Trading Optimization using Directional Changes and AI

Chung Wai Chan

Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam, Hong Kong

Abstract - Most of the financial forecasting methods use a traditional physical time scale, which is taken at fixed intervals (interval-based scale). In this paper, an intrinsic time (event-based scale) using Directional Changes (DC) is applied to focus only on important points that a crucial event happened, filtering out other irrelevant price details and noise. An improved version of the DC algorithms and DC formulas is proposed based on previous research papers. We will then take advantage of this directional changes paradigm to formulate different DC trading strategies. Previous research focuses on intraday tick data and 10-minute interval data in FX markets. To investigate whether the strategies work on non-high frequency data and other markets, we will apply the DC trading strategies to daily close price data and different stocks. The goal of this paper is to demonstrate the DC trading strategies can make profitable returns, evaluate the performance of the DC trading strategies on daily closing price data and stocks, and find out the best heuristic algorithm among Virus Spread Optimization (VSO) and Genetic Algorithm (GA) for the DC trading strategy. The results show that the DC trading strategies can make promising returns on daily closing price data of stocks, but the performance is not stable. Also, the performance of GA is generally better than that of VSO, but more experiments with different parameter settings are needed to have a more comprehensive analysis.

Keywords: Directional Changes (DC), Heuristic Algorithm, Genetic Algorithm (GA), Virus Spread Optimization (VSO)

1. Introduction

1.1. Background

Traditional physical time scale is applied by using market snapshots, taken at fixed intervals (interval-based scale), generating an interval-based summary such as daily close price. But using this interval-based scale to study price fluctuations makes the flow of physical time discontinuous and cannot capture important price movements happening in a short period of time [1]. For example, under an interval-based scale, we cannot capture the 2010 Flash Crash which occurred in the U.S. financial market and lasted for about 36 minutes.

An alternative approach which is using Directional Changes (DC) to create intrinsic time (event-based scale) is introduced [2]. It can capture important price movements that traditional interval-based scale cannot in the market. Under this paradigm, whenever the price changes from the peak (the highest price in the current upward trend) or trough (the lowest price in the current downward trend) by a pre-specified percentage (threshold θ), the change is captured as a directional change. Then, the stock price is considered as changing the direction of its moving trend (from downward to upward, or from upward to downward). All the time points of the price curve are then divided and summarized into upward and downward trends.

Traders can apply this directional changes paradigm on high frequency data which traditional physical time does not work to capture important price movements in the market. Moreover, it helps filter out irrelevant information and noise because it only captures important movements in the market. In this paper, we will use the directional changes paradigm to formulate different trading strategies.

1.2. Review of Literature

Research on directional changes has been conducted continuously. Kampouridis, Gypteau and Otero proposed combing directional changes and genetic programming to predict the daily closing price of the markets [3]. Bakhach, Tsang and Jalalian used classification to construct the directional changes paradigm to predict the FX market [4]. Adegboye, Kampouridis and Johnson used genetic programming to predict the length of directional change and overshoot events to build the trading strategy and then tested it on intraday data of FX market [5]. Evans, Pappas and Xhafa built a decision-making model based on artificial neural networks (ANN) and genetic algorithms (GA) and tested it on intraday data of FX market [6]. Kampouridis and Otero have also given a thorough analysis of the directional changes and built single-threshold and multi-threshold DC trading strategies with genetic algorithm (GA) [7]. They ran the strategies on intraday tick data and 10-minute data on the FX market. From the results, they successfully used the multi-threshold DC strategy combined with GA to generate promising returns and outperform benchmarks which are two traditional physical time approaches: technical analysis, and buy and hold. However, the performance is only promising when using tick data. The performance of 10-minute data is not as good as that of tick data.

In this paper, we aim to offer more analysis on the directional changes paradigm and complement the research gap. Previous papers generally focus on intraday data of the FX market [5], [6], [7]. To investigate whether the strategies also work on non-high frequency data and other markets, we will apply the DC trading strategies to daily close price data and different stocks. Also, the threshold value is an important parameter in the DC

trading strategy, yet most of the papers do not optimize it via heuristic algorithm. The threshold values are tuned by other parameter tuning methods [7], [8], so we will investigate including the optimization of threshold values in the heuristic algorithm.

Besides, under the no-free-lunch theorem (NFL), there is no single heuristic algorithm that can optimally tackle all optimization problems [9], so other heuristic algorithms are worthy to try to combine with the DC trading strategy. In this paper, we will apply a newly proposed heuristic algorithm - virus spread optimization (VSO) which is proven to outperform conventional and state-of-the-art heuristic algorithms in terms of solution fitness, convergence rate, reliability and flexibility [10]. Including the results of VSO can provide more research analysis on DC trading strategies from a different perspective.

In addition, we are in disagreement with some parts of the DC algorithms and DC formulas used in the research paper [7], which we think may depreciate the profit generated by the strategies. We will try to complement the algorithms and further details will be discussed in Sections 2.1 and 2.2.

1.3. Objectives and Deliverable

The objective of this project is to provide possibly improvement to the existing DC trading strategies and provide more experimental results to support the directional changes research area. Also, the DC trading strategies can be used by traders to generate profitable returns. The scope is to apply directional changes to forecast the price trend of the market and build trading strategies based on this concept. The final deliverable is: (i) to demonstrate DC trading strategies can produce at least 12% yearly returns; (ii) to evaluate the performance of DC trading strategies on daily closing price data and stocks; (iii) to find out the best heuristic algorithm among GA and VSO for DC trading strategy.

1.4. Outline

The rest of the paper is organized as follows: Section 2 presents the methodology used in this paper. Section 2.1 presents the theory of directional changes. Section 2.2 presents DC-derived trading strategies. Section 2.3 and 2.4 present how we use genetic algorithm and virus spread optimization respectively to optimize the parameters of these trading strategies. Section 2.5 presents the experimental setup. Section 3 presents and discusses the experimental results. Section 4 concludes this paper.

2. Methodology

2.1. Directional Changes

The directional changes approach can fragment and summarize all the time points into upward and downward trends. A directional change (DC) is identified as a price change that is defined by a pre-specified percentage. The pre-specified percentage is called the threshold value θ, which is decided by the trader in advance. If the price increases by the threshold value θ from the trough (the lowest price in the current downward trend) when it is in a downward trend, the price change is classified as an upward DC. If the price decreases by the threshold value from the peak (the highest price in the current upward trend) when it is in an upward trend, the price change is classified as a downward DC. After an upward (downward) DC event is confirmed, an upward (downward) overshoot (OS) event starts and lasts until the start point of the next DC event. An upward trend is formed by an upward DC event and an upward OS event while a downward trend is formed by a downward DC event and a downward OS event.

Fig.1 shows how to apply directional changes to transform a traditional physical time into an intrinsic time, summarizing into upward and downward trends [7]. If the size of the price change is less than the threshold value θ, it is not considered a DC. On the other hand, if the size of the price change is equal to the threshold value θ, it is considered a DC. We can also observe an OS event always follows a DC. What should also be noted is that different threshold values generate DC and OS events at different time points. In Fig.1, under θ = 0.01%, point A to B is identified as a downward DC while under θ = 0.018%, point A to B' is identified as a downward DC because they are using different threshold values.

