average convergence divergence (MACD).
Buy and hold Buy and hold (BandH) is a well-known benchmark for trading algorithms. Under this trading strategy we buy the quoted currency in the first month of the non-tuning data, and then sell it in exchange for the base currency after the 10 month period.

5 Trading results

This section presents experimental results for our proposed MTDC algorithm.[7] We first compare MTDC's performance against five STDC strategies (Sect. 5.1), and afterwards to

[7] As the focus of this paper is the effectiveness of the proposed MTDC algorithm, we do not present results of its individual components, i.e., the classification and the regression steps. These components have been

Table 4 Average return result (%) for trading strategies of individual single-threshold strategies and multi-threshold strategy

Dataset	STDC1	STDC2	STDC3	STDC4	STDC5	MTDC
AUD/JPY	0.9032	1.1177	1.0361	1.0132	1.2644	**1.4018**
AUD/NZD	0.4716	0.4831	0.3926	0.3365	0.2377	**1.1877**
AUD/USD	0.3970	0.5281	0.5813	0.7310	0.7253	**0.8701**
CAD/JPY	0.8736	0.8969	0.8264	0.7082	0.7935	**1.3208**
EUR/AUD	0.6808	0.5261	0.3586	0.3850	0.3508	**1.0787**
EUR/CAD	0.4677	0.3900	0.3471	0.4886	0.4250	**0.9773**
EUR/CSK	0.0232	0.0372	0.0025	0.0474	0.0432	**0.3955**
EUR/GBP	0.2132	0.2712	0.0583	0.2139	0.2121	**0.8233**
EUR/JPY	0.5475	0.5171	0.4380	0.5985	0.5385	**0.8509**
EUR/NOK	0.2632	0.4388	0.3222	0.6373	0.2553	**0.8889**
EUR/USD	0.2139	0.2427	0.1494	0.1022	0.0777	**1.0474**
GBP/AUD	0.5770	0.3854	0.5816	0.7964	0.6471	**1.4298**
GBP/CHF	0.2575	0.0779	0.6074	0.1904	0.3013	**0.5371**
GBP/USD	0.1141	0.1997	0.0648	0.2228	0.1140	**0.8567**
NZD/USD	0.5130	0.5937	0.7069	0.7858	0.5984	**0.9422**
USD/CAD	0.2078	0.1658	0.4274	0.3773	0.4194	**0.8522**
USD/JPY	0.4411	0.6448	0.3829	0.3914	0.3428	**1.2062**
USD/NOK	0.3836	0.4253	1.0093	0.4595	0.4502	**1.5360**
USD/SGD	0.1525	0.1325	0.2305	0.2991	0.3777	**0.7704**
USD/ZAR	1.5811	1.4437	1.8097	1.7583	1.4155	**4.1808**
Average	0.4641	0.4759	0.5167	0.5271	0.4795	**1.1577**

10-min interval out-of-sample data. 20 Different currency pairs and 10 calendar months each representing the physical dataset. Five DC dataset were generated using five dynamically generated thresholds tailored to each DC dataset. Best value for each row (currency pair) is shown in boldface

financial benchmarks (Sect. 5.2). Then, in Sect. 5.3 we discuss computational times. Lastly, we summarise the main findings of our results in Sect. 5.4.

5.1 Summary statistics

Table 4 presents returns of single-threshold and multi-threshold trading strategies calculated monthly. In this table, cases where 0.00 is reported as return indicates that the strategy is passive (i.e., hold action). Trading return results show that the multi-threshold strategy has the highest return (1.15%), which is over 100% better than the best single threshold-based strategy that recorded return of 0.53%. The result of the multi-threshold strategy was also the best per currency pair. Table 5 presents the non-parametric Friedman test with the Hommel post hoc test to determine if the differences in performance are statistically significant. The null hypothesis is that the strategies come from the same continuous distribution. As we can observe, the best ranking strategy is the multi-threshold strategy, and it statistically outranks the 5 single-threshold strategies at the 5% significance level in all pairs.

thoroughly examined in the past, over a total of 1000 datasets by Adegboye and Kampouridis (2021) and Adegboye et al (2021).

