scientific reports

Check for updates

OPEN

Multiple objectives escaping bird search optimization and its application in stock market prediction based on transformer model

Dedai Wei[1,7], Zimo Wang[2,7], Minyu Qiu[3], Juntao Yu[4], Jiaquan Yu[4], Yurun Jin[5], Xinye Sha[6]✉ & Kaichen Ouyang[4]✉

Stock market prediction has long attracted the attention of academia and industry due to its potential for substantial financial returns. Despite the availability of various forecasting methods, such as CNN, LSTM, BiLSTM, GRU, and Transformer, the hyperparameter optimization of these models often faces limitations, particularly in single-objective optimization, where they can easily fall into local optima. To address this issue, this paper proposes an innovative multi-objective optimization algorithm—the Multi-Objective Escape Bird Algorithm (MOEBS)—and introduces the MOEBS-Transformer architecture to enhance the efficiency and effectiveness of hyper-parameter optimization for Transformer models. This study first validates the performance of MOEBS through a series of multi-objective benchmark tests on standard problem sets such as ZDT, DTLZ, and WFG, comparing it with other multi-objective optimization algorithms (e.g., MOMVO, MSSA, and MOEAD) using evaluation metrics such as GD, Spacing, IGD, and HV for comprehensive analysis. In the context of stock price prediction, we select the closing price datasets of Amazon, Google, and Uniqlo, using MOEBS to optimize the core hyper parameters of the Transformer while considering multiple objectives, including training set RMSE, testing set RMSE, and testing set error variance. In the experiments, this paper first compares CNN, LSTM, BiLSTM, GRU, and traditional Transformer models to establish the Transformer as the optimal model for stock market prediction. Subsequently, the study compares the MOEBS-Transformer with Transformer models optimized using various hyperparameter optimization methods, including MOMVO-Transformer, MSSA-Transformer, and MOEAD-Transformer. Additionally, it evaluates Transformer models optimized through conventional methods: Random Search (RS-Transformer), Grid Search (GS-Transformer), and Bayesian Optimization (BO-Transformer). By assessing the performance of these models using R^2, RMSE, and RPD metrics on both training and testing sets, the results demonstrate that the Transformer model optimized by MOEBS significantly outperforms the other methods in terms of prediction accuracy and prediction stability. This research offers a new solution for complex optimization scenarios and lays a foundation for advancements in stock market prediction technologies.

Keywords Stock price prediction, Multi-objective optimization, Escaping bird search, Transformer, Hyper-parameter optimization

Stock price forecasting is a key focus area within the financial industry, aiming to accurately forecast the future trajectory of stock prices through market data analysis. This task is influenced by a multitude of economic,

[1]College of Economics, Shenyang University, Shenyang 110000, China. [2]Department of Philosophy, Nankai University, Tianjin 300381, China. [3]College of Information Engineering, China Jiliang University, Hangzhou 310018, Zhejiang, China. [4]Department of Mathematics, University of Science and Technology of China, Hefei 230026, China. [5]School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China. [6]Graduate School of Arts and Sciences, Columbia University, NY 10027, USA. [7]Dedai Wei and Zimo Wang contributed equally to this work. ✉email: xs2399@columbia.edu; oykc@mail.ustc.edu.cn

political, and social factors, making it extremely complex and unpredictable. Factors such as global economic growth, policy changes, market sentiment, and unforeseen events can significantly impact stock prices[1–3]. Additionally, behaviors of market participants, companies' financial performance, and macroeconomic conditions continually affect the stock market[4–6]. The dynamic interplay of these factors presents a challenge in accurately predicting stock price movements[7–10].

With advancements in technology, particularly in computer science, artificial intelligence, and machine learning, modern financial analysis is undergoing a revolution. Machine learning techniques like support vector machines[11], neural networks, and deep learning models have been widely utilized for stock market predictions[12–15]. These models can process vast amounts of complex data and identify trends elusive to people. Thus, machine learning demonstrates immense potential in enhancing the accuracy and efficiency of predictions[16–18].

In the realm of deep learning, the Transformer model has garnered extensive attention and research due to its exceptional ability to handle long sequence data[19]. This model utilizes a unique attention mechanism to efficiently grasp intricate dependencies within time series data, thereby exhibiting outstanding performance in various tasks, particularly in natural language processing (NLP) and financial time series analysis[20,21]. However, despite the Transformer model's potential across multiple domains, optimizing its hyper-parameters such as Learning rate, Number of Heads and regularization coefficient to achieve optimal performance remains a significant challenge[22]. In complex application scenarios like financial market analysis, traditional single-objective optimization methods often fall short as they typically focus on optimizing one performance metric while neglecting others that might be equally important[23–28]. Moreover, the single-objective optimization Transformer model performs well when handling specific tasks, but it also has significant limitations. Firstly, the model is optimized for only one task, meaning it can only focus on a single objective during training and inference, which limits its adaptability and flexibility, especially in scenarios that require handling complex, diverse tasks. Additionally, single-objective models cannot effectively utilize shared data across tasks in a multi-task learning setting, resulting in lower data efficiency. In multi-task learning, shared information and knowledge among tasks can significantly improve model performance, but the limitations of single-objective models prevent them from benefiting from these shared insights, reducing their generalization ability and adaptability to new tasks. Lastly, since each task requires a separate model to be trained independently, this increases the demand for computational resources and extends training time, decreasing overall efficiency.