We should bear in mind that a directional change is confirmed only after the price has changed by the threshold value θ from the peak (highest price in the current upward trend) or trough (lowest price in the current downward trend) to a lower value or a higher value. In Fig.1, under θ = 0.01%, point C to point D is only confirmed as a directional change at point D because the price has changed by 0.01% from the trough (the price at point C) to a higher value at point D, so point D is also defined as a confirmation point. Before point D, the price had not changed by θ, so the trader would still believe the price is in a downward OS event. We can conclude that point B to C is a downward OS and point C to D is an upward DC only when we look retrospectively at point D. Algorithm 1.1 shows the high-level pseudocode for capturing directional changes [11], yet I am in disagreement with some parts of the algorithm. The algorithm does not cover some special cases, so I have made a complement and built a new Algorithm 1.2. The detailed justification is attached below the algorithm.

Under intrinsic time, an important regularity is the scaling law. From Fig.2, if the threshold of a DC is **θ**, the percentage change of the OS event following the DC is averagely **θ**. If the length of a DC is **t** amount of physical time, the length of the OS event following the DC is averagely **2t** amount of physical time [1]. This regularity

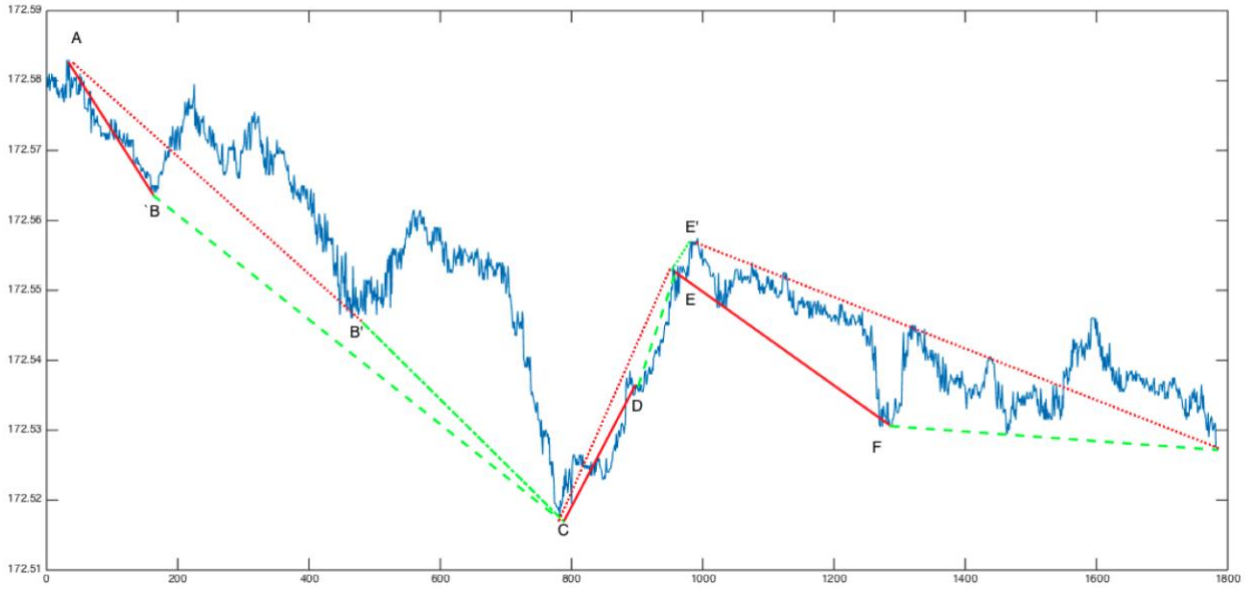

Fig.1 Direction Changes for Tick Data for the GBP/JPY Currency Pair. Under threshold value θ = 0.01%, the solid lines refer to DC events and dash lines refer to OS events. Under threshold value θ = 0.018%, the dotted lines refer to DC events and the dot-dashed lines refer to OS events. Under θ = 0.01%, Point A -> B (Downward DC), Point B -> C (Downward OS), Point C -> D (Upward DC), Point D -> E (Upward OS), Point E -> F (Downward DC). Under θ = 0.018%, Point A -> B' (Downward DC), Point B' -> C (Downward OS), Point C -> E (Upward DC), Point E -> E' (Upward OS) [7].

is important to anticipate the reverse in trend, and then we can perform buy action at the end of the downward OS event and perform sell action at the end of the upward OS event. It is worthy to mention that this regularity is only valid under intrinsic time, which explains why we can make a profit using DC-derived trading strategies.

Lastly, we need to determine the threshold value θ and the weights of each threshold. It is an optimization problem. In previous research papers, threshold values θ are mostly decided by parameter tuning and threshold weights are optimized via heuristic algorithms [7], [8]. In this paper, we will try to optimize both the threshold values θ and threshold weights via heuristic algorithms.

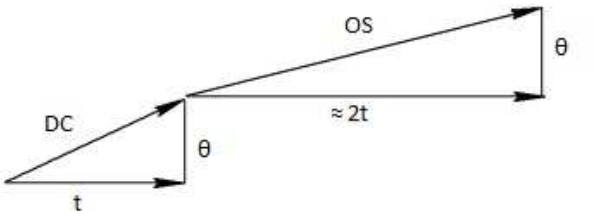

Fig.2 The Scaling Law. If the threshold of a DC is **θ**, the percentage change of the OS event following the DC is averagely **θ**. If the length of a DC is **t** amount of physical time, the length of the OS event following the DC is averagely **2t** amount of physical time [1].

2.2. DC-derived trading strategies

Although we will only build multi-threshold DC trading strategy in this paper, the idea of single-threshold DC trading strategy is still important since the idea of multi-threshold DC trading strategy is based on that of single-threshold DC trading strategy.

2.2.1. Single-threshold DC trading strategy

As we have discussed in Section 2.1, the directional changes approach can transform a physical time into intrinsic time by dividing the price curve into (upward or downward) DC events and OS events. After an upward (downward) DC event happens, an upward (downward) OS event follows. To generate the maximum profit using DC-derived trading strategies, we should act right before the reverse of the trend, which is the endpoint of OS events. If the price curve is in a downward trend, we should buy right at the endpoint of the downward OS event. After the DC updates the trend from downward to upward, we should sell at the endpoint of the upward OS event.

However, it is important to state again that a DC is confirmed only after the price has changed by the threshold value θ. Before the confirmation point of DC, we still believe the price curve is in an (upward or downward) OS event. As a result, anticipating the endpoint of the OS event is required.

To predict the endpoint of the OS event, we use the scaling law, the regularity we have mentioned in Section 2.1. If the length of a DC is **t** amount of physical time, the length of the OS event following the DC is averagely **2t** amount of physical time [1]. We create 2 variables: r_u and r_d. r_u is the ratio of the upward OS event to the upward DC while r_d is the ratio of the downward OS event to the downward DC. By experiments performed by Kampouridis and Otero, these 2 variables are around 2, which has proven the scaling law [7].

r_u and r_d are not enough to define the best trading window since they are only average ratios. OS events last longer or shorter than predicted in most of the cases, so we create 2 more variables: b_1 and b_2, where b_2 is always larger than b_1. They are user-specified parameters which define the trading window during OS event. For example, we set $[b_1, b_2]$ as [0.8, 1] now. If we predict the OS event lasts from t = 10 to t = 20, then the trading window is

Algorithm 1.1 Pseudocode for generating directional changes events [11].