Table 5 Statistical test results for average returns according to the non-parametric Friedman test with the Hommel post hoc test of multi-threshold (c) vs. other single-threshold based trading strategies

Trading strategies	Average rank	$Adjust_{pHomm}$
MTDC (c)	1.0000	–
STDC4	**3.3000**	1.4284E−4
STDC2	**3.9999**	1.2302E−6
STDC3	**4.1000**	7.5910E−7
STDC1	**4.2500**	2.5353E−7
STDC5	**4.3000**	1.5846E−7

10-min interval out-of-sample date. Significant differences between the control algorithm [denoted with (c) and the algorithms represented by a row at the $\alpha = 5\%$ level are shown in boldface indicating that the adjusted p value is lower than 0.05]

We also evaluated the risk adjusted return (Sharpe ratio) over the transactions that occurred in the 10-min monthly dataset. Table 6 presents the result, and it shows that multi-threshold strategy outperformed single threshold-based strategy in all 20 currency pairs. The Sharpe ratio of 0.78 is over 200% better than the Sharpe ratio of the best single threshold-based strategy. We also tested the statistical significance of the Sharpe ratio result using Friedman non-parametric test. The null hypothesis is that the strategies come from the same continuous distribution. We reject the null hypothesis because the statistical test results presented in Table 7 shows that multi-threshold strategy outperformed the 5 single-threshold strategies.

We also performed risk analysis, measuring maximum drawdown and standard deviation of our daily return. Table 8 presents the maximum drawdown results, where the lower the drawdown the better the result. Our multi-threshold strategy recorded the lowest overall average maximum drawdown (0.02). On average, the risk was 10 times lower than trading using single-threshold strategies. We also perform Friedman test and Table 9 shows that multi-threshold strategy statistically outperforms all single-threshold strategies at the 5% significance level.

Finally, Table 10 presents the standard deviation results. The results are not as homogenous as in the previous tables, where the multi-threshold strategy is ranking first across all datasets. Nevertheless, MTDC strategy ranked the highest for the number of currency pairs (7 currency pairs), it had the lowest average standard deviation (0.1638). We also performed Friedman statistical test, presented in Table 11 and the multi-threshold strategy ranks first overall. The performance was not statistically significant against any of the single-threshold strategies. Nonetheless, the volatility risk was slightly lowered when trading with MTDC which is noteworthy considering that the average return (see Table 4) more than doubled.

5.2 Financial benchmarks

Since MTDC was found to be the best algorithm in the above tests with other DC-based strategies, we now proceed to compare it to financial benchmarks, namely the RSI, EMA and MACD technical indicators, as well as the well-known BandH benchmark. Under BandH, we buy on the first day of the first month and sell on the last day of the tenth month.

Table 12 compares the mean returns of MTDC and the other strategies. MTDC outperforms the four benchmarks in 13 currency pairs with an overall average return of 1.1577% against a negative average return of −0.128% under BandH, −0.0378% for RSI, 0.1117% for EMA, and −0.1879% for MACD. In addition, MTDC's variance is 0.76; BandH's is 6.91; RSI's

Table 6 Average Sharpe ratio result for trading strategies of individual single-threshold strategies and multi-threshold strategy

Dataset	STDC1	STDC2	STDC3	STDC4	STDC5	MTDC
AUD/JPY	0.3469	0.2082	0.2335	0.2245	0.2451	**0.7026**
AUD/NZD	0.2565	0.2129	0.2289	0.1246	0.3348	**0.7912**
AUD/USD	0.2749	0.2183	0.3149	0.3396	0.3702	**0.8014**
CAD/JPY	0.2614	0.1879	0.3268	0.1664	0.2708	**0.6804**
EUR/AUD	0.2812	0.2310	0.2358	0.2855	0.2961	**0.9101**
EUR/CAD	0.3972	0.1807	0.2865	0.3229	0.2964	**0.7496**
EUR/CSK	0.0970	0.1190	0.0370	0.1893	-0.0555	**1.2658**
EUR/GBP	0.0845	0.0035	0.1589	0.1077	0.2287	**0.7330**
EUR/JPY	0.3539	0.3183	0.3371	0.4049	0.2846	**1.0389**
EUR/NOK	0.1292	0.2177	0.2578	0.2430	0.2778	**0.5835**
EUR/USD	0.2370	0.1381	0.1073	0.1328	0.1258	**0.5673**
GBP/AUD	0.2579	0.2179	0.2326	0.2619	0.3402	**0.9387**
GBP/CHF	0.2793	0.0216	0.3019	0.2840	0.2367	**0.7413**
GBP/USD	0.0779	0.2178	0.1344	0.2539	0.1855	**0.6961**
NZD/USD	0.1753	0.2463	0.2388	0.3418	0.2365	**0.6223**
USD/CAD	0.1780	0.3044	0.3232	0.3508	0.2181	**0.6328**
USD/JPY	0.2140	0.2205	0.0582	0.2940	0.2303	**0.6499**
USD/NOK	0.2614	0.2526	0.3395	0.1712	0.2156	**0.7604**
USD/SGD	0.0434	0.1260	0.1219	0.1236	0.1910	**0.7305**
USD/ZAR	0.2555	0.2741	0.2420	0.2576	0.2401	**0.9430**
Average	0.2231	0.1958	0.2259	0.2440	0.2384	**0.7769**

