Contents lists available at ScienceDirect

Knowledge-Based Systems

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

# On the design of searching algorithm for parameter plateau in quantitative trading strategies using particle swarm optimization

Jimmy Ming-Tai Wu [a], Wen-Yu Lin [b], Ko-Wei Huang [a], Mu-En Wu [b,*]

[a] National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan
[b] National Taipei University of Technology, Taipei, Taiwan

## ARTICLE INFO

*Keywords:*
Quantitative trading
Trading strategy
Parameter plateau
Optimization algorithm
Uniform design
Particle swarm optimization

## ABSTRACT

Quantitative trading, relying on diverse parameter combinations, is becoming increasingly the norm for trading strategies in financial investments. The performance of these strategies is intricately linked to these parameters. However, the performance on the training set after backtesting does not ensure success on a test set and may lead to overfitting. This study emphasizes enhancing stability and robustness in trading-strategy parameters by introducing a 'parameter plateau.' Traditional brute-force methods for exploring high-dimensional parameter spaces can be intricate and time-consuming. To address this challenge, we present an efficient alternative that identifies stable and robust parameters by configuring parameter plateaus to mitigate overfitting risks. A step-by-step search algorithm is proposed to determine the optimal parameters, leveraging the power of particle-swarm optimization. In continuous, multi-dimensional solution spaces, particle-swarm optimization is invaluable for the swift and effective discovery of the desired solutions. Experiments underscore the substantial influence of the parameter plateau concept on parameter selection, highlighting the pivotal role of particle-swarm optimization in efficiently navigating complex solution spaces and thereby enabling the discovery of stable and profitable trading strategies.

## 1. Introduction

Financial trading encompasses both subjective and quantitative approaches [1,2]. The former is prone to irrational operations driven by human emotions such as profit-seeking, fear of loss, and emotional susceptibility, resulting in decreased profit and increased loss. By contrast, quantitative trading, facilitated by programming, introduces objectivity and aids in developing trading discipline, position control, and money management. In general, the cumulative equity curve of quantitative trading exhibits smoother trends compared to subjective trading, leading to its increasing adoption for continuous and stable returns. A trading strategy consists of indicators, signals, and methods, whereby the various technical indicators are used to generate signals for market entry and exit. These strategies are validated through backtesting with historical data, to ensure alignment with performance expectations.

In addition to quantifying subjective trading strategies through programming, artificial intelligence technologies, including machine learning, are increasingly applied to enhance trading strategies. Advances in information technology have led to the development of novel technologies such as machine learning [3–9], neural networks [10–17], and data mining [18–20]. Notably, there are technology applications in financial markets, such as using genetic algorithms or neural networks to develop investment portfolios [21–24] show the evolution of trading strategies. Furthermore, decision tree classifiers can analyze past stock prices and determine optimal buying or selling timings [25].

In addition to quantifying subjective trading strategies through programming, artificial intelligence technologies, including machine learning, are increasingly applied to enhance trading strategies. Advances in information technology have led to the development of novel technologies such as machine learning [3–9], neural networks [10–17], and data mining [18–20]. Notably, in financial markets, information technology applications such as genetic algorithms or neural networks are used to develop investment portfolios [21–24], demonstrating the evolution of trading strategies. Furthermore, decision tree classifiers can analyze past stock prices and determine optimal buying or selling timings [25].

Numerous combinations of trading strategies exist, with each strategy including market entry and exit methods and various parameter settings. Strategy backtesting performance significantly influences parameter settings, making parameter estimation more challenging as strategies become more complex. Traditional methods such as brute-force searches are time-consuming and may not effectively find stable parameters, often leading to overfitting. Regardless of the approach,

* Corresponding author.
*E-mail address:* mnasia1@gmail.com (M.-E. Wu).

https://doi.org/10.1016/j.knosys.2024.111630
Received 21 April 2022; Received in revised form 29 February 2024; Accepted 8 March 2024
Available online 15 March 2024

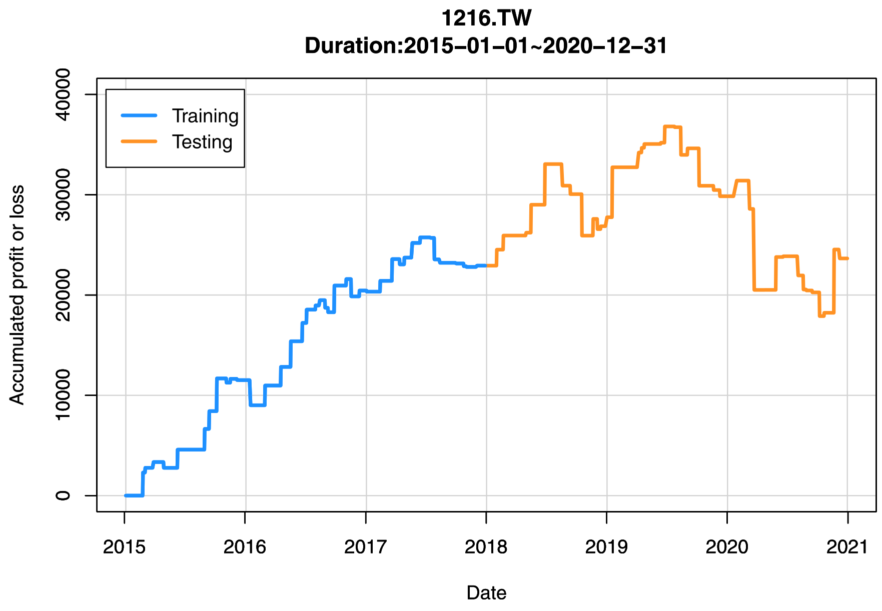


**Fig. 1.** Overfitting of trading strategies.

overfitting remains a challenge in the optimization process [26]. Utilizing parameters from an overfitted model in real trading significantly increases risk [27,28], as the training set data and environment differ from those of the testing set, resulting in poor performance on the latter. To address this issue, this study introduces the concept of a parameter plateau, depicted in Fig. 1, illustrating a key observation. In cases of overfitting, quantitatively derived parameters that yield commendable performance during training exhibit poor performance during testing, indicating a lack of generalizability across diverse scenarios.

A parameter plateau is defined as a continuous block of parameter combinations with performance comparable to other regions and nearby ranges maintaining a certain benchmark. To replace traditional parameter plateau determination, this study provides a definition, proposes a mathematical formula quantifying the degree of a parameter plateau, and introduces a searching algorithm for parameter plateau scores to gradually find optimal parameters. Unlike the time-consuming brute-force search, the study combines Fang's 1980 uniform-design (UD) searching method with the particle evolution process of the particle-swarm optimization (PSO) algorithm for efficient parameter plateau search. The major contributions of this study include:

1. Definition of the parameter plateau and formulation of the plateau score algorithm for searching stable high-performance parameters.
2. Proposal of a mathematical formula to quantify the degree of a parameter plateau, significantly avoiding overfitting.

The remainder of this paper is organized as follows. Section 2 reviews related work and presents preliminaries. Section 3 describes the proposed definition and algorithm in detail. Experimental results are presented and discussed in Section 4. Finally, conclusions are drawn in Section 5.

## 2. Literature review

This section provides a review of related work. Firstly, Sections 2.1 and 2.2 introduce the uniform design (UD) and the particle-swarm optimization (PSO) mechanism, respectively. Section 2.3 then discusses the indicators often used to measure the performance of trading strategies.

### 2.1. Uniform design

Uniform design (UD), proposed by Fang [29] in 1980, operates on the principle of spreading points evenly in an area, aiming to achieve uniform dispersion of experimental points within the test scope, rather than the orderly arrangement in orthogonal designs. Primarily focusing on the uniform distribution, UD ensures that statistical characteristics are evenly spread. Unlike many statistical experiments designed based on model assumptions, UD prioritizes uniformity and orthogonality. Experimental points are designed to avoid row and column repetitions. UD has proven successful in up to 29 dimensions, providing a solution for multi-dimensional search.

Uniform Design is a systematic experimental design methodology that has gained recognition in engineering and product optimization due to its distinctive merits. This approach offers an efficient and well-balanced mechanism for navigating intricate design spaces while minimizing the number of experiments needed. Key advantages include comprehensive coverage of the design space, simplicity in experimental configuration, inherent efficiency in gathering maximal information from minimal trials, and suitability for considering multiple interacting factors. This paper extensively examines the benefits of Uniform Design, emphasizing its potential to substantially enhance the efficiency and efficacy of experimental investigations. To illustrate this potential, UD is integrated with the PSO algorithm to adapt the search strategy for parameter optimization on a plateau.

The systematic experimental methodology of UD has gained recognition in engineering and product optimization because of its distinctive merits. This approach offers an efficient and well-balanced mechanism for navigating intricate design spaces while minimizing the required number of experiments. Key advantages include comprehensive coverage of the design space, simplicity of experimental configuration, inherent efficiency in gathering maximal information from minimal trials, and suitability for considering multiple interacting factors. This paper examines the benefits of UD extensively, emphasizing its potential to substantially enhance the efficiency and efficacy of experimental investigations. To illustrate this potential, UD is integrated with the particle-swarm optimization (PSO) algorithm to adapt the search strategy for parameter optimization on a plateau.

### 2.2. Particle swarm optimization

The PSO algorithm, introduced by Kennedy and Eberhart in 1995 [30,31], draws inspiration from the cooperative behavior observed in animal groups. Examples include birds or fish sharing information and moving in the direction of the group. Individual particles are updated based on the fitness function derived from the environment, propelling them towards better solutions.

The core principle of PSO is to guide individuals towards better positions established by a large number of group members. Individuals in the population move within the solution range according to Eqs. (1) and (2). The position of each particle is influenced by the individual best solution ($pbest$) and the group best solution ($gbest$). Particle positions evolve in each generation, gradually converging to the best solution after several iterations:

$$\begin{aligned} V_i^t = w \cdot V_i^{t-1} &+ C_1 \cdot rand() \cdot (pbest_{t-1} - position_i^{t-1}) \\ &+ C_2 \cdot rand() \cdot (gbest_{t-1} - position_i^{t-1}) \end{aligned} \tag{1}$$

$$position_i^t = position_i^{t-1} + V_i^t \tag{2}$$

In Eq. (1), $w$ represents the inertia weight, multiplied by the previous velocity; $C_1$ is the acceleration weight of the individual best solution; and $C_2$ is the acceleration weight of the group best solution. This equation calculates the velocity of each particle in the population at time $t$. The velocity is influenced by the inertia weight, individual best solution, and group best solution. The terms $C_1 \cdot rand() \cdot (pbest_{t-1} - position_i^{t-1})$ and $C_2 \cdot rand() \cdot (gbest_{t-1} - position_i^{t-1})$ represent the guidance from the individual and group best solutions, respectively. These terms measure the distance between the previous position of a particle and that of the individual or group best solution. The variables $w$, $C_1$, and $C_2$ are hyperparameters of the PSO algorithm.