Require: Initialise variables (event is Upward, $p^h = p^l = p(t_0)$, Δx_{dc} *(Fixed)* ≥ 0 , $t_0^{dc} = t_1^{dc} = t_0^{os} = t_1^{os} = t_0$*)*

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

Algorithm 1.1 shows the Pseudocode for generating directional changes events [11].
In Algorithm 1.1, assuming case 1 that:
at t = 0, event is Upward, $t_0^{dc} = 0$, $t_1^{dc} = 0$
at t = 1, $p^h < p(t)$, so $p^h \leftarrow p(t)$, event is Upward, $t_0^{dc} = 1$, $t_1^{dc} = 0$
at t = 2, $p(t) \leq p^h \times (1 - \Delta x_{dc})$, $p^l \leftarrow p(t)$, event is Downward, it is a DC confirmation point, $t_0^{dc} = 1$, $t_1^{dc} = 2$
at t = 3, $p(t) \geq p^l \times (1 + \Delta x_{dc})$, $p^h \leftarrow p(t)$, event is Upward, it is a DC confirmation point, $t_0^{dc} = 1$, $t_1^{dc} = 3$

We can observe that at t = 3, $t_0^{dc} = 1$ according to Algorithm 1.1, but it should be $t_0^{dc} = 2$. Therefore, a logical error exists if there are consecutive DC confirmation points.

Assuming another case 2 that:
at t = 5, $p^h < p(t)$, so $p^h \leftarrow p(t)$, event is Upward, $t_0^{dc} = 5$, $t_1^{dc} = 0$
at t = 6, $p(t) \leq p^h \times (1 - \Delta x_{dc})$, $p^l \leftarrow p(t)$, event is Downward, it is a DC confirmation point, $t_0^{dc} = 5$, $t_1^{dc} = 6$
at t = 7, $p^l \leq p(t) \leq p^l \times (1 + \Delta x_{dc})$, event is Downward, $t_0^{dc} = 5$, $t_1^{dc} = 6$
at t = 8, $p(t) \geq p^l \times (1 + \Delta x_{dc})$, $p^h \leftarrow p(t)$, event is Upward, it is a DC confirmation point, $t_0^{dc} = 5$, $t_1^{dc} = 8$

We can observe that at t = 8, $t_0^{dc} = 5$ according to Algorithm 1.1, but it should be $t_0^{dc} = 6$. Therefore, a logical error exists if the point right after the confirmation point is not high (low) enough to be a directional change, but it is higher (lower) than or same as the confirmation point.

It should be noted that the cases are simplified. It is possible that the false t_0^{dc} and the true t_0^{dc} can have large difference. For example, in case 2, if $p(1)$ is always the p^h till t = 6. Then,
at t = 6, $t_0^{dc} = 1$, $t_1^{dc} = 6$
at t = 7, $t_0^{dc} = 1$, $t_1^{dc} = 6$
at t =8, $t_0^{dc} = 1$, $t_1^{dc} = 8$

at t = 8, $t_0^{dc} = 1$ according to Algorithm 1.1, but it should be $t_0^{dc} = 6$ This large error can depreciate the profit generated by the trading strategies. Considering this, I have amended the algorithm and built Algorithm 1.2.

Algorithm 1.2 Pseudocode for generating directional changes events

Require: Initialise variables (event is Upward, $p^h = p^l = p(t_0)$, Δx_{dc} *(Fixed)* ≥ 0 , $t_0^{dc} = t_1^{dc} = t_0^{os} = t_1^{os} = t_0$*)*

if event is Upward **then**
  if $p(t) \leq p^h \times (1 - \Delta x_{dc})$ **then**
    event ← Downward
    if it is a consecutive confirmation point **then**
      $t_0^{dc} \leftarrow$ t - 1 // Start time for a Downward DC Event
    end if
    $p^l \leftarrow p(t)$
    $t_1^{dc} \leftarrow t$ // End time for a Downward DC Event
    $t_0^{os} \leftarrow$ t + 1 // Start time for a Downward Overshoot Event
  else
    if $p^h < p(t)$ **then**
      $p^h \leftarrow p(t)$
      $t_0^{dc} \leftarrow$ t // Start time for a Downward DC Event
      $t_1^{os} \leftarrow$ t − 1 // End time for an Upward Overshoot Event
    else
      if t - 1 is a confirmation point **then**
        $t_0^{dc} \leftarrow$ t - 1 // Start time for a Downward DC Event
      end if
    end if
  end if
else
  if $p(t) \geq p^l \times (1 + \Delta x_{dc})$ **then**
    event ← Upward
    if it is a consecutive confirmation point **then**
      $t_0^{dc} \leftarrow$ t - 1 // Start time for an Upward DC Event
    end if
    $p^h \leftarrow p(t)$
    $t_1^{dc} \leftarrow t$ // End time for an Upward DC Event
    $t_0^{os} \leftarrow$ t + 1 // Start time for an Upward Overshoot Event
  else
    if $p^l > p(t)$ **then**
      $p^l \leftarrow p(t)$
      $t_0^{dc} \leftarrow$ t // Start time for an Upward DC Event
      $t_1^{os} \leftarrow$ t − 1 // End time for a Downward Overshoot Event
    else
      if t - 1 is a confirmation point **then**
        $t_0^{dc} \leftarrow$ t - 1 // Start time for an Upward DC Event
      end if
    end if
  end if
end if

Algorithm 1.2 shows the improved Pseudocode for generating directional changes events. It may be confused that sometimes $t_0^{dc} > t_1^{dc}$, and $t_0^{os} > t_1^{os}$. In fact, at different time points, they may not refer to the same DC or OS events. But only at the confirmation time points, t_0^{dc} and t_1^{dc} are referring to the same DC event and t_0^{dc} must $< t_1^{dc}$. That is why we only calculate the trading window using Eq.(2) at the confirmation point of DC in Algorithm 2. Also, only at the point right before the DC confirmation point, t_0^{os} and t_1^{os} are referring to the same OS event and t_0^{os} must $< t_1^{os}$.

defined from t = 18 to t = 10. Eventually, we create the formulas in Eq. (1) [7]

$$\begin{aligned} t_0^U &= (t_1^{dc} - t_0^{dc}) \times r_u \times b_1 \\ t_1^U &= (t_1^{dc} - t_0^{dc}) \times r_u \times b_2 \\ t_0^D &= (t_1^{dc} - t_0^{dc}) \times r_d \times b_1 \\ t_1^D &= (t_1^{dc} - t_0^{dc}) \times r_d \times b_2 \end{aligned} \quad (1)$$

where t_0^U and t_1^U are defined as the start and end times for upward OS event, t_0^D and t_1^D are defined as the start and end times for downward OS event by Kampouridis and Otero [7]. t_0^{dc} and t_1^{dc} are the start and end times of the current DC. However, I hold different opinions on these formulas. In my views, t_1^{dc} + 1 should be the start time of upward (or downward) OS event, while t_1^{dc} + 1 + ($t_1^{dc} - t_0^{dc}$) $\times r_u$ (or r_d) should be the end time of upward (or downward) OS event, where ($t_1^{dc} - t_0^{dc}$) $\times r_u$ (or r_d) is the time duration of OS event. ($t_1^{dc} - t_0^{dc}$) $\times r_u$ (or r_d) $\times b_1$ is the time duration between t_1^{dc} and the start time of the trading window. ($t_1^{dc} - t_0^{dc}$) $\times r_u$ (or r_d) $\times b_2$ is the time duration between t_1^{dc} and the end time of the trading window. Therefore. the 4 variables t_0^U, t_1^U, t_0^D, t_1^D should be defined as the trading window instead and the formulas should be defined like Eq. (2).

$$\begin{aligned} t_0^U &= t_1^{dc} + 1 + (t_1^{dc} - t_0^{dc}) \times r_u \times b_1 \\ t_1^U &= t_1^{dc} + 1 + (t_1^{dc} - t_0^{dc}) \times r_u \times b_2 \\ t_0^D &= t_1^{dc} + 1 + (t_1^{dc} - t_0^{dc}) \times r_d \times b_1 \\ t_1^D &= t_1^{dc} + 1 + (t_1^{dc} - t_0^{dc}) \times r_d \times b_2 \end{aligned} \quad (2)$$

where t_0^U and t_1^U are defined as the start and end times of the trading window when it is in an upward trend, t_0^D and t_1^D are defined as the start and end times of the trading window when it is in a downward trend.