Multi-Objective Optimization (MOO) offers a more comprehensive solution by simultaneously considering and optimizing multiple conflicting objectives[29–31]. This approach seeks not just to identify optimal solutions for one single objective but to find the best trade-offs among all objectives[32,33]. In practice, this often involves a process known as Pareto optimization, where the core concept is the Pareto front. The Pareto front comprises a set of non-dominated solutions, indicating a state where any improvement in one objective does not lead to the deterioration of any other objectives[17,34]. Therefore, adopting a multi-objective learning approach combined with Transformer models offers significant advantages. First, multi-objective learning allows the model to share data and knowledge across multiple related tasks, thereby utilizing existing data resources more efficiently and enhancing overall learning outcomes. Additionally, by simultaneously handling multiple tasks, the multi-task model can better capture diverse features within the data, improving the model's generalization ability and making it more robust when faced with new tasks. Moreover, multi-objective learning reduces the need to independently train multiple single-objective models, thereby lowering computational resource consumption and shortening training time, ultimately increasing development efficiency. In summary, combining multi-objective learning with Transformer models not only overcomes the limitations of single-objective models but also enhances the model's overall performance and practical value.

In order to realize the idea of combining multi-objective learning with Transformer models, we propose multi-objective Escaping Bird Search algorithm (MOEBS). First, the Escaping Bird Search (EBS) is a unique heuristic optimization method built upon the behavior of bird flocks evading predators in nature[6,7]. EBS tackles optimization problems by simulating this evasion behavior, showing good dynamic adaptability and global search capabilities[6,12]. Second, when introduced to multi-objective scenarios, we specifically developed MOEBS, an extension of EBS designed for multi-objective problems[17]. MOEBS leverages the dynamic evasion mechanism of EBS to emulate the maneuvers of birds evading predators in the search space, effectively exploring optimal solutions among various objectives[35]. This method places particular emphasis on archival strategies, systematically preserving non-dominated solutions from history, which gradually approach the Pareto front through iterative processes, offering key stakeholders a variety of viable optimal choices[9,36,37]. In order to assess the effectiveness of MOEBS, we conducted wilcoxon signed-rank test and detailed benchmark testing built upon problems like ZDT, DTLZ, WFG on the 5 standard multi-objective benchmark problem sets, and evaluated its effect using four indexes: Generational Distance(GD), Inverted Generational Distance(IGD),Hyper-volume (HV), and Spacing(SP)[34]. The experiment is not limited to MOEBS, but also compared with the following three MOO algorithms: Multi-Objective Multi-Verse Optimization (MOMVO), Multi-Objective Evolutionary Algorithm based on Decomposition (MOEAD), and Multi-Objective Salp Swarm Algorithm (MSSA)[16,35]. These experiments clearly demonstrate the outstanding performance of MOEBS in solving multi-objective optimization problems. Compared to the other three benchmark algorithms, MOEBS consistently matched or outperformed their performance across various evaluation metrics. The results not only validate the superiority of MOEBS in handling complex optimization tasks but also highlight its potential and value in the field of multi-objective optimization[12].

To validate the superior optimization capability of MOEBS, we proposed the MOEBS-Transformer architecture and applied it to hyperparameter optimization tasks in machine learning, conducting experiments on the closing price datasets of Amazon, Google, and Uniqlo. Specifically, we optimized key hyperparameters of the Transformer model, including the number of attention heads, initial learning rate, and regularization

coefficient. The performance of the optimized MOEBS-Transformer was compared against various optimization methods, including Random Search (RS)[38], Grid Search (GS)[39], Bayesian Optimization (BO)[40], and Transformer models optimized using three multi-objective optimization algorithms: MOMVO, MSSA, and MOEAD. Additionally, to comprehensively assess the prediction performance of different models, we selected five classic deep learning models: Convolutional Neural Network (CNN)[41], Long Short-Term Memory Network (LSTM)[42], Bidirectional Long Short-Term Memory Network (BiLSTM)[43], Gated Recurrent Unit (GRU)[44], and the traditional Transformer model[45], all tested on the same closing price datasets. The model performance was evaluated using metrics such as R^2(coefficient of determination), RMSE (Root Mean Squared Error), and RPD (Residual Prediction Deviation). This structured comparison not only highlighted the effectiveness of the MOEBS-Transformer in optimizing hyperparameters but also demonstrated its superior performance relative to other optimization algorithms in terms of prediction accuracy and prediction stability. This comprehensive experimental design provides strong evidence and reliable support for the superior performance of MOEBS in complex optimization scenarios[46]. Comparative experiments showed that MOEBS-Transformer significantly outperformed other competitor models mentioned earlier across multiple performance indicators[12,24] such as RMSE, RPD, and R^2[47]. These results not only demonstrate the effectiveness of MOEBS in real-world financial applications but also showcase its powerful potential in handling complex multi-objective scenarios, providing a new, more efficient tool for financial market analysis and prediction[24]. This comprehensive optimization strategy allows the Transformer model[20]to better adapt to the needs of financial market analysis, improving its competitiveness and efficiency in practical applications[24].