Jiao et al. [32] proposed an optimized PSO algorithm in 2008 that incorporates a dynamic weight such that the inertia weight decreases with increasing iterations. The function $rand()$ generates a random number between zero and one ([0, 1]). Eq. (2) calculates the position of each particle in the population at time $t$, updating based on the particle position at time $t-1$ and its velocity at time $t$. Due to PSO's tendency to converge to local optima, UD is used to generate a uniformly distributed solution in vector space characterized by independence [33,34]; this aims to achieve global search and improve search time. Zhu [35] combined UD and PSO to maintain particle-swarm diversity and address local optimal solution issues. In our proposed approach, both PSO and UD operate within the same encoding space. PSO is responsible for direct search and exploration, whereas UD designs the fitness function to evaluate the optimality and stability of the solution found by PSO. These two methodologies work concurrently to enhance the overall optimization process. It is essential to note that PSO and UD are not executed sequentially; instead, they function concurrently, with PSO conducting the search and UD shaping the fitness landscape for the search. This design choice leverages the strengths of both PSO and UD to collectively achieve improved optimization results.

### 2.3. Performance measures

Various profit and risk measures in financial markets focus on different aspects. The Sharpe ratio (SR) [36–38] calculates the expected rate of return divided by the standard deviation of daily returns. The standard deviation of daily returns serves as a measure of volatility. SR represents the trade-off between the expected rate of return and risk, with higher values indicating better performance or lower investment risk. The profit factor [39] is determined by dividing the total profit by the absolute value of the total loss, indicating the profit obtained for each dollar lost. A profit factor of one signifies an equal balance between investment profit and loss, whereas a value greater than one indicates a profitable investment.

In the financial market, various profit and risk measures focus on different aspects. The Sharpe ratio (SR) [36–38] calculates the expected rate of return divided by the standard deviation of daily returns. The standard deviation of daily returns serves as a measure of volatility. The SR represents the trade-off between the expected rate of return and risk, with higher values indicating better performance and/or lower investment risk. The profit factor [39] is determined by dividing the total profit by the absolute value of the total loss, which indicates the profit obtained for each dollar lost. A profit factor of one signifies an equal balance between investment profit and loss, whereas a value greater than one indicates a profitable investment.

Maximum drawdown (MDD) [40,41] measures the accumulated loss after reaching a new high, representing the loss from the highest to the lowest point of total assets. This measure serves as a downside risk indicator for a specific period. An MDD value of 100% implies that the previous profit was entirely erased by the loss. Maximum drawdown (MDD) [40,41] measures the accumulated loss after reaching a new high, representing the loss from the highest to the lowest point of the total asset. This measure serves as a downside risk indicator over a specific time period. An MDD value of 100% implies that the previous profit was entirely erased by the loss.

The managed accounts report (MAR) ratio provides a risk-adjusted measure of investment return, commonly used for assessing hedge funds or trading strategies. The MAR ratio is calculated by dividing the annual rate of return by the MDD. A larger MAR ratio indicates a superior annual rate of return or a smaller MDD. Offering a more comprehensive evaluation of trading strategy performance, the MAR ratio considers the average over the entire time interval, without determining the ratio for each specific time interval.

## 3. Parameter plateau and related searching algorithms

In Section 3.1, a measure function is defined and proposed for parameter plateaus, to evaluate the level of stability and quality of certain parameters. Section 3.2 introduces a searching algorithm developed to discover the optimal parameters within parameter plateaus, utilizing a brute-force search approach which is inefficient. To address this issue, the uniform design search method is integrated with PSO to find a nearly optimal solution and enhance efficiency.

### 3.1. Preliminaries and problem statement

Geographically, a plateau is defined as a relatively flat, extensive land area sharply raised above the surrounding area. A parameter plateau is thereby intuitively defined as a contiguous area of many highly similar parameter combinations exhibiting outstanding performance. Therefore, in adjusting a set of parameters for a trading strategy, the new parameter combination is expected to maintain a similar performance level. Thus, a set of parameters in a parameter plateau can be flexibly adjusted as needed without greatly affecting performance. By contrast, on a parameter island, the performance of the parameters in a small area is excellent; however, the performance of nearby parameters is greatly reduced. Even though a satisfactory combination of parameters can be found in the case of parameter islands, the parameters cannot be flexibly adjusted without affecting performance. Thus, a completely new parameter combination should be determined. In terms of machine learning, the performance of a trading strategy despite being outstanding on the training set is sometimes significantly degraded on the testing set. Selecting a solution on a parameter island may also cause overfitting in the training set. Ideally, large performance differences between the training and testing sets should be avoided. Hence, if the proposed framework could select a solution on a parameter plateau, the chance of overfitting would be greatly reduced. A parameter plateau is shown in Fig. 2.

In Fig. 2, the $x$-axis corresponds to the possible parameter combination (solution) set $P = \{p_1, p_2, p_3, \dots\}$, whereas the $y$-axis corresponds to the values of a certain performance measure, denoted as $Perf$. It represents the evaluated performance based on users' definitions. A parameter plateau may involve either continuous or discrete values. In the solution space, there are $N_k$ choices for each parameter, and there are infinitely many choices for $N_k$, which is a positive integer. Due to

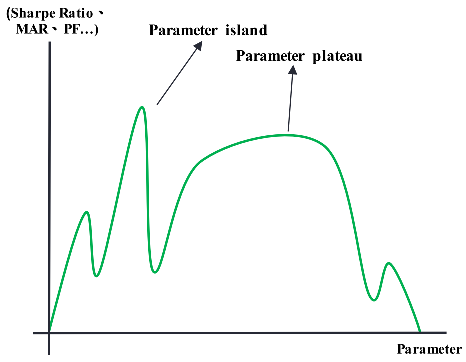


**Fig. 2.** Parameter plateau.

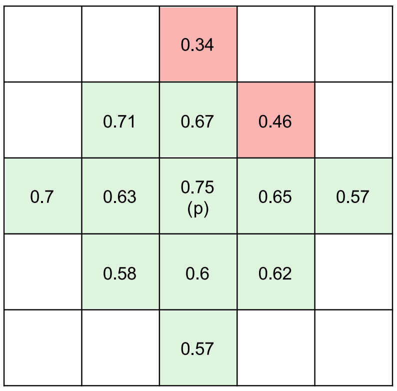


**Fig. 3.** Example of plateau score algorithm.

the limitations of the evaluation algorithm, assume discrete parameters. A trading strategy has discrete indicators, such as moving averages, and continuous indicators, such as KD and RSI. In this case, consider discrete parameters. Continuous parameters can also be converted to discrete to define the distance in the parameter combination space.

In Fig. 2, the $x$-axis corresponds to the possible parameter combination (solution) set $P = \{p_1, p_2, p_3, \dots\}$, whereas the $y$-axis corresponds to the values of a certain performance measure, denoted as $Perf$, which represents the performance evaluated based on user definitions. A parameter plateau may involve either continuous or discrete values. In the solution space, there are $N_k$ choices for each parameter and there are infinitely many choices for $Nk$, which is a positive integer. Due to the limitations of the evaluation algorithm, discrete parameters were assumed. A trading strategy has discrete indicators, such as moving averages, and continuous indicators, such as the stochastic oscillator displaying the %K and %D (KD) lines and the relative strength index (RSI). Continuous parameters can also be converted to discrete when defining the distance in the parameter combination space.

More specifically, each combination of $P_n$ can be encoded and described as follows: $P_n = (P_{n1}, P_{n2}, \dots, P_{nk})$, where $P_{n1}$ is the first type of parameter, with $N_1$ choices in total, encoded as $\{1, 2, \dots, N_1\}$, that is, $P_{n1} \in \{1, 2, \dots, N_1\}$. The variable $P_{n2}$ is the second type of parameter, with $N_2$ choices in total, encoded as $\{1, 2, \dots, N_2\}$, that is, $P_{n2} \in \{1, 2, \dots, N_2\}$. Finally, $P_{nk}$ is the $K_{th}$ type of parameter, with $N_k$ choices in total, encoded as $\{1, 2, \dots, N_k\}$, that is, $P_{nk} \in \{1, 2, \dots, N_k\}$.

In this study, all possible parameter combinations are regarded as a space, and a metric is defined on this space. That is, these parameter combinations are considered as points in space, whereby a distance between any two types of points (parameter combinations) is defined. This distance is called the parameter distance, denoted as $\|\cdot\|_q$, and defined as

$$\|P_m - P_n\|_q = \sum_{k=1}^{K} |P_{mk} - P_{nk}|. \tag{3}$$

The evaluation value algorithm for evaluating the degree of a plateau is expressed as follows:

$$PS_{n,\alpha}(p) = \frac{\#\{x| \ \|x-p\| < n, \text{Perf.}(x) > \alpha\}}{\#\{x| \ \|x-p\| < n\}}, \tag{4}$$

where $x$ represents all the parameters surrounding point $p$ within a step range of $n$. $\text{Perf.}(x) > \alpha$ indicates that the performance indicator of $x$ should be above threshold $\alpha$. The denominator $\#\{x| \ \|x-p\| < n\}$ is the total number of $x$ within a range of $n$ steps surrounding point $p$, and the numerator is the total number of $x$ satisfying the conditions. Using this formula, the plateau score of all parameters can be obtained.

Based on the definition of the parameter plateau, not only does the performance of the parameters on the plateau improve, but the performance of the surrounding parameters is also comparable. An evaluation method and algorithm for parameter plateaus are defined to measure the degree of the parameter plateau. The proposed algorithm assigned scores according to the completeness of a plateau. After selecting a point, the algorithm calculates the performance indicator of the parameters surrounding this point; if this indicator is above a preset threshold, it is included in the plateau score. In this section, an algorithm is defined that measures the parameter plateau degree to replace the traditional method of determining whether a parameter is a plateau. The quality of the parameter is evaluated, and the degree of the parameter plateau is expressed as a score within $[0, 1]$. If the plateau score is closer to one, the parameter plateau is more complete, and investors can trust the performance of the strategy based on this parameter. By contrast, if the plateau score of the parameter is closer to zero, the parameter is not a plateau or the plateau is not complete. Even if this parameter leads to excellent strategy performance, the probability that it is a parameter island is very high.