Although we have defined the trading window, there still can be hundreds of time points to trade. Trading in multiple price levels is not effective to generate the highest return. The best way to generate maximum profit is to buy as cheap as possible and sell as expensive as possible. We only buy at the price close to the price trough P_{trough} and sell at the price close to the peak price P_{peak}. Therefore, we create a variable: b_3, which is a value within the range [0, 1]. That is, we buy only when the price equals P_{trough} + $P_{\text{trough}} \times (1 - b_3)$ and we sell only when the price equals $P_{\text{peak}} \times b_3$. Here, the P_{peak} and P_{trough} are not referring to the peak (the highest price in the current upward trend) and the trough (the lowest price in the current downward trend) respectively. They are the highest recorded price and the lowest recorded price respectively

Lastly, we create a user-specified parameter Q which allows the trader to control the trading quantity.

2.2.2. Multi-threshold DC trading strategy

Instead of using only a single threshold, multi-threshold DC trading strategy combines the information collected by different threshold values and makes a majority vote on whether to buy or sell at that time point [7]. In Section 2.1, we have mentioned that different threshold values generate DC and OS events at different time points. Small thresholds detect more events, allowing quicker action. Large thresholds detect fewer events, focusing on large price variations. Multi-threshold DC trading strategy tries to combine their advantages and performs trade actions.

Since different threshold values generate DC and OS events at different time points, each threshold recommends different trade actions per time point. To decide the final action, a majority vote is performed. We assign each threshold a weight: W_1, W_2, W_3, …, $W_{N\theta}$, where N_θ is the total number of thresholds. The weight of each threshold is assigned a different value because some thresholds may be more important. Each threshold recommends buy or sell action per time point. If the sum of weights of all thresholds recommending a buy action > the sum of weights of all thresholds recommending a sell action, the system performs buy action, and vice versa. Hold action is performed when $N_\uparrow$ or $N_\downarrow$ is < 0 or the price is not within the range defined by b_3. Details will be shown in Algorithm 2 and Algorithm 3.

Also, the multi-threshold trading strategy can recommend the trading quantity Q_{trade}. If many thresholds recommend a buy (sell) action, Q_{trade} increases.

$$Q_{trade} = (1 + \frac{N_\downarrow}{N_\theta}) \times Q \quad (3a)$$

$$Q_{trade} = (1 + \frac{N_\uparrow}{N_\theta}) \times Q \quad (3b)$$

where $N_\downarrow$ is the number of thresholds that recommending a buy action, $N_\uparrow$ is the number of thresholds that recommending a sell action. Q is the parameter that has been discussed in Section 2.2.1. Algorithm 2 presents the return of multi-threshold trading strategy and Algorithm 3 presents how to perform buy and sell actions. They can also be applied to the single-threshold trading strategy. But there is only one single weight of threshold, we set W_1 = 1 and Q_{trade} = Q. The algorithms in [7] have been amended again given that some parts of the original algorithms are not optimal. In the improved version, t_0^U, t_1^U, t_0^D, t_1^D are only calculated when t is a confirmation point of DC. And trade actions are not performed at the confirmation point of DC. Algorithm 2 and Algorithm 3 are the improved versions of the algorithms.

Algorithm 2 Pseudocode for calculating the return of multi-threshold DC trading strategy

Require: Initialise variables (cash = budget, $Q_{trade} = 0$)
Require: b_1, b_2, b_3, Q and weight of thresholds: W_1, W_2, W_3, …, $W_{N\theta}$
for t = 0; t < dataset_length; t + + **do**
  Initialize variables weights for buy and sell: $W_B = W_S = 0$, number of thresholds that recommending a buy and sell action: $N_\downarrow = N_\uparrow = 0$
  t ← t + 1
  if $p(t) > P_{\text{peak}}$ **then** // $p(t)$ is the current price and P_{peak} is the highest so-far price
    $P_{\text{peak}} \leftarrow p(t)$
  else if $p(t) < P_{\text{trough}}$ **then** //P_{trough} is the lowest so-far price
    $P_{\text{trough}} \leftarrow p(t)$
  end if
  for j = 0, j < N_θ ; j + + **do** //for each threshold
    if t is a confirmation point of DC **then**
      calculate t_0^U, t_1^U, t_0^D, t_1^D as explained in Eq.(2)
      // t_0^U, t_1^U, t_0^D, t_1^D are only updated at confirmation points of DC
    end if
    if t is between the confirmation point of a downward DC and the next confirmation point of an upward DC **then**
      $W_B \leftarrow W_B + W_j$
      if t within range of $[t_0^D, t_1^D]$ **then**
        $N_\downarrow \leftarrow N_\downarrow + 1$
      else
        $N_\downarrow \leftarrow N_\downarrow - 1$
      end if
    else if t is between the confirmation point of an upward DC and the next confirmation point of a downward DC **then**
      $W_S \leftarrow W_S + W_j$
      if t within range of $[t_0^U, t_1^U]$ **then**
        $N_\uparrow \leftarrow N_\uparrow + 1$
      else
        $N_\uparrow \leftarrow N_\uparrow - 1$
      end if
  end for
  if $W_S > W_B$ **then**
    Perform the sell action [See Algorithm 3]
  else if $W_S < W_B$ **then**
    Perform the buy action [See Algorithm 3]
  end if
end for
Wealth ← cash + PFL × $p(t)$
Return ← 100 × (Wealth – budget) / budget

2.3. Optimizing Multi-threshold DC trading strategy via Genetic Algorithm

Genetic algorithm (GA) is a meta heuristic optimization algorithm that applies the Charles Darwin's theory of natural evolution to solve problems. GA starts with a population of randomly generated solutions (chromosomes) and then applies the natural selection process to choose and evolve the fittest individuals while the weakest individuals are discarded among each generation (iteration). A fitness function is defined to evaluate the quality of each individual. The natural selection process is performed based on the quality of individuals. The higher the quality of the individual, the higher the possibility that its genes are passed to the next generation. It guides the search towards promising directions with better solution quality. Besides, the stochastic nature of GA diversifies the search and helps

Algorithm 3 Pseudocode for performing the buy and sell actions

if $W_S > W_B$ **then**
  if $N_\uparrow > 0$ and $p(t) \geq P_{\text{peak}} \times b_3$ **then**
    $Q_{trade} \leftarrow (1 + \frac{N_\uparrow}{N_\theta}) \times Q$
    cash ← cash + $Q_{trade} \times p(t)$
    PFL ← PFL - Q_{trade} // PFL stands for Portfolio, i.e. the amount/quantity for the stock we are currently holding
  else
    Hold
  end if
else if $W_S < W_B$ **then**
  if $N_\downarrow > 0$ and $p(t) \leq P_{\text{trough}} + (P_{\text{trough}} \times (1 - b_3))$ **then**
    $Q_{trade} \leftarrow (1 + \frac{N_\downarrow}{N_\theta}) \times Q$
    if cash > $Q_{trade} \times p(t)$ **then**
      cash ← cash - $Q_{trade} \times p(t)$
      PFL ← PFL + Q_{trade}
    else // Buy as much as you can afford
      $Q_{trade}{}' \leftarrow$ cash ÷ $p(t)$
      cash ← cash - $Q_{trade}{}' \times p(t)$
      PFL ← PFL + $Q_{trade}{}'$
    end if
  else
    Hold
  end if
end if

GA to escape from local optima. It first starts with a population of randomly generated individuals. Then, genetic operators such as crossover and mutation are used to produce offsprings for the new generation from individuals parents with a stochastic selection. GA has been effectively used in many financial problems [12], [13], [14].