10-min interval out-of-sample data. 20 Different currency pairs and 10 calendar months each representing the physical dataset. 5 DC dataset were generated using 5 dynamically generated thresholds tailored to each DC dataset. Best value for each row (currency pair) is shown in boldface

Table 7 Statistical test results for average Sharpe ratio according to the non-parametric Friedman test with the Hommel post hoc test of multi-threshold (c) vs. other single-threshold based trading strategies

Trading strategies	Average rank	$Adjust_{pHomm}$
MTDC (c)	1.0000	–
STDC4	**3.4500**	3.4541E−5
STDC5	**3.7000**	1.0046E−5
STDC3	**4.1000**	4.8184E−7
STDC1	**4.2000**	2.5353E−7
STDC2	**4.5500**	9.8300E−9

10-min interval out-of-sample date. Significant differences between the control algorithm [denoted with (c) and the algorithms represented by a row at the $\alpha = 5\%$ level are shown in boldface indicating that the adjusted p value is lower than 0.05]

Table 8 Average maximum drawdown (%) result for trading strategies of individual single-threshold strategies and multi-threshold strategy

Dataset	STDC1	STDC2	STDC3	STDC4	STDC5	MTDC
AUD/JPY	0.7447	0.2441	0.2796	0.2773	0.5053	**0.0262**
AUD/NZD	0.3235	0.2914	0.2642	0.2910	0.1143	**0.0177**
AUD/USD	0.2810	0.1617	0.2001	0.3173	0.2748	**0.0261**
CAD/JPY	0.1864	0.2537	0.2720	0.1687	0.3897	**0.0124**
EUR/AUD	0.5627	0.4365	0.0977	0.2828	0.2400	**0.0087**
EUR/CAD	0.2956	0.3051	0.0928	0.0964	0.1094	**0.0293**
EUR/CSK	0.0007	0.0302	0.0076	0.0383	0.0706	**0.0000**
EUR/GBP	0.1635	0.2833	0.0445	0.1823	0.1492	**0.0077**
EUR/JPY	0.2932	0.3777	0.2856	0.3772	0.3927	**0.0391**
EUR/NOK	0.1774	0.2326	0.1978	0.4144	0.0894	**0.0112**
EUR/USD	0.1499	0.2006	0.0973	0.0832	0.0487	**0.0303**
GBP/AUD	0.3190	0.2690	0.4058	0.4627	0.3772	**0.0074**
GBP/CHF	0.1020	0.1239	0.4167	0.1069	0.0923	**0.0035**
GBP/USD	0.1311	0.1223	0.0753	0.1477	0.0598	**0.0057**
NZD/USD	0.1884	0.1971	0.2058	0.2131	0.1720	**0.0342**
USD/CAD	0.1451	0.0469	0.2685	0.1434	0.3030	**0.0353**
USD/JPY	0.2563	0.3516	0.2688	0.3132	0.1097	**0.0160**
USD/NOK	0.2655	0.3375	0.6848	0.3467	0.3476	**0.0243**
USD/SGD	0.0383	0.0354	0.1351	0.1351	0.1890	**0.0071**
USD/ZAR	1.1300	1.0217	1.3708	1.3680	1.0740	**0.0196**
Average	0.2877	0.2661	0.2835	0.2883	0.2554	**0.0181**