Let us consider an example to explain the plateau score algorithm more clearly, which is shown in Fig. 3. The variable $x$ represents all the parameter points instead of $p$; $n$ is set to two; and $\alpha$ is set to 0.5. The formula $\#\{x| \ \|x-p\| < n\}$ is 13 because there are 13 points within $n$ steps from $p$. The numerator $\#\{x| \ \|x-p\| < n, \text{Perf.}(x) > \alpha\}$ is 11 because there are 11 points within $n$ steps from $p$ above threshold $\alpha$. Thus, the plateau score of point $p$ is $PS_{n,\alpha}(p) = \frac{11}{13}$.

The above calculation of the plateau score is quite intuitive; however, it is the most basic plateau score algorithm. The weighted plateau score algorithm is as follows: intuitively, the parameters closer to the center point have a greater effect on the plateau shape, and thus they should be assigned more weight, as shown in Fig. 4.

In Fig. 4, the red blocks represent parameters with poor performance, whereas the green blocks represent parameters with better performance. In calculating the plateau score of point $p$ in the figure according to the above simple plateau score algorithm, the plateau scores calculated for the left and right figures are both $\frac{11}{13}$. The left one shows that farther parameters have poorer performance, whereas the right one shows that closer parameters have poorer performance. The parameter plateau on the left is relatively strong and has high confidence. Accordingly, a weighted plateau score algorithm is proposed as follows:

$$PPScore = \sum_{r=0}^{n} \text{Ball}_{n,\alpha}(p). \tag{5}$$

In Eq. (5), $r$ represents the parameters in each step within a range of $n$ steps from point $p$, and $\alpha$ represents the threshold of the performance indicator of the parameters within this range. The above formula is the weighted plateau score. If a parameter is closer to point $p$, it is included in the summation more than once, which aligns with the original intention. A parameter closer to $p$ is more important, and its influence on the plateau degree is greater. An example of a weighted plateau-score algorithm is shown in Fig. 5.

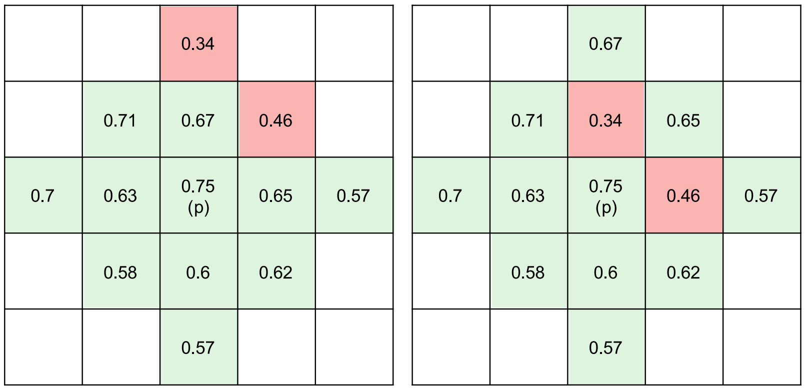
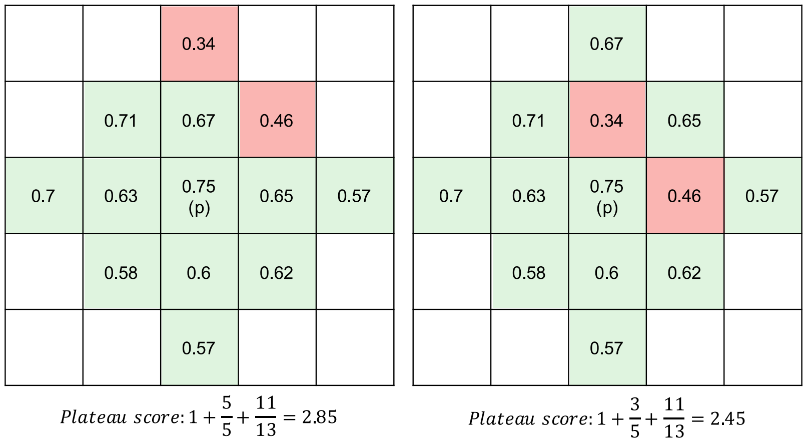


**Fig. 4.** Weighted plateau score.



$$Plateau\ score: 1 + \frac{5}{5} + \frac{11}{13} = 2.85 \qquad Plateau\ score: 1 + \frac{3}{5} + \frac{11}{13} = 2.45$$

**Fig. 5.** Example of weighted plateau score algorithm.

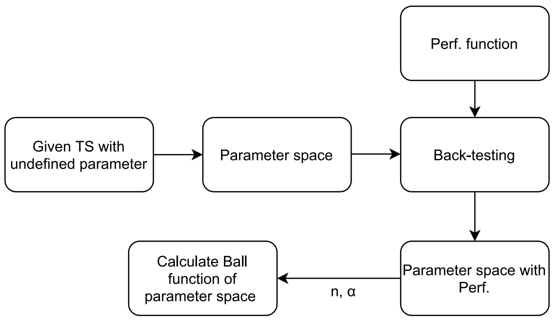

Perf. function
Given TS with undefined parameter
Parameter space
Back-testing
Calculate Ball function of parameter space
n, α
Parameter space with Perf.

**Fig. 6.** Mechanism of the plateau score search algorithm.

The value in Fig. 5 is assumed to be the SR of each parameter. The threshold of the performance indicator of the parameter is set to 0.5, and the plateau score is calculated within a range of two steps. Thus, the calculated plateau score in the left figure is 2.85, and the calculated plateau score in the right figure is 2.45. According to the weighted plateau-score algorithm, if the parameters closer to the center have poor performance, they have a greater impact on the plateau score.

### *3.2. Proposed parameter plateau efficient searching algorithm*

In the parameter plateau searching algorithm presented in this section, a detailed explanation is provided for the calculation and selection of plateau scores. The mechanism of the plateau score search algorithm is illustrated in Fig. 6.

In step one, a trading strategy (TS) is specified. As an example, the common moving average strategy is used. The moving average is calculated from the close price of a stock over a certain period. The moving average strategy is described as follows: the entry condition requires that the close price be higher than the moving average of the entry; the overweight strategy is that the close price is higher than the overweight moving average; the exit strategy is that the close price is lower than the exit moving average, and the entry and exit filters add one more condition to the entry and exit strategies, respectively. Once the parameter range is set, all the parameters within the range are combined into a parameter space. Subsequently, backtesting is performed for these parameters, and the results are recorded as $Perf.$ Subsequently, $n$ (the parameter plateau range) and $\alpha$ (parameter performance threshold) are input. Finally, the plateau score of each parameter in parameter space is calculated.

The proposed approach maintains flexibility in accommodating various quantitative trading strategies without imposing any restrictions on their choice. Let each trading strategy consist of ”n” input parameters, each with its respective value range. A single particle in PSO is encoded directly and corresponds to the “$n$” parameter values, $(x_1, x_2, \dots, x_n)$, which effectively translates the parameter space into the PSO solution space. These encoded particles serve as inputs for the backtesting function ($Perf.$) evaluates the performance of the trading strategy. This encoding method ensures that the PSO algorithm operates directly within the space of trading strategy parameters, facilitating parameter space exploration without imposing any specific constraints on the structure or characteristics of the trading strategy, thus enabling efficient optimization of quantitative trading strategies across diverse domains.

**Table 1**
Parameter complexity.

| Dimensional | Parameter | Count |
|---|---|---|
| 1 | Vol1 | 50 |
| 2 | Vol2 | 50 |
| 3 | Entry MA | 20 |
| 4 | Overweight MA | 20 |
| 5 | Exit MA | 20 |
| 6 | Capital | 6 |

**Algorithm 1** Data preprocessing

```
Input: strategy (A quantitative trading strategy with parameters.).
Output: undefinedPara is the undefined parameter range.
1: Calculate the Parameter Space of all undefined parameter range
```

The data preprocessing for the plateau-score calculation is outlined in Algorithm 1. The input to this algorithm is a trading strategy, and the output, obtained through data preprocessing, is a parameter space combination encompassing all the parameters in the trading strategy.

**Algorithm 2** Plateau score searching algorithm

```
Input: Parameter space $paraSpace$, range of parameter plateau $n$,
    threshold of parameter performance $\alpha$.
Output: Plateau score of all parameters $PlateauScore$
1: for $x$ in $paraSpace$ do
2:     $ParaPerf$ = Performance(x)
3: end for
4: for $s$ in $ParaPerf$ do
5:     $PlateauScore$ = Ball(s,n,$\alpha$)
6: end for
```

The plateau score searching algorithm is presented in Algorithm 2. In Line 2, the performance indicator of all parameters in parameter space is calculated. Line 5 calculates the plateau score of all parameters in parameter space. Finally, the plateau score of all parameters is output. According to the definition of the parameter plateau score, the most basic approach is searching for each parameter, which is quite complicated and time-consuming. In view of this, PSO is applied to parameter plateau searching, to replace the traditional brute-force search method.

As the dimension and search range become higher, the range that the brute-force search method can handle within a reasonable time may be exceeded. In the brute-force search method, each additional search parameter requires an additional for-loop. An example in six dimensions is shown in Table 1.

The estimated total number of parameters to be calculated is $50 \times 50 \times 20 \times 20 \times 20 \times 6 = 120000000$, requiring that more than 100 million parameters be calculated. First, the performance of each parameter is calculated. Then, all parameters within n steps from each parameter are obtained, and the respective plateau score is calculated. In this study, a method is proposed to reduce the search time of the traditional brute-force method. The UD method can be used to improve the search location, and then PSO is used to improve the search efficiency. The characteristics of UD require that the limited number of scatter points be determined. Uniform designs are characterized by evenly dispersed scatters, and if the position of the scatter is not exactly where the plateau score is maximized, then the parameter may not be suitable. As the initial population is randomly generated by the traditional PSO algorithm, it may be excessively concentrated, making it difficult to determine an appropriate initial position. In view of this, the UD searching method and the PSO algorithm are combined, and the position of the UD scatters is used as the position of the initial population of the PSO algorithm, instead of randomly generating the initial population. Thus, the position of the initial population is more evenly dispersed, and the algorithm evolves more efficiently.