In the paper, we construct the GA in the way which Algorithm 4 presents [7]. For each generation, the quality of each individual is evaluated by a fitness function, and then elitism is carried out to choose the best *r* individuals. Next, we use tournament selection to select *(p – r)* individuals, where *p* is the population size. For each of the selected individuals, we perform uniform crossover and uniform mutation. The modified individuals are then combined with the elitism group to form the new population. The next generation starts. The whole process is repeated until the optimal solution is found or the maximum number of generations is reached.

Genetic operators. (a) In elitism, the group of the best individuals is passed to the next generation without modification. This guarantees the quality of individuals in the next generation will not decrease. (b) In tournament selection, which is one of the selection strategies, we randomly pick *k* individuals from the population and choose the best individual among the *k* individuals, and then put it in the selected group. This process is performed *(p – r)* times to select *(p – r)* individuals. (c) In uniform crossover, the *pc* determines whether the chosen pair of individuals undergo uniform crossover in which each pair of genes between the two selected parents are swapped with a probability of 0.5. In uniform mutation, the *pm* determines whether the chosen individuals undergo

Q	b_1	b_2	b_3	W_1	W_2	W_3	W_4	W_5	θ_1	θ_2	θ_3	θ_4	θ_5
10	0.8	1.0	0.9	0.4	0.2	0.3	0.2	0.8	0.01%	0.013%	0.015%	0.018%	0.02%

Fig.3. An Example of a 14-Gene GA Chromosome. The first 4 genes are the threshold values of each threshold respectively. The next 5 genes are their weights respectively. The remaining 5 genes are the parameters: Q, b_1, b_2, b_3 respectively.

Algorithm 4 Pseudocode for performing genetic algorithm [7]

GA (*pop, iter, Fitness, r, k, pc, pm*)
p: population size
iter: max number of iterations
Fitness: fitness function which determines the quality of solutions
r: number of elites selected and passed forward to the next generation without modification
k: tournament selection size
pc: crossover rate
pm: mutation rate
1: Initialize population: $P \leftarrow$ Generate *p* individuals (solutions) randomly
2: Evaluate: **for each** *i* **in** *P*, calculate *Fitness(i)*
3: **While** max fitness or iter not achieved **do**
4: $P_g \leftarrow$ Create *new population* for *generation g*
5: *Elitism*: copy the best *r* individuals from *P* to P_g
6: *Selection*: perform *tournament selection* with size *k* and select *(p – r)* individuals of *P* to add to *selected group*
7: **for each** *individual* **in** *selected group*:
8: perform *uniform crossover* between a pair of *individuals* according to *pc*
9: perform *uniform mutation* on the *individual* according to *pm*
10: Add *individuals* of *selected group* to P_g
11: Update: $P \leftarrow P_g$
12: Evaluate: **for each** *i* **in** *P*, calculate *Fitness(i)*
13: end **while**
14: **Return** the individual with the highest fitness from *P*

uniform mutation in which each gene of the selected parent is mutated to a random domain value with a probability of 0.5.

In Section 2.2, we have established multi-threshold trading strategy, the weight of each threshold is not decided yet. Of course, we can simply assign an equal weight of 1 to each threshold but it is not an effective trading strategy because some thresholds may be more important and they should be assigned higher weights. As a result, we will use genetic algorithm (GA) to evolve each threshold weight and other parameters: Q, b_1, b_2, b_3 that affect the profit of the trading strategy. In this paper, we also try to evolve the threshold values θ.

Representation. Each individual in GA is represented as a string of either binary or numeric values. Each index is regarded as a gene and each gene represents a parameter to be optimized. In this paper, we use numeric values. We try to optimize chromosomes with $4 + N_\theta \times 2$ genes, where N_θ is the total number of thresholds in multi-threshold DC trading strategy. In Fig.3, we can observe the chromosome with 14 genes has 5 thresholds. The first 4 genes are Q, b_1, b_2, b_3 respectively. The next 5 genes are the weights of thresholds. The last 5 genes are the threshold values θ. Genes are initialized and evolved in the specified domain value. Then, GA evolves the genes of the chromosome, and passes the best combination of parameters to the multi-threshold DC trading model to perform trading.

2.4. Optimizing Multi-threshold DC strategy via Virus Spread Optimization

Virus Spread Optimization (VSO) is a recently proposed meta heuristic optimization algorithm that mimics the spread of viruses among hosts [10]. VSO starts with a population of hosts with randomly generated ribonucleic acid (RNA) which represents the solution. Different viral operations – selection, recovery, infection, mutation are performed in each round of iterations to conduct: (a) diversified searches in search space to prevent falling into local optima, and (b) neighborhood searches around the discovered local optima to achieve better solutions. In [10], VSO is evaluated on a total 46 well-known benchmark functions [15], [16], [17], [18], and problems of financial portfolio optimization and SVMs optimization. The results show that VSO outperforms conventional and state-of-the-art heuristic algorithms such as GA [19], DE [20], PSO [21] and CMA-ES [22] in terms of solution fitness, convergence rate, reliability and flexibility.

There are 4 types of hosts: healthy, mild, severe and critical in which mild, severe and critical hosts are infectious hosts. All hosts have their own RNA which is represented as Eq. (4). Each type of infectious host has its own mutation and infection behavior. The mutation rate of critical, severe, mild and healthy hosts is in increasing order while the infection rate of them is in decreasing order. During the optimization process, at each iteration, there is only 1 critical host which represents the best solution obtained so far.

$$X_i = [x_i^1, x_i^2, \dots, x_i^D], X_i \in S \tag{4}$$

where X_i is a vector to denote a possible solution, i is the i^{th} iteration, D is the number of parameters and S is the domain search space

VSO has 6 basic operations: initialization, selection, mutation, infection, recovery and imported infection:

1) Initialization: At the beginning when the number of iterations = 0, similar to GA, a population of hosts with randomly generated RNA which represents a possible solution are initialized as healthy hosts according to Eq. (5). The mutation intensities of mild and severe hosts are also initialized according to Eq. (6) and (7).

$$X_{i=0} = bound_l + rand\,(0, 1) * (bound_u - bound_l) \tag{5}$$

$$intensity_{i=0}^{M} = U\,(bound_l, bound_u) / 10 \tag{6}$$

$$intensity_{i=0}^{S} = 1 / rand\,(0, 1) \tag{7}$$

where $bound_u$ and $bound_l$ represent the upper and lower bounds of domain, *rand* (0, 1) is a random number between 0 and 1.

2) Selection: At each iteration, the host with the best solution is selected as the critical host. The previous critical host is downgraded as a severe host.