Moreover, the innovation of this study lies in proposing MOEBS and MOEBS-Transformer, Transformer integrated with MOEBS for hyperparameter fine-tuning, to optimize the complex task of stock market prediction[48–50]. This method enhances the model's performance by effectively and efficiently fine-tuning hyperparameters to achieve better performance in terms of financial time series prediction, offering a new efficient tool for financial time series analysis[7,51–53]. The significant contributions of this article can be summarized as follows:

- Introduces a novel approach by integrating multi-objective optimization with Transformer models to enhance prediction accuracy in financial data.
- Proposes a multi-objective extension of the Escaping Bird Search (EBS) algorithm, called MOEBS, which was validated through benchmark tests on problems like ZDT, DTLZ, and WFG, and compared with other advanced optimization methods using metrics such as GD, SP, IGD, and HV.
- MOEBS is then applied to the optimization of hyperparameters of Transformer including learning rate, the number of attention heads, and regularization coefficient.
- In experiments on the stock data of Amazon, Google, and Uniqlo, the performance of the MOEBS-Transformer model surpassed that of other competing algorithms, as evidenced by improvements in metrics such as RMSE, RPD, and R^2.
- Five deep learning models (CNN, LSTM, BiLSTM, GRU, and Transformer) were validated on multiple real-world datasets, providing substantial experimental evidence
- The MOEBS-Transformer model proves to be a powerful tool for stock market prediction, offering enhanced accuracy and stability, making it valuable for both research and practical financial applications.

The structure of this paper is as follows. Section "Literature review" briefly introduces definitions and preliminary knowledge of MOO. The original EBS is conceptually reviewed, and the multi-objective version is presented in Section "Multiple Objective Escaping Bird Search Optimization". Section "Discussion of numerical experiment" includes the MOEBS experimental results in benchmark tests on problems like ZDT, DTLZ, and WFG. Section "Review of Classical Time Series Models and Experiments" utilizes five classic deep learning models (CNN, LSTM, BiLSTM, GRU, and Transformer) to conduct experiments on the closing prices of Amazon, Google, and Uniqlo. In section "Hyperparameter optimization experiments for the Transformer model", the experiments compared Transformer models under different hyperparameter optimization methods, including Random Search (RS), Grid Search (GS), Bayesian Optimization (BO), and multi-objective optimization methods such as MOEBS-Transformer, MOMVO-Transformer, MOEAD-Transformer, and MSSA-Transformer. Lastly, Section "Conclusion and future work" includes the conclusions, limitations and future work.

Literature review

In the first section, we provided a detailed overview of the research framework and experimental procedures of this study. In this section, we will build on the findings from various prior studies and analyses to offer an in-depth explanation of the Pareto front in the context of current multi-objective optimization problems.

Numerous practical challenges require optimizing several objectives that often conflict with one another and must be minimized or maximized simultaneously. Such types of challenges are known as Multi-Objective Optimization Problems (MOOPs)[54,55]. A MOOP aimed to minimize can be represented by the mathematical expression as below:

$$\begin{aligned} &\text{Minimize } F(X) = (f_1(X), f_2(X), ..., f_m(X)) \\ &\text{subject to } g_i(X) \geqslant 0, i = 1, 2, ..., p \\ &h_j(X) = 0, j = 1, 2, ..., s \\ &L_k \leqslant X_k \leqslant U_k, k = 1, 2, ..., n \end{aligned} \tag{1}$$

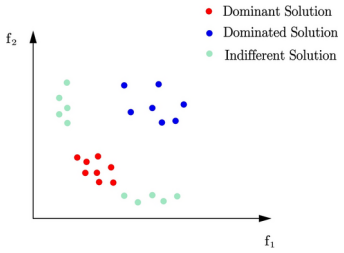

Fig. 1. Pareto dominance.

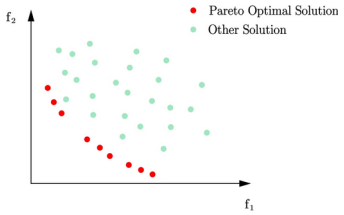

Fig. 2. Pareto optimal.

where $f_1, f_2, ..., f_m$ specify the objective functions for minimization, m represents the quantity of objective functions, X denotes the decision variables vectors, p indicates the quantity of inequality constraints, s stands for the quantity of equality constraints, g_i refers to the i^{th}inequality constraint[56], h_j stands for the j^{th} equality constraint, $n$denotes the number of decision variables[55], L_k and U_k stand for the lower and upper bound of the k^{th}decision variable within the solution space[31,33]. MOO differs significantly from Single-Objective Optimization (SOO). In SOO, the aim is to find one single global optimal solution within the population, while MOO yields a set of various solutions named Pareto-optimal solutions[32,57]. For SOO, the objective function is scalar, meaning that comparing two candidate solutions requires only an evaluation of their objective function values[58]. In a minimization task, the solution with the lowest objective function value is considered the optimal[59]. Conversely, MOO involves vector-valued objective functions, and the comparison between 2 solutions is based on the concept of Pareto dominance, which allows for the evaluation of multiple objective function values simultaneously[36,37]. For a minimization problem, if X and Z are the candidate solutions, the set of all feasible solutions, considering the constraints, is denoted as Ω. The four distinct mathematical definitions that express the concepts of Pareto dominance are presented as below:

Pareto dominance

$$X, Z \in \Omega$$
$$if \quad \begin{array}{c} \forall i \in \{1, 2, ..., m\}, f_i(X) \leqslant f_i(Z) \\ \wedge \exists j \in \{1, 2, ..., m\}, f_j(X) < f_j(Z) \end{array} , X \succ Z \tag{2}$$

where, for the X solution to be considered superior (dominance) over the Z solution, it must not perform worse than Z in any objective function and must achieve a better value in at least one objective function. If this is satisfied, the X solution is said to dominate the Z solution (Pareto dominates), which is denoted as $X \succ Z$. Figure 1 illustrates the concept of Pareto dominance. The red dots stand for the Dominant Solution, the blue dots stand for the Dominated Solution, and the green dots stand for the Indifferent Solution. The dominant solution is not inferior to the dominated solution on all goals and is superior to the dominated solution on at least one goal.