10-min interval out-of-sample data. 20 Different currency pairs and 10 calendar months each representing the physical dataset. 5 DC dataset were generated using 5 dynamically generated thresholds tailored to each DC dataset. Best value for each row (currency pair) is shown in boldface

Table 9 Statistical test results for average maximum drawdown according to the non-parametric Friedman test with the Hommel post hoc test of multi-threshold (c) vs. other single-threshold based trading strategies

Trading strategies	Average rank	$Adjust_{pHomm}$
MTDC (c)	1.0000	–
STDC5	**3.7500**	3.3460E−6
STDC3	**3.9000**	1.8983E−6
STDC1	**3.9000**	1.8983E−6
STDC2	**4.0000**	1.2655E−6
STDC4	**4.4500**	2.7455E−8

10-min interval out-of-sample date. Significant differences between the control algorithm [denoted with (c) and the algorithms represented by a row at the $\alpha = 5\%$ level are shown in boldface indicating that the adjusted p value is lower than 0.05]

Table 10 % Average standard deviation (SD) result for trading strategies of individual single-threshold strategies and multi-threshold strategy

Dataset	STDC1	STDC2	STDC3	STDC4	STDC5	MTDC
AUD/JPY	0.4511	0.5528	0.5686	0.5502	**0.4419**	0.5334
AUD/NZD	0.2481	0.1745	0.1339	**0.0882**	0.0939	0.1048
AUD/USD	0.2142	0.2355	0.2512	0.2939	0.3751	**0.1945**
CAD/JPY	0.3759	0.3256	0.3656	0.2963	**0.2130**	0.3167
EUR/AUD	0.2798	0.2698	0.1849	0.1667	**0.1491**	0.1571
EUR/CAD	0.2184	**0.1827**	0.1910	0.2509	0.1993	0.2714
EUR/CSK	0.0144	0.0247	**0.0087**	0.0301	0.0380	0.0418
EUR/GBP	0.0898	0.1465	**0.0250**	0.0846	0.0802	0.0812
EUR/JPY	0.2309	0.2323	0.2104	0.2762	0.2718	**0.1408**
EUR/NOK	0.1155	0.1676	0.0993	0.1956	0.1349	**0.0863**
EUR/USD	0.0898	0.1326	0.0859	**0.0577**	0.0311	0.0884
GBP/AUD	0.2618	**0.1671**	0.2520	0.3044	0.2601	0.1867
GBP/CHF	**0.1021**	0.1575	0.2216	0.1277	0.1421	0.1647
GBP/USD	0.1104	0.1164	0.0841	0.1188	**0.0993**	0.1073
NZD/USD	0.2209	0.2201	0.2944	0.3090	0.2113	**0.1483**
USD/CAD	0.1218	**0.0676**	0.2414	0.1647	0.2023	0.1356
USD/JPY	0.2053	0.2749	0.2051	0.1725	0.1658	**0.1186**
USD/NOK	0.1543	0.1999	0.4327	0.1649	0.2033	**0.1299**
USD/SGD	**0.0651**	0.0721	0.0868	0.1636	0.1629	0.0925
USD/ZAR	0.4146	0.4244	0.5034	0.6268	0.3746	**0.1764**
Average SD	0.1992	0.2072	0.2223	0.2221	0.1925	**0.1638**

10-min interval out-of-sample data. 20 Different currency pairs and 10 calendar months each representing the physical dataset. 5 DC dataset were generated using 5 dynamically generated thresholds tailored to each DC dataset. Best value for each row (currency pair) is shown in boldface

Table 11 Statistical test results for average standard deviation according to the non-parametric Friedman test with the Hommel post hoc test of MTDC (c) vs. other single-threshold based trading strategies

Trading strategies	Average rank	$Adjust_{pHomm}$
MTDC (c)	2.6999	–
STDC5	3.0500	0.5541
STDC1	3.5500	0.3016
STDC3	3.6500	0.2262
STDC2	3.8500	0.2010
STDC4	4.2000	0.0561

10-min interval out-of-sample date. Significant differences between the control algorithm [denoted with (c) and the algorithms represented by a row at the $\alpha = 5\%$ level are shown in boldface indicating that the adjusted p value is lower than 0.05]

Table 12 Comparison of MTDC to RSI, EMA, MACD and buy-and-hold in terms of average return (%)