3) Mutation: The mutation rate of critical, severe, mild and healthy hosts is in increasing order. Different hosts mutate with the different mechanisms at each iteration:

- *A Healthy Host*
  Since healthy hosts denote bad quality solutions, they mutate by undergoing the initialization as Eq. (8), where *S* is the whole search space and *U* is a function which generates random vectors based on the uniform distribution of *S* to perform global search.

$$X_i = U(S) \tag{8}$$

- *A Mild Host*
  A mild host mutates according to the Eq. (9) and (10), where $intensity_i^M$ is a vector denoting the mutation intensity of that mild host at the i^{th} iteration, $\alpha \in [0, 1]$ and $\gamma \in [1, 2]$ are the scaling factors, $gbest_{i-1}$ is the best solution at the $(i-1)^{th}$ iteration

$$intensity_i^M = \alpha * intensity_{i-1}^M + \gamma * rand\,(0, 1) * (gbest_{i-1} - X_i) \tag{9}$$

$$X_{i+1} = X_i + intensity_i^M \tag{10}$$

- *A Severe Host*
  A severe host mutates according to the Eq. (11) and (12), where $intensity_i^S$ is a scalar denoting the mutation intensity of that severe host at the i^{th} iteration, $\delta_S \in [0, 1]$ is the decay rate, $Gaussian\,(0, intensity_i^S)$ is a Gaussian function with mean = 0 and standard deviation = $intensity_i^S$.

$$intensity_i^S = \delta_S * intensity_{i-1}^S \tag{11}$$

$$X_{i+1} = X_i + Gaussian\,(0, intensity_i^S) * X_i \tag{12}$$

- *A Critical Host*
  The critical host has the RNA of the best solution. To keep the search information of the solution, the critical host does not mutate according to Eq. (13).

$$X_{i+1} = X_i \tag{13}$$

4) Infection: The critical host has the highest infection rate R^C, severe hosts have the second-highest rate R^S, mild hosts have the lowest rate R^M and healthy hosts cannot infect other hosts. At each iteration, every infectious host can contact a certain number *H* (maximum number of contacts) of healthy hosts and perform infection according to their infection rate. After successful infection, the healthy host is infected as a mild host or a severe host depending on $P_{trans,}$ the matrix of transformation probabilities in Eq. (14).

$$P_{trans} = \begin{bmatrix} P_{H\to M}^C & P_{H\to S}^C \\ P_{H\to M}^S & P_{H\to S}^S \\ P_{H\to M}^M & P_{H\to S}^M \end{bmatrix} \tag{14}$$

For example, $P_{H\to M}^C$ is the conditional probability of a healthy host becoming the mild host given that it is infected by a critical host. Becoming a severe host and becoming a mild host is mutually exclusive so the summation of each row of probabilities is 1. So, if $P_{H\to M}^C$ = 0.3, then if $P_{H\to S}^C$ = 0.7. Also, $P_{H\to S}^M$ is always 0.

- When a healthy host is infected as a severe host, the RNA of the infectious host is directly duplicated to the healthy host.
- When a healthy host is infected as a mild host, each value of the RNA of the infectious host is duplicated to the healthy host with a fixed probability of 0.5.

5) Recovery: Since all hosts may be quickly infected and the algorithm falls into local optima, a certain percentage *revPercen*t of the infected hosts is recovered when all hosts are infected. The hosts are sorted by their quality of solutions and the hosts with lower quality of solutions are recovered first. The recovered hosts will reconstruct their RNA according to Eq. (5), (6) and (7). Besides, a downgrading mechanism is used. A severe host becomes a mild host while a mild host becomes a healthy host

6) Imported Infection: An imported heuristic algorithm DE algorithm is used to construct potentially better solutions. It may improve the search performance in some complex problems. However, this operation is not used in this paper.

Representation. Each RNA in VSO is represented the same as the chromosome in GA like Fig.3.

2.5. Experimental setup

2.5.1. Data

[5], [6], [7] only applied DC trading strategies on intraday data in FX markets. To investigate whether the DC trading strategies work on non-high frequency data and other markets, we use datasets with several years' daily closing prices downloaded from Yahoo Finance to test our trading strategies. We use FX currency pairs GBP/JPY and GBP/USD in [7] to compare the results with [7]. Apart from FX market, we also use 4 stocks which are Tracker Fund of Hong Kong (2800.HK), Hong Kong Exchanges and Clearing Limited (0388.HK), Tesla, Inc. (TSLA) and Vanguard S&P 500 ETF (VOO) to test whether the DC trading strategies work on stocks. The period is 17 years for FX data, from 01 January 2004 to 31 December 2020. The period is 10 years for stock data, from 01 January 2011 to 31 December 2020. We use a rolling window, where 10 years' data are used for FX and 4 years' data are used for stocks to train the parameters of the model via heuristic algorithms, and the consecutive year's data is used for testing the returned model. For instance, for

stocks, the 2011 – 2014 data are used to train the model and the 2015 data is used to test the model. Then, the 2012 – 2015 data are used to train the model and the 2016 data is used to test the model.

2.5.2 Algorithmic experimental setup

[7] has demonstrated that multi-threshold DC trading strategy (MDC) outperforms single-threshold DC trading strategy (SDC) in terms of return and stability. Also, MDC with GA optimizing Q, b1, b2, b3 and [W_1, W_2, W_3, …, $W_{N\theta}$] (called DC+GA in [7]) outperforms EDDIE, Buy and Hold, SDC without GA and MDC without GA. It shows that optimizing the DC parameters and weights increases the return. Therefore, we will directly construct MDC without heuristic algorithms, MDC with GA and MDC with VSO. For the MDC without heuristic algorithms, we will use 5 different thresholds: [0.01%, 0.013%, 0.015%, 0.018% and 0.02%], following the threshold values θ of [7]. For the other two models, the threshold values are optimized using their respective heuristic algorithms. For all models, the budget is set to be £500K and the number of DC thresholds is set to be 5 which are the same as [7].

1. Multi-threshold DC trading strategy (MDC)
The 4 parameters: Q, b_1, b_2, b_3 are assigned fixed values: Q = 1, b_1 = 0, b_2 = 1, b_3 = 1. The threshold values are assigned fixed values: [0.01%, 0.013%, 0.015%, 0.018% and 0.02%]. The weights of thresholds are also assigned fixed values: [1, 1, 1, 1, 1]. This setup makes the trading window equals to the period of the whole OS event exactly. Obviously, it is not the optimal setup because these parameters are not optimized by heuristic algorithms. However, it is still important to include the result of this setup to show the importance to evolve the parameters to optimal values.

2. MDC+GA
The 4 parameters: Q, b_1, b_2, b_3, weight of threshold values [W_1, W_2, W_3, W_4, W_5] and DC threshold values [θ_1, θ_2, θ_3, θ_4, θ_5] are constructed as a GA chromosome according to Fig. 3 and optimized via genetic algorithm (GA). Table 1 presents the GA parameters setting.

3. MDC+VSO
The 4 parameters: Q, b_1, b_2, b_3, weight of threshold values [W_1, W_2, W_3, W_4, W_5] and DC threshold values [θ_1, θ_2, θ_3, θ_4, θ_5] are constructed as a virus RNA according to Fig. 3 and optimized via virus spread optimization (VSO). The VSO parameters are set according to the parameter sensitivity analysis of VSO [23]. Table 2 presents the VSO parameters setting.

In GA and VSO, both their fitness is evaluated using the profit return of Algorithm 2. The range of each DC parameter is specified according to Table 3. We set the ranges of each θ to include small thresholds and large thresholds so that the model can analyze the market from different points of view. As mentioned in Section 2.2.2., small thresholds detect more events, allowing quicker action while large thresholds detect fewer events, focusing on large price variations.