Pareto optimal

$$If \neg \exists Z \in \Omega : Z \succ X, X \text{ is Pareto optimal} \tag{3}$$

where, every candidate within the Ω set is evaluated against the other solutions based on Pareto dominance (introduced in Def. 1). Following this evaluation, if no Z solution is found to dominate the X solution, X is considered a Pareto optimal solution. Figure 2 depicts the concept of Pareto optimal solutions. Within the figure, red dots represent Pareto Optimal Solutions and green dots represent Other Solutions. Pareto optimal solutions are the ones that are not dominated by others on all goals.

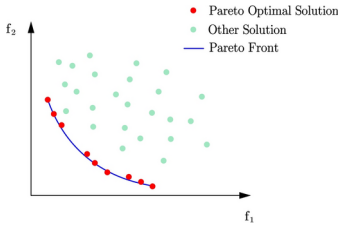

Fig. 3. Pareto front.

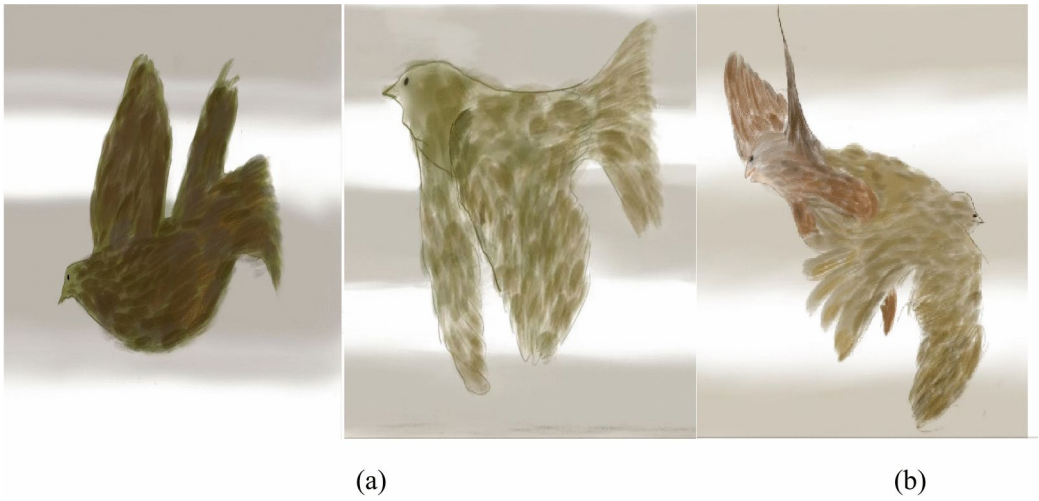

Fig. 4. A swift (**a**) utilizing wing beat-rate during straight flight, (**b**) in a turning maneuver.

Pareto optimal set (PS)
The set of Pareto optimal solution vectors.

$$PS = \{X \in \Omega | \neg \exists Z \in \Omega : Z \succ X\} \tag{4}$$

Pareto optimal front (PF)
The set of Pareto optimal objective vectors.

$$PF = \{F(X) | X \in PS\} \tag{5}$$

In Fig. 3, red dots represent Pareto Optimal Solutions, green dots represent Other Solutions, and the solid blue line represents the Pareto front. The pareto front contains every Pareto optimal solution, showing the optimal tradeoff between different objectives within MOO.

Multiple objective escaping bird search optimization
This section provides a detailed introduction to the fundamental concepts of the original Escaping Bird Search (EBS) optimization and delves into the multi-objective optimization mechanism within MOEBS, explaining how EBS has been extended and applied to optimize multi-objective problems.

Escaping bird search optimization
Within many natural ecosystems, the interaction between prey and predators is essential to the survival of species. Different organisms have evolved various strategies to evade predators and ensure their survival. This discussion focuses on avian prey-predator systems in open airspace. Typically, in these systems, the prey has a smaller size compared to the predator. Although the prey, with its smaller wingspan, might fly slower than its predator, it compensates for this disadvantage through other adaptive traits. For instance, these birds are able to alter their wing beat-rate during attacking and escaping maneuvers. Additionally, the great maneuverability of the prey serves as a crucial advantage in evading predators. The swift (Apus apus) is a gliding bird known for its remarkable ability to increase speed by sweeping its wings back and for making sharp turns by spreading them. Compared to many other birds, swifts exhibit exceptionally high maneuverability (as shown in Fig. 4).

The unique shape of their wings, combined with their rapid wing beat-rate, allows them to execute swift turns in nearly any direction. In simulations, such a prey bird might employ one of the following strategies to evade a hunter bird.:

- Level maneuvering: The prey bird chooses to execute rapid turns to evade capture by a predator attacking from a horizontal approach. (the hunter-bird).

- Vertical maneuvering: To evade predator diving from above, the prey often employs a strategy of rapid upward climbing flights. Alternatively, when the predator is ascending to capture it, the prey may opt to dive downward as a means of escape.