Trading strategies	MTDC	BandH	RSI	EMA	MACD
AUD/JPY	**1.4018**	−6.278	0.0000	0.0000	0.0000
AUD/NZD	**1.1877**	−0.516	0.0558	0.0017	0.0047
AUD/USD	**0.8701**	−5.728	0.0464	−0.1452	−0.1473
CAD/JPY	**1.3208**	−4.109	0.0000	0.0000	0.0000
EUR/AUD	**1.0787**	−2.672	−0.0596	0.0566	−0.0916
EUR/CAD	0.9773	**18.555**	−0.0127	−0.2260	−0.3459
EUR/CSK	0.3955	**7.770**	−0.1382	−0.2327	−0.2812
EUR/GBP	**0.8233**	−0.292	−0.0275	−0.1347	−0.2398
EUR/JPY	**0.8509**	−6.211	−0.0222	0.0154	0.0135
EUR/NOK	0.8889	**2.046**	−0.0429	−0.1181	−0.2333
EUR/USD	1.0474	**8.801**	−0.1057	−0.4923	−0.4094
GBP/AUD	1.4298	**3.936**	−0.1595	−0.3021	−0.0606
GBP/CHF	**0.5371**	−2.395	0.0355	−0.2677	−0.3305
GBP/USD	0.8567	**8.464**	0.0080	−0.0755	−0.3612
NZD/USD	**0.9422**	−6.443	0.1238	−0.2339	−0.3662
USD/CAD	0.8522	**2.345**	−0.2991	−0.3056	−0.5711
USD/JPY	**1.2062**	−9.430	0.0000	0.0000	0.0000
USD/NOK	**1.5360**	−6.102	−0.1440	−0.0754	−0.1541
USD/SGD	**0.7704**	0.207	−0.0574	−0.0436	−0.2949
USD/ZAR	**4.1808**	−4.505	0.0439	−0.1118	−0.1879
Mean	**1.1577**	−0.128	−0.0378	0.1117	−0.1879

Table 13 Statistical test results for average return according to the non-parametric Friedman test with the Hommel post hoc test of MTDC (c) vs. RSI, EMA, MACD, and BandH

Trading strategies	Average rank	$Adjust_{pHomm}$
MTDC (c)	1.3500	–
RSI	**2.8499**	0.0027
EMA	**3.3499**	1.2668E−4
BandH	**3.4499**	8.0074E−5
MACD	**4.0000**	4.6321E−7

10-min interval out-of-sample date. Significant differences between the control algorithm [denoted with (c) and the algorithms represented by a row at the $\alpha = 5\%$ level are shown in boldface indicating that the adjusted p value is lower than 0.05]

is 0.09; EMA's is 0.14; and MACD's is 0.16. This indicates that MTDC is not only more profitable, but also less risky than BandH. It is riskier to the technical indicators, but given the significantly higher returns, it can be argued that MTDC's performance is worth the increased risk. These results were also confirmed by a Friedman statistical test (Table 13), where MTDC ranks first with 1.35 and statistically outperforms all four benchmarks at the 5% level.

Table 14 Average computational times per trend for single threshold-based strategy and multi-threshold strategy

Trading strategies	Single threshold	Multi-threshold
Classification	∼ 65 min	∼ 330 min
Estimation	∼ 5.45 min	∼ 5.45 min
GA optimisation	–	∼ 7 min
Trading	∼ 3 s	∼ 9 s

5.3 Computational times

Table 14 presents the average computational time for multi-threshold strategy in comparison to the single-threshold strategies. The results show an increase in computation time taken by multi-threshold strategy. This is expected since it includes the time required to train multiple classification models. Additional time is also used in training the GA-based strategy. The computation time was measured on a non-dedicated[8] Red Hat Enterprise Linux (Maipo) with a 24 core, 2.53 GHz processor and 24 Gigabit memory. Although Auto-WEKA, the tool for our classification step can be executed using multiple threads of concurrent execution, we chose to run in serial mode using a single CPU core due to limitation on hardware resources. Besides the classification step, we acknowledge that improvements can be made in computation time through parallelisation of the different steps that make up the trading strategy framework (Brookhouse et al 2014; Ong and Schroder 2020). We do not consider the additional time to be a significant drawback as the framework is used off-line. As a result, training would not be happening at the same time as trading. Instead, one would train the algorithm separately from the live trading process, and then when a best model is chosen, this would be used during live trading. Therefore, we believe that the significant improvements observed in trading results would likely outweigh the extra computational time needed, and the final say would be for the user/trader to decide what is the impact of the computational times of the MTDC algorithm.