Table 1: GA parameters setting

Parameter	Value
Population size	50
Generations	100
Elitism *r*	5
Tournament size *k*	3
Crossover probability	0.9
Mutation probability	0.1

Table 2: VSO parameters setting

Parameter	Value
Population size	50
Iterations	100
R^C	0.8
R^S	0.2
R^M	0.2
$P^C_{H\to S}$	0.8
$P^S_{H\to S}$	0.5
δ_S	0.9
α	0.1
γ	2
revPercent	0.8
H	1

Table 3: range of DC parameters

Parameter	Range
Q	[0, budget / 1000]
b_1	[0, 1]
b_2	[b_1, 1]
b_3	[0, 1]
W_1 *to* $W_{N\theta}$	[0, 1]
θ_1	[0, 100%]
θ_2	[0, 10%]
θ_3	[0, 1%]
θ_4	[0, 0.1%]
θ_5	[0, 0.01%]

3. Experimental Results

3.1. Results

This section presents results for MDC, MDC+GA, MDC+VSO. We also include the results from [7] to make a comparison. [7] demonstrated the DC+GA trading strategy is the best strategy, which used multi-threshold DC trading strategy and evolved only the 4 parameters [Q, b_1, b_2, b_3] and the threshold weights [W_1, W_2, W_3, W_4, W_5] with GA. Since [7] used a year's daily tick data of GBP/JPY to generate daily mean returns and a year's 10-minute interval data of GBP/USD to generate monthly mean returns, we use their daily mean returns and the monthly mean returns to calculate their yearly returns respectively.

Table 4: Average Yearly Returns

	MDC	MDC+GA	MDC+VSO	DC + GA [7]
GBP/JPY	0.02764%	0.79134%	0.42580%	19.141% (tick data)
GBP/USD	0.00058%	0.037711%	-0.030595%	0.035166% (10-min data)
2800.HK	0.0001%	-0.68461%	-0.57892%	/
0388.HK	0.00616%	**11.868%**	-4.2969%	/
TSLA	0.00088%	**24.348%**	**15.720%**	/
VOO	0.00128%	3.9237%	3.0754%	/

Table 5: Statistics of Yearly Returns

MDC+GA

	Mean	SD	Max	Min	Coefficient of Variation
GBP/JPY	0.79134%	1.7884%	3.6028%	-1.5823%	2.2600
GBP/USD	0.037711%	0.15684%	0.38366%	-0.10027%	4.1589
2800.HK	-0.68461%	3.4095%	2.2982%	-6.8969%	-4.9802
0388.HK	11.868%	15.785%	35.118%	-6.5184%	1.3300
TSLA	24.348%	26.930%	70.782%	2.3228%	1.1060
VOO	3.9237%	5.7303%	11.293%	-4.3464%	1.4604

MDC+VSO

	Mean	SD	Max	Min	Coefficient of Variation
GBP/JPY	0.42580%	0.68577%	1.8465%	-0.0076919%	1.6105
GBP/USD	-0.030595%	0.11493%	0.097042%	-0.26983%	-3.7565
2800.HK	-0.57892%	5.6185%	5.1549%	-8.1396%	-9.7051
0388.HK	-4.2969%	16.403%	22.911%	-22.575%	-3.8174
TSLA	15.720%	17.373%	47.781%	3.0533%	1.1051
VOO	3.0754%	7.2223%	14.314%	-7.0622%	2.3484

Table 4 shows the results of the average yearly returns of all stocks and FX currency pairs. Table 5 shows the statistics of their yearly returns. Fig.4 shows the box and whisker plot of their yearly returns using GA or VSO.

We can see that MDC+GA on TSLA yields the best average yearly returns, which is 24.348%. Both MDC+GA and MDC+VSO yield promising returns on some stocks. MDC+GA yields 11.868% and 24.348% on 0388.HK and TSLA respectively while MDC+VSO yields 15.720% on TSLA. However, they occasionally produce negative returns. MDC+GA yields -0.68461% on 2800.HK. MDC+VSO yields -0.57892% and -4.2969% on 2800.HK and 0388.HK respectively.

Compared to stocks, the performance of MDC+GA and MDC+VSO is not promising on FX currency pairs. Most of the time, they yield < 1% returns on FX currency pairs while they can yield > 3% or even > 10% yearly returns on stocks. For MDC without heuristic algorithms, although the returns generated by MDC are all positive, they are very low. Their average yearly returns are all < 0.1%. Generally, the performance of GA is better than that of VSO.

3.2. Discussion

(a) From Table 4, we can see that DC trading strategies can yield promising returns on daily closing price data on stocks. MDC+GA yields 11.868% and 24.348% on 0388.HK and TSLA respectively while MDC+VSO yields 15.720% on TSLA. From Table 5, we can also observe all maximum return entry is a positive value, showing that there was at least one moment the DC trading strategies made a positive return on every stock and FX currency pair. However, the performance is not stable. The DC trading strategies do not yield promising average yearly returns on all stocks.

(b) The results of FX daily closing price data are not promising. One of the possible reasons is that FX currency pairs generally show low percentage changes in the market.

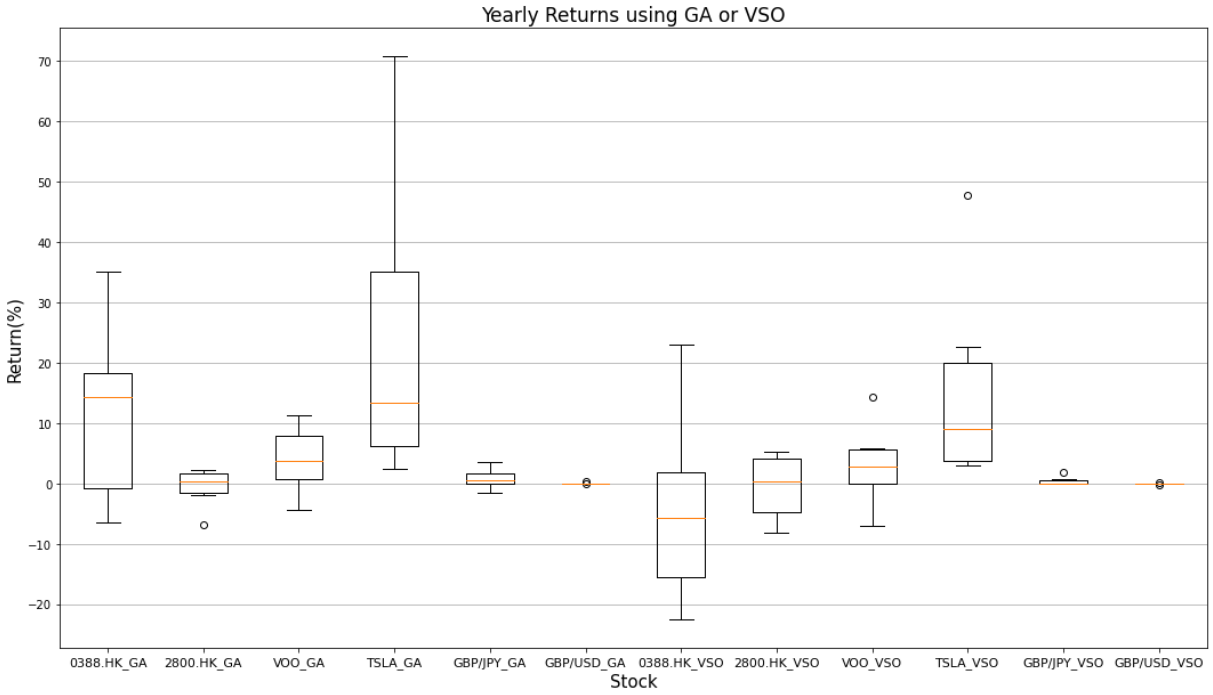

Fig.4. The box and whisker plot of yearly return of all stocks and FX currency pairs using different heuristic algorithms

The daily closing price today and the consecutive day are usually the same. What should also be noted is that the number of daily closing price data is low. There are only 4,437 daily closing price data of GBP/JPY and GBP/USD from 01 January 2004 to 31 December 2020. So, using daily closing price data makes the algorithm hardly capture the price changes in the FX market. On the other hand, this also explains why the tick data of GBP/JPY from [7] yields promising results. The tick data shows detailed change at every tick and there can be 9,000,000 tick data in one single day.