The PSO algorithm combines group wisdom and the information shared by the population to attain a better solution. This is suitable

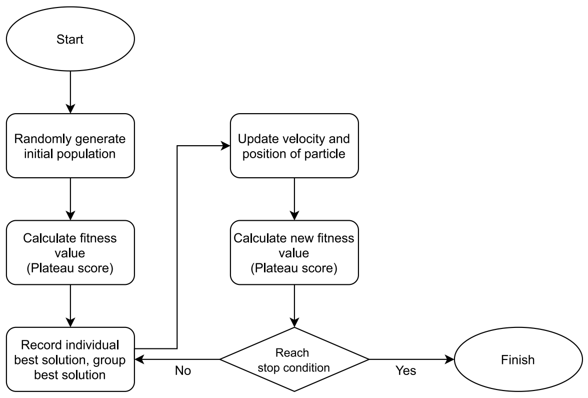

Start
Randomly generate initial population
Calculate fitness value (Plateau score)
Record individual best solution, group best solution
Update velocity and position of particle
Calculate new fitness value (Plateau score)
Reach stop condition
No
Yes
Finish

**Fig. 7.** Flowchart of PSO algorithm for plateau score searching.

for parameter plateau searching because a plateau has continuous and relatively complete large uplift areas. Moreover, in the evolution of the PSO algorithm, when the particles find a raised area that is not the highest, they will approach this area, and after multiple iterations of evolutionary adjustment, they converge to the plateau. The flowchart of the PSO algorithm for plateau score searching is shown in Fig. 7.

First, an initial population is randomly generated. In the application of the parameter plateau score, the value of the fitness function is the value of the plateau score. Then, the individual best solution and the group best solution are recorded, and evolution begins. Under the guidance of the inertia weight, the individual best solution, group best solution, velocity, and position of the particle are updated. Subsequently, the value of the fitness function at the new position is calculated. The stopping condition of the algorithm is that the fitness values of the particles converge to the same position or the iterations in the evolution process reach the maximum value. If this condition is satisfied, then the final result is output; otherwise, the iterative evolution continues. Pseudocode of the proposed PSO algorithm applied to the plateau score is shown in Algorithm 3.

**Algorithm 3** PSO for parameter plateau searching algorithm

```
Input: Parameter space $paraSpace$, range of parameter plateau $n$,
    threshold of parameter performance $\alpha$, hyperparameters of PSO
    algorithm $w$, $C_1$, $C_2$, size of initial population $nPop$, maximum value
    of iteration $MaxIt$.
Output: group best solution of PSO algorithm $gbest$

1: Initialize the hyperparameters of PSO as $w$, $C_1$, $C_2$, $nPop$, $MaxIt$
2: Initialize the swarm of particles as $EachParticle$
3: $fitnessvalue$ = PPScore($paraSpace$, $n$, $\alpha$)
4: Update $pbest$
5: Update $gbest$
6: while the stopping condition is not satisfied do
7:     for $p$ in $EachParticle$ do
8:         Update the velocity of the particle
9:         Update the position of the particle
10:        $fitnessvalue$ = PPScore($paraSpace$, $n$, $\alpha$)
11:        Update $pbest$ if required
12:        Update $gbest$ if required
13:    end for
14: end while
```

Algorithm 3 presents the pseudocode of the PSO algorithm applied to the plateau score. Initially, the algorithm calculates the position and the fitness value of the initial population and records the individual best and group best solutions. Subsequently, evolution begins. The velocity, position, and new plateau score of each particle are updated through evolution. If the stopping condition is satisfied, the final $gbest$ is output, which is a parameter with a relatively high plateau score.

Utilizing the soft-computing PSO as an efficient searching method for parameter plateaus can significantly reduce the search time. However, the opportunity cost of increasing search efficiency is that the plateau parameter may not be found as effectively as by the traditional calculation. The purpose of parameter search in this study is to find parameters with high performance and stability suitable for a trading strategy. We believe that it is relatively important to efficiently find parameters with higher plateau scores even if it is at the expense of sacrificing some profit.

### *3.3. Time complexity analysis and integration with the proposed framework*

The time complexity of the algorithm that combines particle-swarm optimization and uniform design is intricately tied to the nature of both PSO and UD, with a primary focus on the computational effort involved in evaluating candidate solutions through the fitness function. It is important to recognize that the overall time complexity of this algorithm largely depends on the number of fitness function evaluations, denoted as "$n$".

In PSO, the primary computational cost comes from the repeated evaluation of candidate solutions in the search space. Similarly, UD involves the generation and evaluation of a set of uniformly distributed sample points within the design space. The time complexity of UD itself is primarily determined by the number of these sample points. Assuming that the time complexity of the fitness function is not greater than $O(n)$, which is a reasonable and often met criterion, the overall time complexity of the combined algorithm can be approximated as $O(n)$; this is because the number of fitness function evaluations "$n$" becomes the driving factor in determining the algorithm's time complexity.

In practice, this means that the algorithm's efficiency scales with the number of fitness function evaluations. Therefore, for applications in which the fitness function time complexity remains within a reasonable range, the combined PSO-UD algorithm exhibits a time complexity that is predominantly linear, which is directly tied to the number of iterations or evaluations performed.

Furthermore, it is important to highlight that the proposed algorithm has been designed to be integrated with a quantitative trading strategy. In the context of quantitative trading, the computational time associated with trading strategies is typically treated as a constant factor, leading to a time complexity of $O(1)$. Once developed, quantitative trading strategies often require real-time decision-making, and their computational requirements remain relatively stable across different market conditions and trading scenarios.

As a result, when the combined PSO-UD algorithm is used in conjunction with a quantitative trading strategy, the overall time complexity is still primarily determined by the fitness function evaluation ($O(n)$), whereas the computational demands of the trading strategy remain constant ($O(1)$). This ensures that the algorithm's time complexity remains tractable and well-suited for practical quantitative trading applications, with focus on rapid and efficient decision-making in dynamic market environments.

## 4. Experimental results

### *4.1. Proof of the validity of the parameter plateau definition*

To justify the proposed parameter plateau definition, we conducted experiments encompassing a diverse array of stocks and strategies. Specifically, we selected two trending trading strategies and two contrarian trading strategies.

In trending strategies, investors opt for long positions during rising stock prices and short positions during bear markets, driven by the belief that the current trend will persist for a certain period and present an opportunity for profitable outcomes. On the other hand, contrarian trading strategies involve taking short positions during rising prices and long positions during falling prices, anticipating a reversal in the current trend.

The parameters of the volatility filter discussed in this section are as follows: the volatility value is determined by the standard deviation of the close price over the previous 20 days, denoted as $vol$. A trade is initiated when the standard deviation of the close price of the first 20 days is less than $vol$, meeting both entry and volatility filter conditions.

For this experiment, the training set spans from January 1, 2010, to December 31, 2016, and the testing set covers January 1, 2017, to December 31, 2019. Following the plateau score definition, we set $n$ to 5 to determine the parameter plateau range, with the parameter performance indicator denoted as $Perf$. using the Sharpe ratio SR and a threshold set to 0.5.

The key performance indicators considered included SR, profit factor (PF), MDD, annual return (AR), and MAR ratio. The results are presented through five types of figures: a heat map illustrating parameter performance, a heat map indicating plateau scores, a comparison of backtesting performance between the training and testing sets, an accumulated profit or loss curve, and a performance comparison of plateau parameters between the training and testing sets.

In the heat map of parameter performance, a darker color signifies higher indicator performance. Similarly, a darker color in the heat map of plateau scores indicates a more stable plateau. In trading, selecting parameters from the plateau offers a high likelihood of achieving performance comparable to or even better than that on the training set. The figure depicting the comparison of backtesting performance between the training and testing sets illustrates the stability of the parameters in the plateau.

The "Trending-Bollinger Bands" strategy is outlined as follows. Initial capital stands at one million, with each entry unit equivalent to the entire capital. Bollinger bands are configured with a standard deviation set to two. Entry conditions require that the close price standard deviation over the previous 20 days is less than $vol$, and the close price surpasses the Bollinger up band price of $D$. In such cases, a long position is initiated. All positions are exited when the close price falls below the $D$-day Bollinger down band price. The parameters that are sought include a close price standard deviation of less than $vol$ over the previous 20 days and a period $D$ for the Bollinger midline.

The heat map of parameter performance (Fig. 8) depicts the indicator performance, with color intensity indicating the level. However, a darker shade in this map may also suggest a parameter island. To select stable parameters, cross-referencing is performed with the heat map of plateau scores.

In the heat map of plateau scores (Fig. 8), a darker shade signifies a higher score, such as the SR score. This aligns with the figure comparing the backtesting performance between the training and testing sets (Fig. 8), showing minor differences between the two sets. The scatter plot in this figure indicates that points with lower plateau scores are more likely to exhibit significant differences between the testing and training sets.

Fig. 9 illustrates the accumulated profit or loss curve for all parameters in the search range. The initial lines labeled $A$, $B$, and $C$ represent the SRs of the training set, the weighted plateau scores, and the SRs of the testing set, respectively.

The gray background lines in Fig. 9 represent the accumulated profit or loss of each parameter within the specified parameter range. Line $A$ depicts the accumulated profit or loss of the parameter with the highest plateau score, displaying strong performance on the training set.

Adhering to this definition, opting for a parameter from the parameter plateau ensures a stable choice with a high probability, even if its performance is not the absolute best on the training set. The parameter indicated by line $B$ shows the best performance on the training set and was selected based solely on performance. However, the testing set performance of line $B$ is not as robust as that of line $A$ because of the lower plateau score.

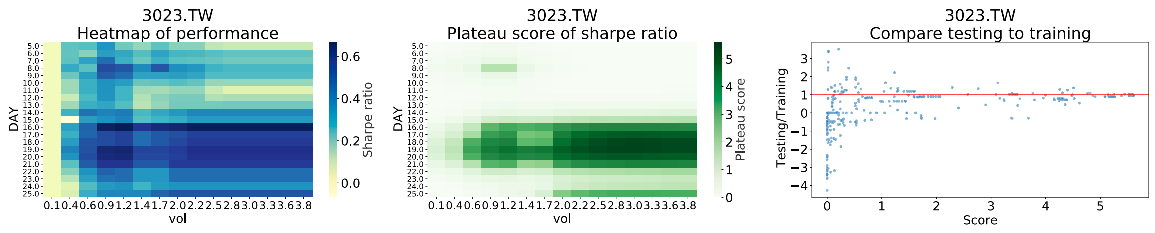


**Fig. 8.** Various experimental results related to the parameter plateau of "Trending-Bollinger Bands" are presented. The left figure displays the heat map depicting SR performance; the middle figure illustrates the SR plateau score; and the right figure provides a performance comparison between testing and training.