Figure 5 provides an illustrative example of the strategies discussed above. The likelihood of a bird executing an aerial maneuver is influenced by factors such as its body area and velocity. A smaller body area reduces aerodynamic resistance force, enhancing maneuverability. These aerial maneuvers form the basis of a novel optimization algorithm named Escaping Bird Search (EBS). EBS is a population-based algorithm that explores the design space through aerial maneuvers performed by pairs of artificial birds, which serve as search agents. Within every randomly selected pair from the population, the bird with lower fitness is designated as the prey (escaping bird), while the other acts as the predator. The position of an artificial bird within the search space corresponds to a design vector, which is adjusted during the bird's flight. A bird's maneuverability can be influenced by aspects such as its body area, wing beat-rate, and speed. Within the current numerical emulation, Maneuver-Power (MP) of the i^{th} artificial bird; MP_i can be denoted as:

$$MP_i = b_i \|V_i\|^{\beta} \tag{6}$$

where, the velocity vector V_i is calculated by taking the difference between current and previous positions of the i^{th} bird. $\|V_i\| = \sqrt{\sum_j V_{i,j}^2}$ is computed as the norm of V_i. The influence of wing beat-rate variation on MP is represented by the parameter β that arbitrarily alternates between 0 and 2. The body factor b_i models the impact of the bird's body area (cost) using the normalization equation below:

$$b_i = \frac{C_{\max} - C_i}{C_{\max} - C_{\min} + \varepsilon} \tag{7}$$

where C_i refers to the cost of the i^{th} agent; C_{max} and C_{min} stands for maximum and minimum costs within the present population, accordingly. The infinitesimal constant ε, is introduced to avoid division by zero. Additionally, an Escaping Rate, *ER* is defined using MPs of the Attacking Bird, *AB* acting as the predator and the Escaping Bird, *EB* acting as the prey. This relationship is expressed as the following:

$$ER = \frac{MP_{EB}}{MP_{AB} + MP_{EB}} \tag{8}$$

where, MP_{AB} and MP_{EB} refers to the MPs of the AB and the EB, correspondingly. In vertical maneuvers, the EB chooses a destination in the direction opposite to the predator. This behavior is emulated using the artificial flight model below:

$$X_{EB} = X_{EB} + r * ER * (Opp(X_{AB}) - X_{EB}) \tag{9}$$

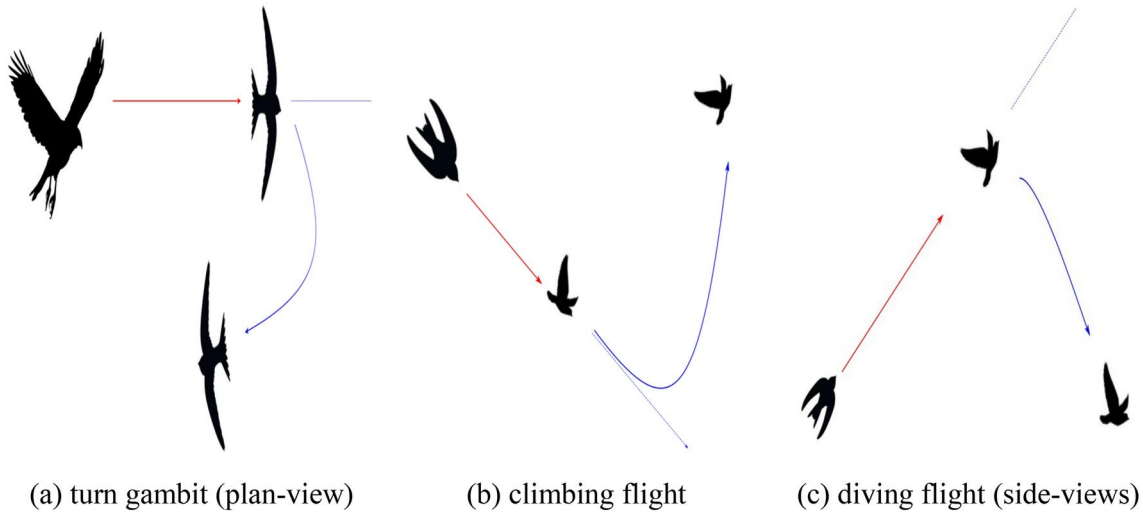

Fig. 5. Aerial escape maneuvers in a prey-predator bird system (The dashed line represents the predator's path immediately after missing the prey).

where, X_{AB} and X_{EB} refer to the positions of the AB and the EB, accordingly. *r* stands for an arbitrarily generated number uniform distributed between 0 and 1. The escaping rate *ER* is computed by Eq. (8). The function $Opp(X_{AB})$ gives a vector's opposite within the design space:

$$Opp(X_{AB}) = X_L + X_U - X_{AB} \tag{10}$$

When ER is low, the artificial prey selects a horizontal maneuver by veering off its current path. This is modeled by changing to an arbitrarily generated position vector as follows:

$$X_{EB} = X_L + R * (X_U - X_L) \tag{11}$$

where, X_L and X_U represent vectors of the lower and upper limits of the design variables, accordingly. The symbol $\otimes$ represents the element-wise multiplication and *R* stands for a vector of random values within the range [0,1]. The predator's flight is primarily directed towards the prey's position. Nevertheless, in this emulation, an additional direction is introduced: flying towards the global optimal position achieved by any prey so far; represented by X_{Gbest}. Hence, the predator maneuver is expressed as:

$$X_{AB}^{Candidate} = X_{AB} + r_1 * CR * (X_{EB} - X_{AB}) + r_2 * CR * (X_{Gbest} - X_{EB}) \tag{12}$$

here, the arbitrary values r_1 and r_2 are separately generated in the range [0,1] and the Capturing Rate, *CR*, is denoted by:

$$CR = 1 - ER \tag{13}$$

When ER is large, CR is close to 0, and vice versa. This is why *CR* and *ER* are considered complementary to each other. Based on Eq. (8), *ER* models the combined effects of both the predator and the prey during an aerial encounter. Equation (12) represents a vector-sum position update, an approach commonly used by other swarm intelligence algorithms such as PSO. However, unlike PSO, EBS employs only 1 adaptive coefficient CR, instead of PSO's cognitive and inertial factors. Additionally, the final component in Eq. (12) is not scaled by any predefined factor. The proposed algorithm, which is based on the emulated maneuvers of artificial predator-prey pairs, is constructed using 4 steps listed below.