Moreover, the domain space of θ for FX currency pairs should also be set different from that of stocks given that FX currency pairs generally show low percentage changes. In this experiment, we used the same domain space of θ on FX currency pairs and stocks according to Table 3.

(c) It is expected to see that MCD without heuristic algorithms does not make profitable returns because the parameters of MCD are not optimized with heuristic algorithms. In the setup, we just set and fix the threshold values θ, threshold weights, and Q, b_1, b_2, b_3 as simple values.

To summarize whether the deliverable of this paper is achieved: (i) to demonstrate DC trading strategies can produce at least 12% yearly returns; (ii) to evaluate the performance of DC trading strategies on daily closing price data and stocks; (iii) to find out the best heuristic algorithm among GA and VSO for DC trading strategy.

(i): The results show that we have partially met the target. The DC trading strategies yield > 12% yearly returns on some stocks, but it does not apply to all stocks and FX currency pairs.

(ii): DC trading strategies can yield promising returns on daily closing price data of some stocks. However, as stated in (i), it does not apply to all stocks and currency pairs.

(iii): The performance of MDC+GA is generally better than that of MDC+VSO. However, we could hardly say GA is strictly better than VSO for DC trading strategy at the moment. More experiments with different parameter settings should be performed to have a more comprehensive analysis.

3.2.1. Limitations of the Model

The major limitation is the long-running time of the model. The program of multi-threshold DC trading strategy is executed via many for-loop executions. When the number of iterations (generation) of GA and VSO is increased to about 500, the running time takes several hours, which is not convenient for testing the model with different parameter settings. In addition, if data of smaller time intervals such as tick data are used to run the model, the running time is not feasible.

3.2.2. Future Development

(a) *Speed up the model with GPU programming and parallel computing.* GPUs can perform multiple computations simultaneously. This enables the distribution of running processes, which significantly speeds up the optimization process of GA and VSO. Also, we can use parallel computing which breaks down problems into smaller problems that can be executed simultaneously by multiple processors. Both ways can greatly reduce the running time of the trading model.

(b) *Parameters.* We could repeat the experiment with different parameter settings such as the population size and the number of iterations of GA and VSO, GA parameters, VSO parameters, and domain space of DC parameters. For instance, a larger population size and number of iterations

may produce solutions of higher quality.

(c) *Data.* We could repeat the experiment on different datasets such as different currency pairs, different markets or different time intervals. Also, a different combination of the training time and test time can be tried. In this paper, we use a rolling window, where 10 years' data are used for FX currency pairs and 4 years' data are used for stocks to train the parameters of the model via heuristic algorithms, and the consecutive year's data is used for testing the returned model. More or fewer years' data for training and test may lead to a better trading result.

4. Conclusion

To conclude, we improved the DC algorithms and formulas to formulate the multi-threshold DC trading strategy and combined it with genetic algorithm and virus spread optimization for optimizing its parameters. We used these strategies to trade on 2 FX currency pairs and 4 stocks. The results show that we can use daily closing price data to produce promising returns on stocks, but the performance is not stable. The DC trading strategies do not yield promising yearly returns on all stocks and FX currency pairs. We believe that this paper provides more analysis and experimental results on DC trading area for future research.

The limitation of the model is the long-running time. Different strategies like parallel computing and GPU programming can be used to speed up the running time. More work could take place in testing the model with different parameter settings and datasets.

Lastly, the deliverable of this project has been achieved partially in this paper: (i) to demonstrate DC trading strategies can produce at least 12% yearly returns; (ii) to evaluate the performance of DC trading strategies on daily closing price data and stocks; (iii) to find out the best heuristic algorithm among GA and VSO for DC trading strategy. The results show that DC trading strategies can yield promising returns on daily closing price data of some stocks, but it does not apply to all stocks and FX currency pairs. Also, we find that the performance of GA is generally better than that of VSO, but more experiments with different parameter settings are needed to have a comprehensive analysis.

Acknowledgment

I would like to express my gratitude to Dr. Vincent Tam, the supervisor of this project, for his patient guidance and useful advice on this project. I am also grateful for this learning opportunity provided by him. My special thanks are extended to Mr. Zhenglong Li, a Ph.D. student who specializes in researching Directional Changes on financial problems. He has given a lot of technical advice on this research project.

References

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

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

[3] J. Gypteau, F. E. Otero, and M. Kampouridis, "Generating Directional Change Based Trading Strategies with Genetic Programming," Applications of Evolutionary Computation, pp. 267–278, 2015.

[4] A. Bakhach, E. P. Tsang, and H. Jalalian, "Forecasting directional changes in the FX markets," 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016.

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

[6] C. Evans, K. Pappas, and F. Xhafa, "Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation," Mathematical and Computer Modelling, vol. 58, no. 5-6, pp. 1249–1266, 2013.

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

[8] E. Tsang and J. Chen, "Regime Change Detection Using Directional Change Indicators in the Foreign Exchange Market to Chart Brexit," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 3, pp. 185–193, 2018.

[9] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67–82, 1997.

[10] Z. Li and V. Tam, "A Novel Meta-Heuristic Optimization Algorithm Inspired by the Spread of Viruses", 2020.

[11] M. Aloud, E. Tsang, R. Olsen, and A. Dupuis, "A Directional-Change Event Approach for Studying Financial Time Series," Economics: The Open-Access, Open-Assessment E-Journal, vol. 6, no. 2012-36, p. 1, 2012.

[12] C. Evans, K. Pappas, and F. Xhafa, "Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation," Mathematical and Computer Modelling, vol. 58, no. 5-6, pp. 1249–1266, 2013.

[13] M. Kampouridis and F. E. Otero, "Heuristic procedures for improving the predictability of a genetic programming financial forecasting algorithm," Soft Computing, vol. 21, no. 2, pp. 295–310, 2015.

[14] D. E. Goldberg, "Genetic Algorithms and Innovation," The Design of Innovation, pp. 1–9, 2002.

[15] M. Jamil and X. S. Yang, "A literature survey of benchmark functions for global optimisation problems," International Journal of Mathematical Modelling and Numerical Optimisation, vol. 4, no. 2, p. 150, 2013.

[16] A. R. Al-Roomi, "Unconstrained Single-Objective Benchmark Functions Repository", 2015.

[17] M. Molga and C. Smutnicki, “Test functions for optimization needs”, Test functions for optimization needs, 101, 2005.

[18] J. J. Liang, B. Y. Qu, and P. N. Suganthan, “Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization”, Computational Intelligence Labora- 45 tory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 635, 2013.

[19] J. H. Holland, “Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence”, MIT press, 1992.

[20] R. Storn and K. Price, “Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces”, Journal of global optimization, 11(4):341–359, 1997.

[21] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory”, In MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pages 39–43. Ieee.

[22] N. Hansen, S. D. Mller and P. Koumoutsakos, “Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)”, Evolutionary computation, 11(1):1–18, 2003.

[23] Z. Li, V. Tam, L. K. Yeung, “A Study on Parameter Sensitivity Analysis of the Virus Spread Optimization”, 2020