5.4 Summary

We can summarise our findings as follows.

DC-based trading strategies embedded with an optimised multi-threshold trend reversal forecasting algorithm improves profit at reduced risk. As we observed in Tables 4 and 6, profit obtained trading using an optimised multi-threshold strategy outperformed single-threshold based strategies two and four folds respectively. The statistical tests performed show that that the increase in profit is statistically significant. In addition, having better insight into price movement enables traders make better decisions without increasing risk. We were able to achieve the aforementioned profit without increasing risk. As we discussed in Table 11, multi-threshold strategy was unable to statistically outperform single thresholds in standard deviation risk measure; however, it was ranked first, and we consider this a positive result.

Optimisation of individual threshold trading recommendation is beneficial. Optimising trade action recommendations from multiple thresholds using a GA is an effective way of determining the best action to take.

[8] There were other processes unrelated to the experiment running on the server at the time the experiments were performed.

The paradigm of directional changes has a lot of potential. Although this is not a paper that discusses and thoroughly compares DC to physical-time trading strategies, we can make two general comments about DC-based trading strategies: (i) they can be profitable and low risk, and (ii) they can outperform technical analysis indicators. In this paper we only compared the DC performance to three popular indicators, so in future work we could further benchmark the MTDC to more technical indicators.

6 Conclusion

To conclude, this paper presented an investigation of 200 monthly datasets from 20 different Forex currency pairs to demonstrate that trading with a strategy that combines information from multiple DC thresholds leads to significant improvements in profit and risk. We benchmarked the results against single threshold DC strategies, as well as financial benchmarks, such as technical analysis indicators and BandH. Our results confirmed that the optimal combination of recommendations from multiple thresholds leads to a very strong performance across the majority of metrics, which was further supported by strong statistical significance results. These are significant results, because they indicate that the paradigm of DCs is able to be competitive to the physical time paradigm. Lastly, the fact that we run extensive experiments over 200 datasets leads us to believe that our results are not only significant, but also widely applicable.

It is yet to be confirmed whether similar performance can be achieved in other markets (i.e., commodities, bond, indices and stocks, cryptocurrency). It will therefore be relevant to experiment our approach in other markets. In this work we experimented with data sampled at 10 min interval. It will also be worth experimenting with data of higher frequency like 1-min physical time data and tick-data to further evaluate the robustness of the approach. Furthermore, it would be interesting to experiment with the number of thresholds of the MTDC algorithm. Currently, we use five thresholds, but it would be worth allowing the algorithm to dynamically select the number of thresholds for each dataset, similarly to what is already happening with the threshold values, which are dynamically selected from a given pool of thresholds. Finally, it would be valuable performing classification features' analysis, to better understand the contribution of each feature to the classification task.

References

Abu-Mostafa YS, Atiya AF (1996) Introduction to financial forecasting. Appl Intell 6(3):205–213

Adegboye A, Kampouridis M (2021) Machine learning classification and regression models for predicting directional changes trend reversal in FX markets. Expert Syst Appl 173(114):645

Adegboye A, Kampouridis M, Johnson CG (2017) Regression genetic programming for estimating trend end in foreign exchange market. In: 2017 IEEE symposium series on computational intelligence (SSCI), 2017. IEEE, pp 1–8

Adegboye A, Kampouridis M, Otero F (2021) Improving trend reversal estimation in forex markets under a directional changes paradigm with classification algorithms. Int J Intell Syst 36(12):7609–7640

Alkhamees N, Fasli M (2017a) A directional change based trading strategy with dynamic thresholds. In: 2017 IEEE international conference on data science and advanced analytics (DSAA), 2017. IEEE, pp 283–292

Alkhamees N, Fasli M (2017b) Event detection from time-series streams using directional change and dynamic thresholds. In: 2017 IEEE international conference on big data (Big Data), 2017. IEEE, pp 1882–1891