Step 1. Initialize a population of *N* artificial birds with arbitrary positions. Spawn each i^{th} bird in the upper and lower bounds of the design variables using the following:

$$X_i = X_L + R * (X_U - X_L) \tag{14}$$

where, the right-hand-side parameters are introduced in Eq. (11). Subsequently, calculate the cost function for every bird and initialize their velocities to 0. The initial population consists of both prey and predator birds to be differentiated in **Step 3**.

Step 2. Compute the MP for every bird using Eq. (6) using the normalized cost values from Eq. (7).

Step 3. Repeat below until the termination criterion is met:
For $i = 1, ...N$ do:

- Pick the i^{th} bird and arbitrarily pair it with another bird within the population. Select the best as *AB* and the worst as *EB*.
- Compute *MP*, *ER* and *CR* of the present pair of predator (*AB*) and prey (*EB*) using Eq. (6)-(8) and Eq. (13).
- Offer a candidate solution for *AB* using Eq. (12).
- Calculate cost of $X_{AB}^{Candidate}$
- Substitute X_{AB} with $X_{AB}^{Candidate}$ if $X_{AB}^{Candidate}$ achieves less cost than X_{AB} (greedy selection)
- Terminate the main iteration once the Number of cost or fitness Function Evaluations; *NFE* reaches predetermined NFE_{max}

Generate a candidate solution for *EB* based on either level-turning using Eq. (11) or vertical-maneuver using Eq. (9). Alternate between the two using the condition below:

$$X_{EB}^{Candidate} = \begin{cases} X_L + R \otimes (X_U - X_L) & if\ ER < 1/N \\ X_{EB} + r * ER * (Opp(X_{AB}) - X_{EB}) & otherwise \end{cases} \tag{15}$$

1) Calculate cost of $X_{AB}^{Candidate}$
2) Substitute X_{EB} with $X_{EB}^{Candidate}$ if $X_{EB}^{Candidate}$ achieves less cost than it (greedy selection) Terminate the main iteration once NFE reaches NFE_{max}

Update XG_{Best}: If $i = N$ and $NFE < NFE_{max}$, return to **Step 3**, otherwise terminate the iteration and advance to **Step 4.**

Step 4: After exiting the main iteration, return the current XG_{Best} as the optimum solution. Clearly, EBS operates with only 2 control parameters of *N* and NFE_{max}.

Multiple objective escaping bird search

To build a multi-objective extension of the base EBS, several primary modifications are necessary. These changes are somewhat analogous to those implemented in MOPSO (Coello et al., 2004). As previously discussed, in MOOPs, a set of non-dominated solutions, known as Pareto optimal solutions, can be achieved. To manage these solutions, a new feature, namely an archive, is introduced. It acts as a storage repository to retain the non-dominated solutions discovered throughout the optimization procedure over multiple iterations. Notably, the maximum capacity of the archive is a parameter determined by the user. By incorporating such a feature into EBS for MOO, 4 scenarios may arise:

(1) A recently found Pareto solution can be dominated by at least one of the current solutions in the archive. In this case, the recently found solution would be ignored.
(2) The new solution dominates one or more solutions in the archive. In this case, the dominated solutions are eliminated from the archive and the new solution would be added to the archive.
(3) If neither the new solution nor archive members dominate each other, the new solution should be added to the archive.
(4) If the archive is full, the grid mechanism should be run to re-arrange the segmentation of the objective space to find the most crowded segment, the best segment to eliminate a solution from.

In the grid mechanism, objective function space is divided into different regions. An example of such a mechanism is illustrated by Fig. 6. Notably, if a new solution introduced to the archive exceeds the established boundaries, a recalculation for the grid must be performed, resulting in the repositioning of each individual within it. Each hypercube within the grid stands for a region consisting of a certain number of individuals.

From the provided graph, the most densely populated segment is identified with ease. The second feature introduced to EBS is the optimal solution selection technique. In the base EBS, identifying the optimal solution is straightforward, based on the objective function values of the candidates. However, for a multi-objective problem, determining the best solution is more complex because of the Pareto optimality. To choose the optimal solution, which serves as the leader for guiding the rest of the candidates in updating their positions during the optimization process, a leader selection mechanism is employed. This mechanism identifies the least populated segments of the objective space using a grid system and selects one of the non-dominated solutions within that segment as the leader. The selection process is then conducted based on the roulette wheel selection approach, after a lower fitness value is assigned to every segment.

$$P_j = \frac{\lambda}{N_j} \tag{16}$$

where λ stands for a constant above 1. N_j refers to the quantity of non-dominated particles within j^{th} segment. The above formulation reveals that it allocates a lower probability to more populated segments. That is to say, the fewer particles within a segment, the greater its probability of being selected. This approach makes sure that the optimal solution (leader) in every iteration is chosen from one of the least populated segments of the objective function space. This strategy enhances the algorithm's coverage by encouraging candidates to update