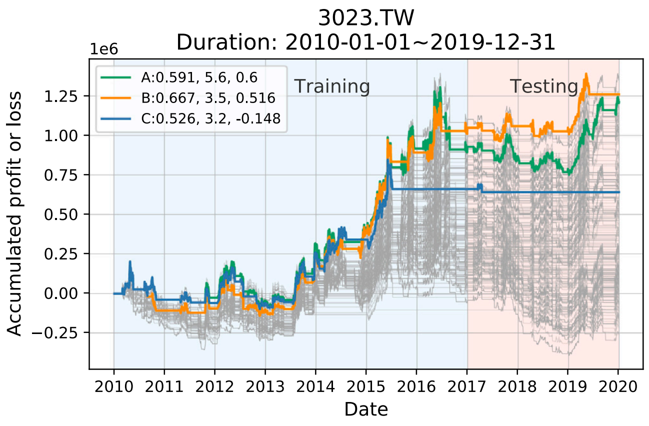


**Fig. 9.** Accumulated profit or loss for "Trending-Bollinger Bands".

**Table 2**
Performance on training and testing sets of "Trending-Bollinger Bands".

| | | Plateau score | Sharpe ratio | | MAR | | Profit factor | |
|---|---|---|---|---|---|---|---|---|
| | | | Train | Test | Train | Test | Train | Test |
| 3023.TW | A | 5.6 | 0.591 | 0.6 | 0.411 | 0.361 | 1.129 | 1.117 |
| | B | 3.5 | 0.667 | 0.516 | 0.379 | 0.488 | 1.183 | 1.132 |
| | C | 3.2 | 0.526 | −0.148 | 0.299 | −0.132 | 1.147 | 0.835 |
| 2207.TW | A | 6 | 0.704 | 1.873 | 0.472 | 2.559 | 1.108 | 1.71 |
| | B | 4.89 | 0.956 | −0.102 | 0.923 | −0.098 | 1.374 | 0.923 |
| | C | 4.87 | 0.925 | −0.352 | 0.794 | −0.188 | 1.428 | 0.812 |

The line $C$ in Fig. 9, does not exhibit poor performance on the training set but shows suboptimal performance on the testing set, possibly due to a lower plateau score. Parameters with higher plateau scores are considered stable and reduce the likelihood of overfitting. Table 2 provides an overview of the training and testing set performance for multiple stock commodities using the "Trending-Bollinger Bands" strategy.

The "Trending-Record High" strategy is implemented as follows. The initial capital is one million, and each entry unit represents the entire capital. Market entry occurs when the standard deviation of the close price over the previous 20 days is less than $vol$, and the close price surpasses the record high price of $D$. A long position is initiated under these conditions. All positions are exited when the close price falls below the record low price over the last $D$ days. The parameters to be explored involve a close price standard deviation of less than $vol$ over the previous 20 days along with period $D$ of record high or record low prices.

The experimental results of the "Trending-Record High" strategy are depicted in Fig. 10, which compares testing and training performance. The figure illustrates that parameters with higher plateau scores tend to have testing and training set performance ratios approaching one.

In Fig. 11, the gray background lines represent the accumulated profit or loss of each parameter within the parameter range. Line $A$ reflects the accumulated profit or loss of the parameter with the highest

**Table 3**
Training and testing set performance of "Trending-Record High".

| | | Plateau score | Sharpe ratio | | MAR | | Profit factor | |
|---|---|---|---|---|---|---|---|---|
| | | | Train | Test | Train | Test | Train | Test |
| 3406.TW | A | 6 | 1.238 | 1.487 | 0.918 | 1.962 | 1.269 | 1.401 |
| | B | 5.7 | 1.565 | 0.999 | 1.559 | 0.905 | 1.357 | 1.452 |
| | C | 4.6 | 1.324 | −0.753 | 1.313 | −0.276 | 1.31 | 0.332 |
| 2912.TW | A | 4.45 | 0.605 | 0.921 | 0.416 | 1.962 | 1.269 | 1.401 |
| | B | 4.2 | 0.695 | 0.704 | 0.486 | 0.905 | 1.357 | 1.452 |
| | C | 1.26 | 0.495 | −0.267 | 0.259 | −0.276 | 1.31 | 0.332 |

**Table 4**
Training and testing set performance of "Contrarian-Record Low".

| | | Plateau score | Sharpe ratio | | MAR | | Profit factor | |
|---|---|---|---|---|---|---|---|---|
| | | | Train | Test | Train | Test | Train | Test |
| 1216.TW | A | 5.8 | 0.602 | 0.785 | 0.485 | 0.774 | 1.135 | 1.201 |
| | B | 4.2 | 0.7 | 0.221 | 0.536 | 0.107 | 1.162 | 1.054 |
| | C | 1.9 | 0.47 | −0.039 | 0.357 | −0.042 | 1.133 | 0.966 |
| 2345.TW | A | 2.09 | 0.542 | 1.638 | 0.226 | 3.802 | 1.14 | 2.53 |
| | B | 1.5 | 0.623 | 0 | 3.211 | 0 | 2.7 | 0 |
| | C | 0.6 | 0.357 | −0.16 | 0.119 | −0.131 | 1.071 | 0.9 |

plateau score, demonstrating stable performance despite not being the best on the training set. The selection of this parameter in real trading achieves good and stable performance attributed to its superior plateau score.

Line $B$ exhibits the highest performance on the training set; however, its plateau score is not particularly high. Consequently, there is a significant performance gap on the testing set compared to the training set, indicating a lack of stability. Traditional parameter selection based on training set performance may lead to disappointing testing set results.

Line $C$ outperforms line $A$ on the training set; but its plateau score is very low. The backtesting results reveal poor performance on the testing set, with a negative profit for this parameter. Thus, parameters with high plateau scores are deemed stable, effectively avoiding parameter islands. Table 3 provides the performance on the training and testing set of multiple stock commodities for the "Trending-Record High" strategy.

The "Contrarian-Record Low" strategy is outlined as follows. The initial capital is one million, and each entry unit is the entire capital. To enter the market, the strategy requires that the standard deviation of the close price of the previous 20 days be less than $vol$ and the close price be lower than the record low price of $D$. A long position is then initiated. All positions are exited when the close price surpasses the record high price over $D$ days. The parameters to be searched are those with a close price standard deviation of less than $vol$ over the previous 20 days or period $D$ of the record high or record low prices.

The experimental results for the "Contrarian-Record Low" strategy are depicted in Fig. 12, comparing the testing and training performance. Parameters with higher plateau scores tend to converge on the testing and training sets. The gray background lines in Fig. 13 represent the accumulated profit or loss of each parameter within the parameter

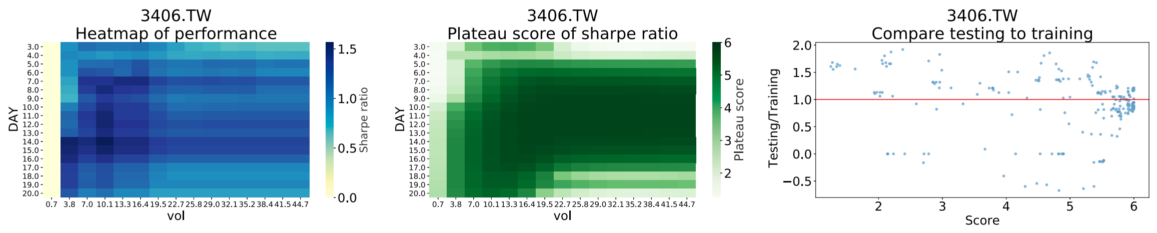


**Fig. 10.** Various experimental results regarding the parameter plateau for "Trending-Record High". The left figure is the heat map of SR performance; the middle figure is the SR plateau score; and the right figure compares the performance between testing and training.

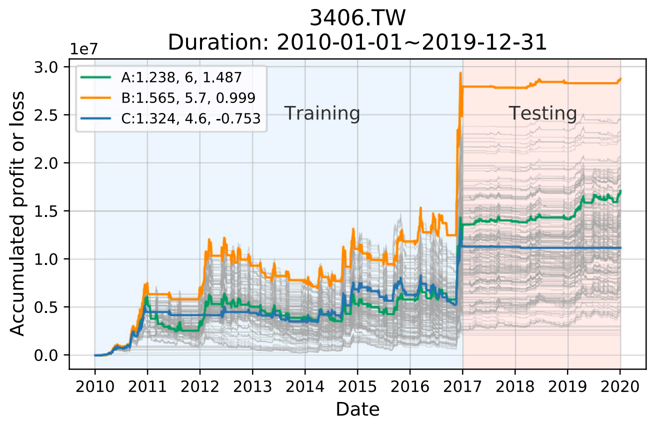


**Fig. 11.** Accumulated profit or loss for "Trending-Record High".

**Table 5**
Training and testing set performance of "Contrarian-Bollinger Bands".

| | | Plateau score | Sharpe ratio | | MAR | | Profit factor | |
|---|---|---|---|---|---|---|---|---|
| | | | Train | Test | Train | Test | Train | Test |
| 5269.TW | A | 5.2 | 0.768 | 1.123 | 0.504 | 1.297 | 1.217 | 1.388 |
| | B | 3.7 | 0.789 | 0.792 | 0.519 | 0.569 | 1.246 | 1.388 |
| | C | 3.7 | 0.601 | 0.508 | 0.381 | 0.31 | 1.166 | 1.187 |
| 1216.TW | A | 6 | 0.768 | 0.801 | 0.513 | 0.811 | 1.144 | 1.193 |
| | B | 5.82 | 0.962 | 0.836 | 0.966 | 0.706 | 1.241 | 1.221 |
| | C | 4.27 | 0.896 | −0.015 | 0.406 | −0.023 | 1.673 | 0.979 |

range. Line $A$ represents the accumulated profit or loss of the parameter with the highest plateau score, indicating stable performance, even though it may not be the best on the training set. In contrast, line $B$ shows the highest performance on the training set, but its plateau score is not particularly high, resulting in reduced profits and performance on the testing set. Line $C$ exhibits a decent performance on the training set; however, its low plateau score leads to negative profits on the testing set. Parameters with the highest plateau scores, such as line $A$, are preferred as they offer stability. Table 4 provides the training and testing set performance for multiple stock commodities using the "Contrarian-Record Low" strategy.

The "Contrarian-Bollinger Bands" strategy is outlined as follows: The initial capital is one million, and each entry unit constitutes the entire capital. The entry conditions are met when the standard deviation of the close price for the previous 20 days is less than $vol$, and the close price is lower than the Bollinger down band price of $D$. A long position is taken accordingly. All positions are exited when the close price exceeds the $D$ days Bollinger up band price. The parameters under consideration have a close price standard deviation of less than $vol$ for the previous 20 days and the period $D$ of the Bollinger midline (see Fig. 15).