Aloud M (2020) An intelligent stock trading decision support system using the genetic algorithm. Int J Decis Support Syst Technol 12(4):43–54

Aloud M (2021) Intelligent algorithmic trading strategy using reinforcement learning and directional change. IEEE Access 9:114659–114671

Bakhach A, Tsang EPK, Jalalian H (2016) Forecasting directional changes in FX markets. In: IEEE symposium on computational intelligence for financial engineering and economics (IEEE CIFEr'16), 2016, Athens Greece. IEEE, pp 6–9

Bilgin Y, Camgoz SM, Karan MB et al (2020) Understanding the investment behavior of individual investors: an empirical study on forex markets. In: Handbook of research on decision-making techniques in financial marketing. IGI Global, Hershey, pp 228–246

Brabazon A, Kampouridis M, O'Neill M (2020) Applications of genetic programming to finance and economics: past, present, future. Genet Program Evolvable Mach 21(1):33–53

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

Chang JF, Huang YM (2014) PSO based time series models applied in exchange rate forecasting for business performance management. Electron Commerce Res 14(3):417–434

Cheung YW, Chinn MD (2001) Currency traders and exchange rate dynamics: a survey of the us market. J Int Money Finance 20(4):439–471

Fernald JG, Hsu E, Spiegel MM (2021) Is China fudging its GDP figures? Evidence from trading partner data. J Int Money Finance 110(102):262. https://doi.org/10.1016/j.jimonfin.2020.102262

Frieden JA (2014) Currency politics: the political economy of exchange rate policy. Princeton University Press, Princeton

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

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

Gypteau J, Otero FE, Kampouridis M (2015) Generating directional change based trading strategies with genetic programming. In: European conference on the applications of evolutionary computation, 2015. Springer, pp 267–278

Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence. MIT Press, Cambridge

Kampouridis M, Otero FE (2017) Evolving trading strategies using directional changes. Expert Syst Appl 73:145–160

Kampouridis M, Adegboye A, Johnson C (2017) Evolving directional changes trading strategies with a new event-based indicator. In: Asia–Pacific conference on simulated evolution and learning, 2017. Springer, pp 727–738

Kamruzzaman J, Sarker RA, Ahmad I (2003) SVM based models for predicting foreign currency exchange rates. In: Third IEEE international conference on data mining, 2003. ICDM 2003. IEEE, pp 557–560

Long X, Kampouridis M, Kanellopoulos P (2022) Genetic programming for combining directional changes indicators in international stock markets. In: Proceedings of the 17th international conference on parallel problem solving from nature (PPSN), 2022. Springer

López-Ibánez M, Dubois-Lacoste J, Stützle T et al (2011) The iRace package, iterated race for automatic algorithm configuration. Technical report. Citeseer

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

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

Nassirtoussi AK, Wah TY, Ling DNC (2011) A novel forex prediction methodology based on fundamental data. Afr J Bus Manag 5(20):8322

Ong BW, Schroder JB (2020) Applications of time parallelization. Comput Vis Sci 23(1):1–15

Pascual-Ezama D, Scandroglio B, Gil-Gomez de Liaño B (2014) Can we predict individual investors' behavior in stock markets? A psychological approach. Univ Psychol 13(1):25–35

Petropoulos A, Chatzis SP, Siakoulis V et al (2017) A stacked generalization system for automated forex portfolio trading. Expert Syst Appl 90:290–302

Salman O, Kampouridis M, Jarchi D (2022) Trading strategies optimization by genetic algorithm under the directional changes paradigm. In: Proceedings IEEE congress on evolutionary computation, 2022. IEEE

Sobol I, Szmelter M (2020) Retail investors in the foreign exchange market. Prace Nauk Uniwers Ekon Wrocławiu 64(6):168–181

Spero JEE, Hart JA (2009) The politics of international economic relations. Cengage Learning, Boston

Thornton C, Hutter F, Hoos HH et al (2013) Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, 2013. ACM, pp 847–855

Wooldridge PD (2019) FX and OTC derivatives markets through the lens of the triennial survey. BIS Q Rev, December

Ye A, Chinthalapati VR, Serguieva A et al (2017) Developing sustainable trading strategies using directional changes with high frequency data. In: 2017 IEEE international conference on big data (Big Data), 2017. IEEE, pp 4265–4271