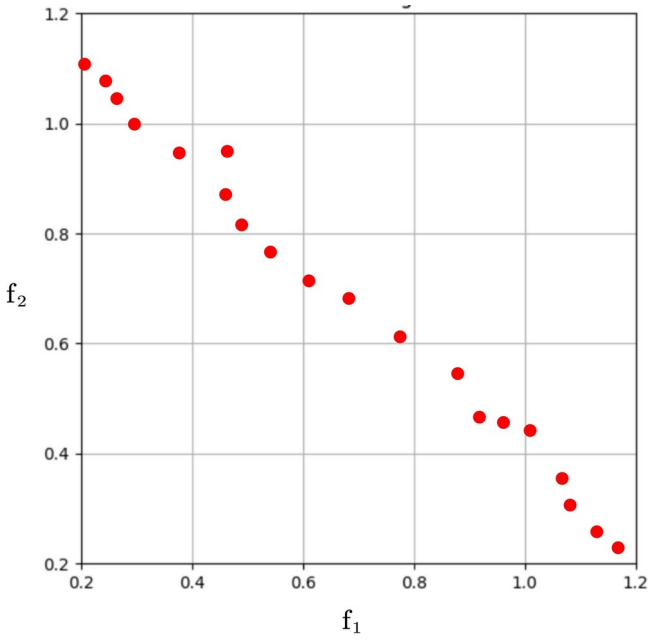

Fig. 6. Grid mechanism.

their positions toward less-explored regions within the solution space. The framework of MOEBS is shown in Fig. 7. To summarize, the key strengths of the newly introduced MOEBS are as follows:

- The use of an external archive guarantees the efficient storage of the optimal non-dominated solutions gathered throughout the optimization process.
- User determined parameters like the quantity of candidates generated through the diffusion process, allow MOEBS to effectively tackle complex MOO-problems.
- Gradually reducing the parameter ER throughout the iteration process ensures a balanced trade-off between coverage and convergence in MOEBS.
- Because MOEBS retains the updating mechanism from the basic EBS, it can efficiently explore the solution space.
- Unlike many multi-objective algorithms, MOEBS requires only a few parameters to be tuned.
- The grid mechanism enables MOEBS to identify the most and least populated regions within the solution space.
- The leader selection mechanism allows MOEBS to enhance the diversity of the non-dominated solutions obtained.
- By allocating lower probabilities to highly populated segments, MOEBS reduces the probability of local optima trapping.

The pseudo-code for MOEBS is included as follows:

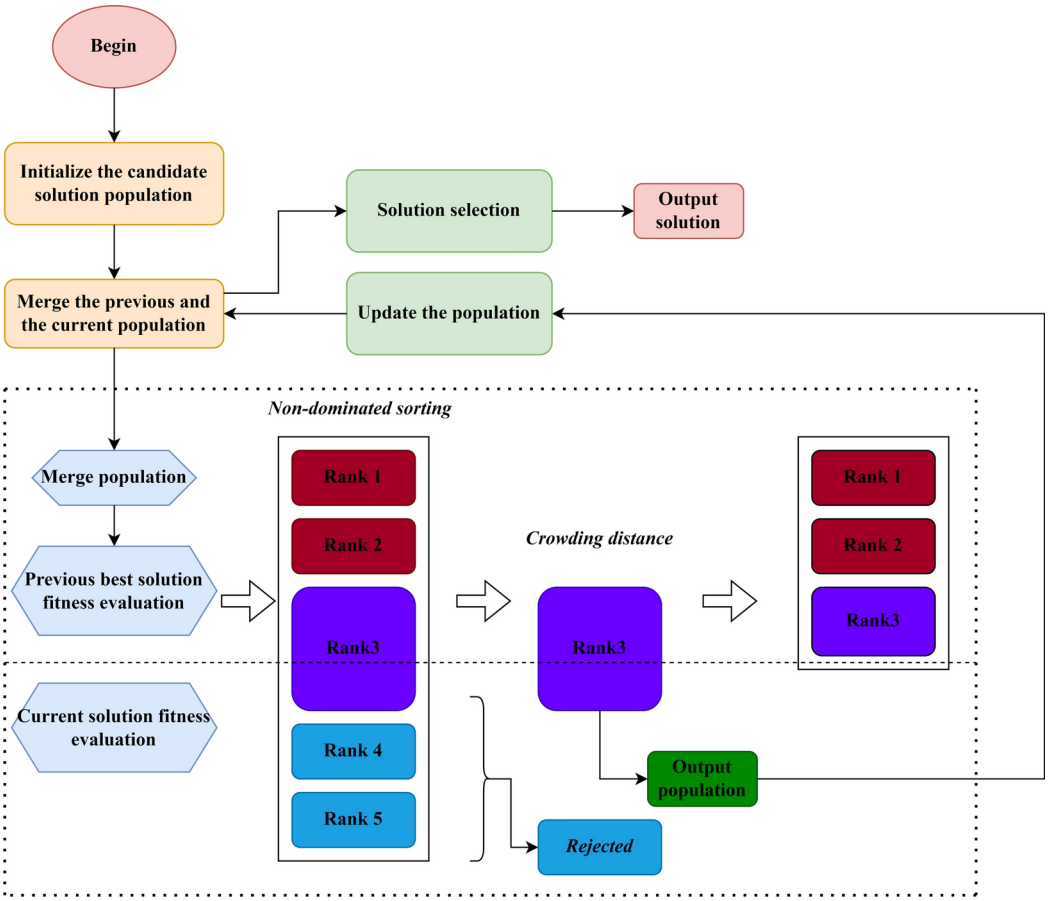

Fig. 7. The framework of MOEBS.