Experimental results for the "Contrarian-Bollinger Bands" strategy are presented in Fig. 14. For parameters with high plateau scores, the ratio of the testing set performance to the training set performance is close to one, indicating that the performance remains stable across testing and training sets. The gray background lines represent the accumulated profit or loss for each parameter within the specified range. Line $A$ represents the accumulated profit or loss of the parameter with the highest plateau score, showcasing stable performance despite not being the best on the training set. Line $B$ exhibits the highest performance on the training set; however, the plateau score is not particularly high, resulting in less impressive performance on the testing set compared to Line $A$. Line $C$ does not display poor performance on the training set, but its low plateau score leads to suboptimal testing set results. Table 5 details the performance on the training and testing sets for multiple stock commodities using the "Contrarian-Bollinger Bands" strategy. Parameters with high plateau scores do not surpass the training set performance, but demonstrate stability through backtesting.

## 4.2. High-efficiency search algorithm

This experiment introduces a more intricate trading strategy with a greater number of parameters. The selected stock for this experiment was 5871.TW, and the trading strategy is outlined as follows. The initial capital was set to ten million, and each entry unit was denoted as $Cap$. The volatility filter entry requires that the standard deviation of the close price over the previous 20 days be less than $vol1$. An additional entry condition mandates that the close price surpass the $EntryMA$ moving average. Upon satisfying both the entry and volatility filter conditions, a long position of one unit is initiated. After establishing a stock position, if the close price exceeds the $AddMA$ moving average, an additional unit is added. The exit filter activates when the standard deviation of the previous 20 days' close price is higher than $vol2$. Furthermore, the exit condition necessitates that the close price be lower than the $ExitMA$ moving average. All positions are liquidated when the exit filter and exit conditions are satisfied simultaneously. The parameter search range for $vol1$ and $vol2$ spans from the minimum to the maximum of the 20-day standard deviation of the close price during the training period. Parameters related to entry, overweight, and exit conditions range from 3 to 22. The market entry capital sizes are as follows: one million per unit, two million per unit, five million per unit, 1/10 of the capital per unit, 1/5 of the capital per unit, and 1/2 of the capital per unit.

In this comparison, the efficiency of the brute-force search method is assessed against the PSO algorithm. The experiments were conducted on a Windows 10 PC with a 2.9 GHz octa-core Intel Core i7-10700 and 32 GB DDR4-3200 RAM. The program was implemented in Python 3.8, 64-bit. The training set spanned from January 1, 2010, to December 31, 2016, whereas the testing set covered January 1, 2017 to December 31, 2019.

The PSO algorithm configuration for this experiment included a population size of 20, randomly generated initial population position, 100 evolution iterations, and a stopping condition based on reaching the maximum evolution number or the best parameters remaining unchanged for 10 consecutive iterations. The inertia weight ($w$) underwent dynamic adjustment: it was initially set to one and multiplied by 0.99 after each iteration. The variables $C_1$ and $C_2$ were typically set to

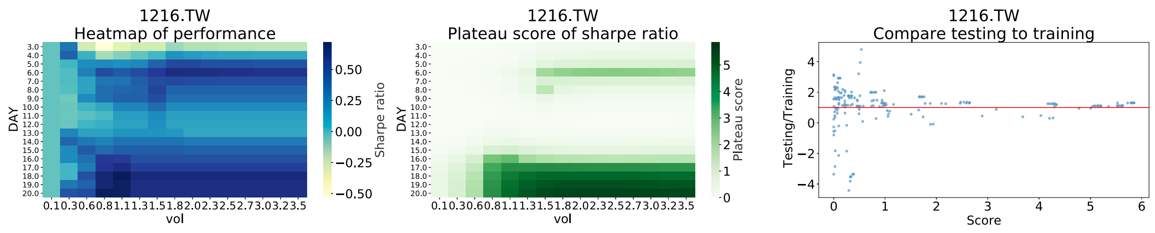


**Fig. 12.** Experimental results regarding the parameter plateau for "Contrarian-Record Low". The left figure is the heat map of SR performance; the middle figure is the SR plateau score; and the right figure is a performance comparison between testing and training.

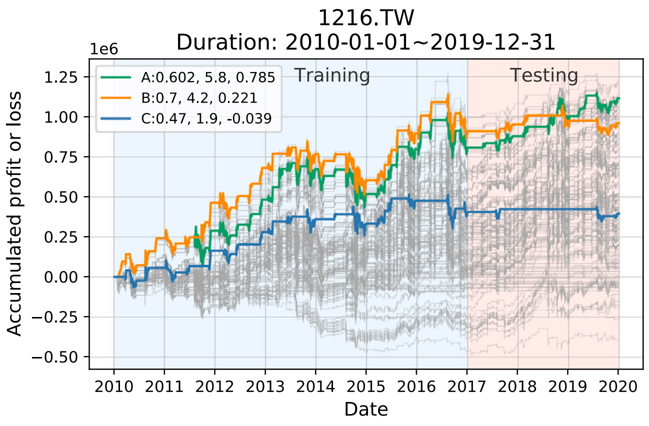


**Fig. 13.** Accumulated profit or loss for "Contrarian-Record Low".

two. The parameter settings for the plateau score algorithm involved configuring the performance indicator $Perf.$, whereby using SR with a threshold value of 0.8, a plateau score was calculated within a range of three steps.

For the brute-force method, the number of two-dimensional search parameters was 3364, requiring a total of 410 s. This results in an approximate calculation time of 0.12 s per parameter. However, the number of six-dimensional search parameters for the brute-force method was 161,472,000, with an estimated total time of 19,376,640 s. The search complexity in this case was represented by 161,472,000 sets of parameters. In contrast, the PSO algorithm completed the search for six-dimensional parameters in 108,123 s, whereas the estimated time required for the brute-force search method was 19,376,640 s. Consequently, the PSO algorithm required about 30 h for the search, significantly outperforming the brute-force method, which would take approximately 225 days in this experiment.

The accumulated profit or loss curve depicted in Fig. 16 indicates that the performance of the parameters acquired through the PSO algorithm is satisfactory. Although the algorithm might not have identified the absolute best parameter, the search time was significantly reduced with respect to the brute-force search method. Such efficiency gains may come at the cost of sacrificing some profit. However, in a dynamic and rapidly changing financial market, this outcome aligns well with the overall benefits.

### *4.3. Hyperparameters of PSO*

In the context of the parameter plateau, adjustments were made to the hyperparameters of the PSO algorithm. These hyperparameters include the initial population size, number of evolutionary iterations, inertia weight $w$, acceleration weight $C_1$ for the individual best solution, and acceleration weight $C_2$ for the group best solution. For this experiment, the initial population size was set to 20, and the positions of the initial population are the coordinates of evenly scattered points provided by UD. The number of evolutionary iterations was set to 500, and convergence was defined as the evolutionary process stabilizing for 10 consecutive iterations. The inertia weight $w$ was adjusted dynamically, decreasing gradually with each iteration. Initially preset to one, $w$ was multiplied by 0.99 after each iteration. The primary focus of analysis and adjustment lies in $C_1$ and $C_2$, which are commonly set to two. For this experiment, $C_1$ and $C_2$ were explored in the range of zero to three in increments of 0.5.

The experiments were conducted using three- and four-dimensional parameters from the trending trading strategy discussed in the previous section. The impact of hyperparameter adjustments on the final three-dimensional parameters is illustrated in Fig. 17, whereas Fig. 18 presents the effect on the final four-dimensional parameters.

In Figs. 17 and 18, darker regions indicate better final parameters. Setting $C_2$ to 0 leads to the worst evolution result. This is because, without considering the optimal solution of the group, each particle in the PSO algorithm tends to find the optimal solution only in its own region, hindering the purpose of evolution. Particle movement speed decreases when $C_1$ is set to a large value, which allows particles to potentially discover other optimal solutions during slow movement; however, this has the drawback of lower convergence speed. On the other hand, larger $C_2$ accelerates particle movement toward the group's optimal solution. However, this may lead to premature convergence as particles move excessively fast towards the group's optimal solution.

Assuming search time is not a constraint, the conclusion from this experiment suggests that when applying PSO to the parameter plateau, it is advisable to set $C_1$ between 1.5 and three, and $C_2$ between 0.5 and 2.5. This interval helps the PSO algorithm evolve towards better and more stable solutions. Assuming search time is not a constraint, the conclusion from this experiment suggests that when applying PSO to the parameter plateau, it is advisable to set $C_1$ in the range of 1.5–3, and $C_2$ in the range of 0.5–2.5. This interval helps the PSO algorithm evolve towards better and more stable solutions.

### *4.4. Efficiency comparison of the proposed algorithm and brute-force search in large solution spaces*

In the preceding section, the effectiveness of the defined parameters and plateau score were demonstrated. This section aims to enhance the complexity and sophistication of the strategy by exploring additional parameters. The experimental target for this section is Chailease Finance Co., Ltd. (stock code: 5871.TW). The trading strategy employed in this experiment is described as follows. In this experiment, the initial capital was set to 10 million, and the symbol "Cap" represents the quantity of shares bought or sold. The entry filter stipulates that the standard deviation of closing prices over the preceding 20 days is less than "vol1" and the entry condition requires the closing price to exceed the "Entry_MA"-day moving average ("Entry_MA"-day moving average refers to the $n$-day daily average for Entry_MA equal to $n$). When entry filter and condition are jointly satisfied, a single unit is purchased. Following the initial purchase, an additional unit is added if the closing price surpasses the "Add_MA" ("Add_MA"-day moving average refers to the $n$-day daily average for Add_MA equal to $n$). The exit filter demands that the standard deviation of closing prices over the preceding 20 days exceed "vol2" and the exit condition necessitates the

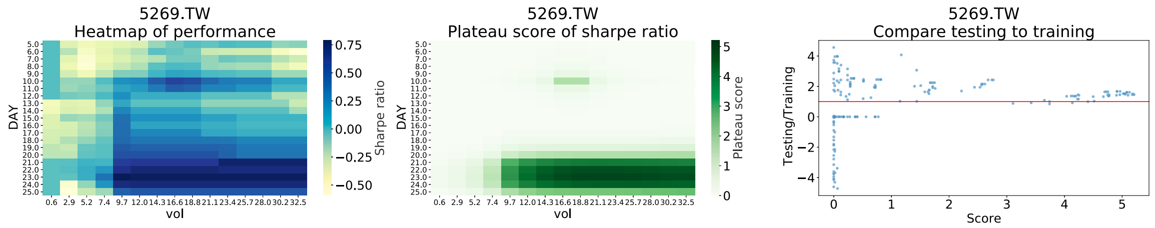


**Fig. 14.** Various experimental results regarding the parameter plateau for "Contrarian-Bollinger Bands". The left figure is the heat map of SR performance; the middle figure is the SR plateau score; and the right figure is a performance comparison between the testing and the training sets.

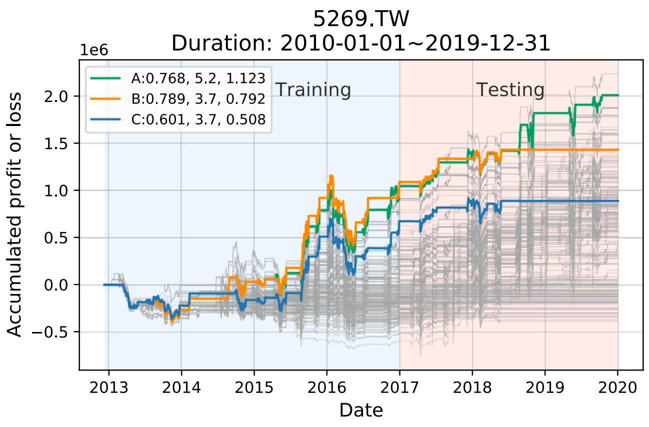


**Fig. 15.** Accumulated profit or loss of "Contrarian-Bollinger Bands".

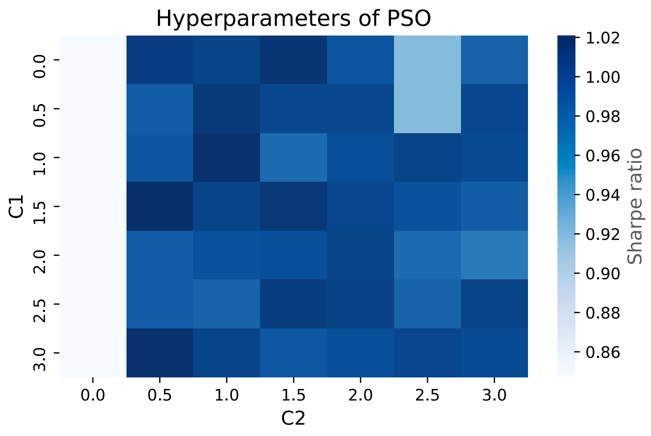


**Fig. 18.** PSO hyperparameter adjustment for four-dimensional trading strategy.

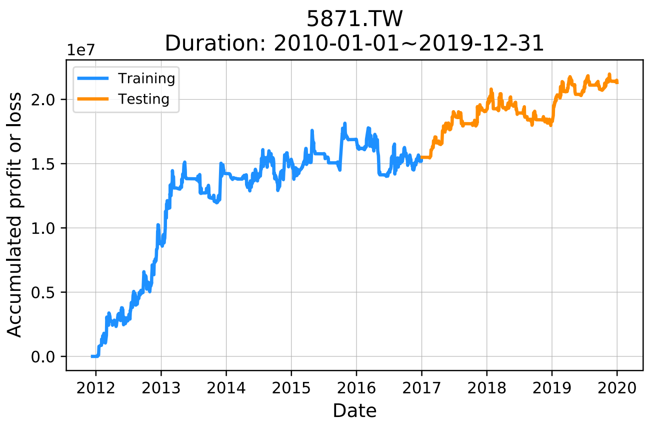


**Fig. 16.** Accumulated profit or loss of six-dimensional parameter plateau.

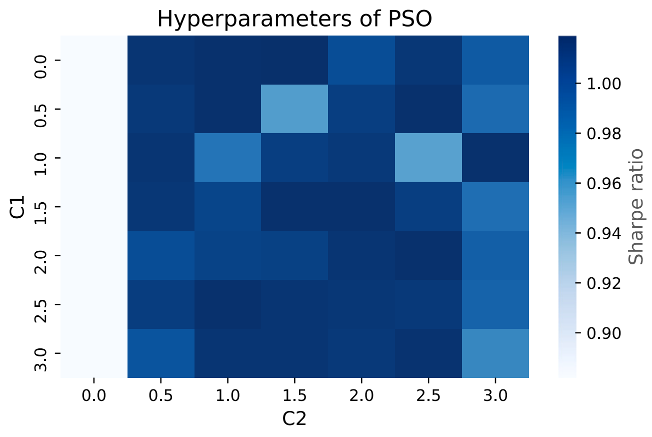


**Fig. 17.** PSO hyperparameter adjustment for three-dimensional trading strategy.

closing price to fall below "Exit_MA" ("Exit_MA"-day moving average refers to the *n*-day daily average for Exit_MA equal to *n*). When exit filter and conditions are jointly satisfied, a complete exit from the position is executed.

The parameters subject to optimization included "vol1" and "vol2" ranges spanning from the minimum to maximum values of the 20-day closing price standard deviation during the training period. The parameters for entry conditions (Entry_MA), additional conditions (Add_MA), and exit conditions (Exit_MA) ranged from 3 to 22. The exploration of the entry capital size (Cap) encompasses the following options: one million, two million, five million, 1/10 of the initial capital, 1/5 of the initial capital, and 1/2 of the initial capital. The parameter search ranges for each dimension of this trading strategy are summarized in Table 6.

In Table 6, the variable *n* represents the number of parameters to be optimized (dimensions of the search space). The ranges for "vol1" and "vol2" extended from the minimum to maximum values of the 20-day closing price standard deviation during the training period and were divided into 58 intervals. The values for Entry_MA, Add_MA, and Exit_MA ranged from 3 to 22, with the range for "Cap" including the previously mentioned six values.

In this section, the efficiency of brute-force search, UD, and PSO were compared. Due to the extremely time-consuming nature of brute-force search, for three or more dimensions, only estimations are presented. The hardware used for this experiment included a 2.9 GHz octacore Intel Core i7-10700 processor, 32 GB DDR43200 RAM, running on a Windows 10 operating system, implemented using Python 3.8 64 bit.

The target asset for this experiment was Chailease Finance Co., Ltd. (stock code: 5871.TW), suitable for employing a trend following strategy. The training dataset covered the period from January 1, 2010, to December 31, 2016, whereas the testing dataset spanned from January 1, 2017, to December 31, 2019. The PSO algorithm settings for this experiment included an initial population size of 20, the initial population located at the coordinates provided by UD's evenly

**Table 6**
Parameter search for trading strategies in different-sized search spaces (different dimensions)

| $n$ | Parameters for exploration | Size of the search space |
|---|---|---|
| 2 | 1. Entry Filter: "vol1"<br>2. Exit Filter: "vol2" | $58 \times 58$<br>$= 3,364$ |
| 3 | 1. Entry Filter: "vol1"<br>2. Exit Filter: "vol2"<br>3. Entry_MA | $58 \times 58 \times 20$<br>$= 67,280$ |
| 4 | 1. Entry Filter: "vol1"<br>2. Exit Filter: "vol2"<br>3. Entry_MA<br>4. Add_MA | $58 \times 58 \times 20 \times 20$<br>$= 1,345,600$ |
| 5 | 1. Entry Filter: "vol1"<br>2. Exit Filter: "vol2"<br>3. Entry_MA<br>4. Add_MA<br>5. Exit_MA | $58 \times 58 \times 20 \times 20 \times 20$<br>$= 26,912,000$ |
| 6 | 1. Entry Filter: "vol1"<br>2. Exit Filter: "vol2"<br>3. Entry_MA<br>4. Add_MA<br>5. Exit_MA<br>6. Cap | $58 \times 58 \times 20 \times 20 \times 20 \times 6$<br>$= 161,472,000$ |

distributed scattering, 100 evolution iterations, and stopping criteria consisting of reaching the maximum evolution generation or achieving the same optimal parameters for 10 consecutive evolutions, indicating termination of evolution or convergence, respectively. The inertia weight $w$ was dynamically adjusted with each iteration, starting at one and multiplied by 0.99 after each iteration, following the algorithm discussed in the literature and background knowledge section. The constants $C_1$ and $C_2$ are commonly set to two. For the UD-based scoring algorithm, the performance metric $Pref.$ exceeded the threshold value $\alpha$ set to a Sharpe ratio of greater than 0.8, thereby calculating the plateau score within a range of three steps.

The optimization of two parameters using the brute-force method yielded a solution space of 3364 feasible solutions, requiring a total of 410 s. The computation time for a single set of parameter solutions was approximately 0.12 s. Extrapolating, the estimation of the time required for three to six dimensions of solution space : optimizing a three-dimensional solution space involves 67,280 feasible solutions, with an estimated total time of 8074 s; for a four-dimensional solution space with 1,345,600 feasible solutions, the estimated total time is 161,472 s; in the case of a five-dimensional solution space with 26,912,000 feasible solutions, the total estimated time is 3,229,440 s; and for a six-dimensional solution space with 161,472,000 feasible solutions, the estimated total time is 19,376,640 s. The time required for parameter search and the final discovery of optimal parameter values for each search method are summarized in Table 7 providing a comparison of the final parameters among the three search methods along with the time. The UD search exhibits the shortest search time; however, the parameters discovered are the least optimal among the three methods. In the UD+PSO search method, the parameters obtained by UD are further subjected to evolution using the PSO algorithm, resulting in not only improved final parameter performance. but also savings on the search time for each parameter, as the PSO algorithm can dynamically converge toward better parameter values Among the three search methods, the brute-force search required an extensive amount of time but reliably found the optimal parameters. The UD+PSO search method reduced the search time significantly although the parameters discovered were not the most ideal. In the context of quantitative trading strategies, trading effectiveness is ephemeral and cannot afford the slow search pace of brute-force methods. The proposed UD+PSO framework demonstrates the ability to obtain sufficiently good solutions within limited time constraints and expansive search spaces. This underscores its contribution and value.

## 5. Conclusion

In this study, the concept of a parameter plateau was introduced, developing a plateau score algorithm, with the aim of replacing the conventional method of directly transferring the best-performing parameters from the training set to the testing set. The plateau score algorithm was effective in avoiding parameter islands, showcasing stable performance with a high probability. Experimental results illustrated that parameters with elevated plateau scores exhibit similar or improved performance on the testing set compared to the training set. This experimental validation underscores the substantial impact of the proposed parameter plateau definition and algorithm on parameter selection.

Subsequently, a more intricate trading strategy was examined, entailing a substantial increase in the number of parameters to be explored. In this context, unified design coupled with particle swarm optimization was employed to compute the plateau scores. An experiment encompassing the search for parameters in two- to six-dimensional trading strategies was conducted. The integration of PSO in plateau score computation significantly enhanced search efficiency compared to the brute-force method, yielding commendable final search parameters. Subsequently, an experiment involving the fine-tuning of hyperparameters for PSO in the parameter plateau was conducted. Disregarding search time considerations, a hyperparameter range was proposed for the parameter plateau applicable to PSO.

In future research endeavors, the concept of parameter plateaus introduced in this study bears substantial potential for expediting the optimization of trading strategy parameters, not only in leveraging this concept for the refinement and diversification of long/short trading strategies but also for extending its application to the optimization of investment portfolios. The present study concentrated on utilizing particle-swarm optimization to explore parameter plateaus; however, future studies may explore alternative optimization algorithms, including genetic algorithms, ant colony optimization, or artificial bee colony algorithms, with the aim of further enhancing search efficiency and solution quality.

A more in-depth exploration of the dynamic nature of parameter plateaus and their interactions with evolving market conditions can involve developing adaptive strategies capable of real-time adjustments to trading parameters based on market volatility and trends, fostering more robust and responsive trading systems. Furthermore, the exploration of machine learning techniques for identifying and exploiting parameter plateaus represents a promising avenue for future research. Machine learning models can undergo training to recognize patterns and features associated with these plateaus, potentially automating the refinement of trading strategies. Additionally, the incorporation of risk management considerations into the parameter plateau search process is a pivotal direction for future work. Crafting methodologies that consider risk factors and constraints during the optimization of trading strategies can contribute to the development of more reliable and stable investment approaches.

In summary, this study establishes a foundation for various future research directions, encompassing diverse optimization algorithms, machine learning applications, risk management, adaptive trading strategies, and real-time parameter adjustments, all aiming to advance the field of algorithmic trading.

## CRediT authorship contribution statement

**Jimmy Ming-Tai Wu:** Writing – review & editing, Supervision, Methodology. **Wen-Yu Lin:** Writing – original draft, Software, Methodology. **Ko-Wei Huang:** Writing – review & editing, Validation. **Mu-En Wu:** Writing – review & editing, Resources, Project administration, Funding acquisition, Conceptualization.

**Table 7**
Performance comparison of various search methods in terms of parameter search time and optimal parameter discovery.

| Dimensions | Bruteforce search | | Uniform design (UD) | | UD+PSO | |
|---|---|---|---|---|---|---|
| | Time | Performance | Time | Performance | Time | Performance |
| 2 | 410 s | Score: 4<br>Perf.: 1.015 | 11 s | Score: 3.96<br>Perf.: 0.912 | 181 s | Score: 4<br>Perf.: 1.01 |
| 3 | 8,074 s<br>(Estimate) | | 76 s | Score: 3.9<br>Perf.: 0.882 | 639 s | Score: 4<br>Perf.: 1.005 |
| 4 | 161,472 s<br>(Estimate) | | 82 s | Score: 3.67<br>Perf.: 0.847 | 2,711 s | Score: 4<br>Perf.: 1.005 |
| 5 | 3,229,440 s<br>(Estimate) | | 701 s | Score: 3.72<br>Perf.: 1.114 | 11,181 s | Score: 4<br>Perf.: 1.195 |
| 6 | 19,376,640 s<br>(Estimate) | | 2,969 s | Score: 3.99<br>Perf.: 1.04 | 108,123 s | Score: 4<br>Perf.: 1.118 |

## Declaration of competing interest

The authors declared that they have no conflicts of interest to this work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

## Data availability

No data was used for the research described in the article.

## Acknowledgment

We would like to express our sincere gratitude to the National Science and Technology Council in Taiwan (NSTC) for providing the funding support for this research. This work was supported by the NSTC under grant number NSTC 112-2622-027003. Their financial support has been instrumental in facilitating the progress and successful completion of this study.

## References

[1] V.-D. Ta, C.-m. Liu, D.A. Tadesse, Portfolio optimization-based stock prediction using long-short term memory network in quantitative trading, Appl. Sci. 10 (2) (2020) 437.
[2] Z. Zou, Z. Qu, Using LSTM in stock prediction and quantitative trading, 2020, CS230: Deep Learning, Winter.
[3] J. Ayala, M. García-Torres, J.L.V. Noguera, F. Gómez-Vela, F. Divina, Technical analysis strategy optimization using a machine learning approach in stock market indices, Knowl.-Based Syst. 225 (2021) 107119.
[4] M.I. Jordan, T.M. Mitchell, Machine learning: Trends, perspectives, and prospects, Science 349 (6245) (2015) 255–260.
[5] W. Khan, M.A. Ghazanfar, M.A. Azam, A. Karami, K.H. Alyoubi, A.S. Alfakeeh, Stock market prediction using machine learning classifiers and social media, news, J. Ambient Intell. Humaniz. Comput. (2020) 1–24.
[6] T.-L. Luo, M.-E. Wu, C.-M. Chen, A framework of deep reinforcement learning for stock evaluation functions, J. Intell. Fuzzy Systems (Preprint) (2020) 1–11.
[7] T.M. Mitchell, et al., Machine Learning, McGraw-hill, New York, 1997.
[8] X.-D. Zhang, Machine learning, in: A Matrix Algebra Approach to Artificial Intelligence, Springer, 2020, pp. 223–440.
[9] Y. Zhou, D. Luo, A. Wen, L. Chen, M. Lin, Simplified particle swarm optimization algorithm with improved learning factor and search method, in: 2021 International Conference on Computer Information Science and Artificial Intelligence, CISAI, 2021, pp. 405–408.
[10] M. Anthony, P.L. Bartlett, Neural Network Learning: Theoretical Foundations, cambridge University Press, 2009.
[11] B. Cao, J. Zhao, Z. Lv, Y. Gu, P. Yang, S.K. Halgamuge, Multiobjective evolution of fuzzy rough neural network via distributed parallelism for stock prediction, IEEE Trans. Fuzzy Syst. 28 (5) (2020) 939–952.
[12] S. Li, J. Wu, X. Jiang, K. Xu, Chart GCN: Learning chart information with a graph convolutional network for stock movement prediction, Knowl.-Based Syst. 248 (2022) 108842.
[13] A. Maalla, C.Y. Zhuang, Q.H. Feng, L. Shen, Research on stock market analysis based on deep learning, in: 2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC, vol. 4, 2021, pp. 1776–1780.
[14] X. Pang, Y. Zhou, P. Wang, W. Lin, V. Chang, An innovative neural network approach for stock market prediction, J. Supercomput. 76 (3) (2020) 2098–2118.
[15] J. Qiu, B. Wang, C. Zhou, Forecasting stock prices with long-short term memory neural network based on attention mechanism, PLoS One 15 (1) (2020) e0227222.
[16] S.-C. Wang, Artificial neural network, in: Interdisciplinary Computing in Java Programming, Springer, 2003, pp. 81–100.
[17] W. Weng, Quantitative trading method based on neural network machine learning, in: 2022 Asia Conference on Algorithms, Computing and Machine Learning, CACML, 2022, pp. 600–603.
[18] D.J. Hand, N.M. Adams, Data mining, in: Wiley StatsRef: Statistics Reference Online, Wiley Online Library, 2014, pp. 1–7.
[19] C. Montenegro, M. Molina, Improving the criteria of the investment on stock market using data mining techniques: The case of S&P500 index, Int. J. Mach. Learn. Comput. 10 (2) (2020).
[20] A. Raghunath, A.A. Rajak, Applications of data mining in predicting stock values, in: Machine Learning for Predictive Analysis, Springer, 2021, pp. 209–215.
[21] C.-H. Chen, Y.-H. Chen, J.C.-W. Lin, M.-E. Wu, An effective approach for obtaining a group trading strategy portfolio using grouping genetic algorithm, IEEE Access 7 (2019) 7313–7325.
[22] U. Pham, Q. Luu, H. Tran, Multi-agent reinforcement learning approach for hedging portfolio problem, Soft Comput. (2021) 1–9.
[23] J. Wang, Y. Zhang, K. Tang, J. Wu, Z. Xiong, Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1900–1908.
[24] W. Zhou, Y. Zhao, W. Chen, Y. Liu, R. Yang, Z. Liu, Research on investment portfolio model based on neural network and genetic algorithm in big data era, EURASIP J. Wireless Commun. Networking 2020 (1) (2020) 1–12.
[25] Q.A. Al-Radaideh, A.A. Assaf, E. Alnagi, Predicting stock prices using data mining techniques, in: The International Arab Conference on Information Technology, ACIT'2013, 2013.
[26] C. Zhang, O. Vinyals, R. Munos, S. Bengio, A study on overfitting in deep reinforcement learning, 2018, arXiv preprint arXiv:1804.06893.
[27] D.H. Bailey, J. Borwein, M. Lopez de Prado, Q.J. Zhu, The probability of backtest overfitting, J. Comput. Finance (2016) forthcoming.
[28] D.M. Hawkins, The problem of overfitting, J. Chem. Inf. Comput. Sci. 44 (1) (2004) 1–12.
[29] K. Fang, Experimental design by uniform distribution, Acta Math. Appl. Sinica 3 (4) (1980) 363–372.
[30] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: 2007 IEEE Swarm Intelligence Symposium, IEEE, 2007, pp. 120–127.
[31] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Onference on Neural Networks, vol. 4, IEEE, 1995, pp. 1942–1948.
[32] B. Jiao, Z. Lian, X. Gu, A dynamic inertia weight particle swarm optimization algorithm, Chaos Solitons Fractals 37 (3) (2008) 698–705.
[33] Y. Wang, C. Dang, H. Li, L. Han, J. Wei, A clustering multi-objective evolutionary algorithm based on orthogonal and uniform design, in: 2009 IEEE Congress on Evolutionary Computation, IEEE, 2009, pp. 2927–2933.
[34] J. Zhang, Y. Wang, J. Feng, Attribute index and uniform design based multiobjective association rule mining with evolutionary algorithm, Sci. World J. 2013 (2013).
[35] X. Zhu, J. Zhang, J. Feng, Multiobjective particle swarm optimization based on PAM and uniform design, Math. Probl. Eng. 2015 (2015).
[36] K. Dowd, Adjusting for risk:: An improved sharpe ratio, Int. Rev. Econ. Finance 9 (3) (2000) 209–222.
[37] A.W. Lo, The statistics of sharpe ratios, Financ. Anal. J. 58 (4) (2002) 36–52.
[38] W.F. Sharpe, The sharpe ratio, J. Portf. Manag. 21 (1) (1994) 49–58.

[39] G. Daskalakis, R.N. Markellos, Are the European carbon markets efficient, Rev. Futures Mark. 17 (2) (2008) 103–128.
[40] M. Magdon-Ismail, A.F. Atiya, Maximum drawdown, Risk Mag. 17 (10) (2004) 99–102.
[41] M. Magdon-Ismail, A.F. Atiya, A. Pratap, Y.S. Abu-Mostafa, On the maximum drawdown of a Brownian motion, J. Appl. Probab. 41 (1) (2004) 147–161.