Set the parameters for MOEBS: Np (No.of particles), Nr (Maximum size of the repository), max_iter (Maximum No. of generations), $ngrid$ (Number of grid divisions), $nVar$ (No. of variables), lb (Lower bounds of variables), ub (Upper bounds of variables)

//Initialization

For each particle $i = 1\ to\ Np$:

Initialize position $POS(i)$ using random values within $[lb, ub]$.

Compute fitness $POS\_fit(i) = fun(POS(i))$.

End For

Store initial positions as $PBEST$ and fitness as $PBEST\_fit$.

//Repository Initialization

Identify non-dominated solutions and initialize the repository *REP* (Repository containing non-dominated solutions).

Update grid for *REP*.

Loop while $iter\ count < max\_iter$:

//Cost Computation

For each particle $i = 1\ to\ Np$:

Calculate $Costs(i) = 0.5 * sum(POS\_fit(i))$.

End For

Normalize velocities $vNrm(i)$ for each particle.

Calculate MP based on costs and velocities.

//Update Positions

For each particle $h = 1\ to\ Np$:

Select $Gbest$ from *REP*.

Pair attacking and escaping birds.

Compute new position $POS(h)$ using a combination of personal best, global best, and opposing positions.

Ensure position $POS(h)$ remains within $[lb, ub]$.

End For

//Velocity Update

Update velocity $vMat = POS - xMat\_old$.

//Evaluate New Population

For each particle $i = 1\ to\ Np$:

Recompute fitness $POS\_fit(i) = fun(POS(i))$.

End For

//Update Repository Using Grid Mechanism

Add new non-dominated solutions to *REP*.

If *REP* exceeds Nr

remove excess particles using crowding distances.

Update grid for *REP*.

//Update Personal Bests (Leader Selection)

Compare current fitness with $PBEST\_fit$, and update $PBEST$ and $PBEST\_fit$ if new fitness dominates the previous best.

//Generation Counter

Increment *iter count*.

End While

Return *REP*

Algorithm 1. MOEBS

Discussion of numerical experiment

In this section, we conducted comparative experiments on five standard multi-objective benchmark problems: ZDT[60], DTLZ[61], WFG[60], UF[61], and CF[60]. We compared the performance of MOEBS against other multi-

objective optimization algorithms, including MOMVO, MOEAD, and MSSA. These experiments allowed us to thoroughly evaluate the effectiveness and advantages of MOEBS in multi-objective optimization tasks, providing strong evidence for its potential in real-world applications.

Experiment in the five standard multi-objective benchmark problem sets

In this section, we detail the experimental setup, display the experimental outcomes, and offer a review. For all experiments, the setup parameters are shown in Table 1. For testing cases, we selected 5 standard multi-objective benchmark problem sets. The effectiveness of MOEBS is evaluated across these benchmark problem sets, as outlined below:

- ZDT problem set: ZDT1-ZDT4, ZDT6
- DTLZ problem set: DTLZ1-DTLZ7
- WFG problem set: WFG1-WFG10
- UF problem set: UF1-UF10
- CF problem set: CF1-CF10

The rationale for conducting a variety of experiments throughout a variety of benchmark problem sets is to create a platform that guarantees impartiality, trustworthiness, and fairness for the methods being compared. This setup minimizes the risk of unintended bias that could advantage a particular candidate because of advantageous testing configurations.

Discussion of Pareto front alignments of the algorithms on benchmark problem sets

Figure 8 illustrates the performance of the CF problem set, highlighting the strong performance of MOEBS. The results indicate that MSSA, MOMVO, and MOEAD also demonstrate a certain competitiveness, albeit with some differences. MSSA effectively satisfies constraints in certain experimental settings but generally falls short in terms of coverage and resolution of the Pareto front. Its solution sets tend to cluster in specific regions, occasionally deviating from the true Pareto front, thus limiting its exploration of the solution space's breadth and depth. A similar situation is observed with MOMVO. On the other hand, MOEAD exhibits better solution distribution in some cases, but its consistency in satisfying complex constraints is not as robust as MOEBS. This inconsistency may stem from MOEAD's algorithm structure and parameter adjustments, leading to instability in some test scenarios. However, MOEBS consistently outperforms in terms of solution coverage and deviation from the true Pareto front across various scenarios. Overall, while MSSA, MOMVO, and MOEAD show some optimization capabilities on the CF problem set, MOEBS demonstrates more significant advantages in solution quality, uniformity, and constraint satisfaction.

The DTLZ problem set is designed to assess an algorithm's capability to deal with multi-dimensional target optimization problems, especially in the case of up to three-dimensional or more targets. DTLZ problems usually involve complex nonlinear relationships and extensive search space. It can be observed from Fig. 9 that MOEBS has significant advantages in DTLZ problems, especially in problems such as DTLZ4 and DTLZ6, which can effectively cover the vast Pareto front and keep a highly uniform distribution of solutions. MSSA only managed to form a cluster of solutions on DTLZ2 and DTLZ4, showing a poor capability in terms of Pareto front coverage. MOMVO exhibits better solution distribution in some cases, but its consistency in satisfying complex constraints is not as robust as MOEBS as it completely fails to approximate the Pareto front in DTLZ6. Overall, MOEBS shows more obvious advantages in the quality, uniformity of generated results and the ability to satisfy constraints in this set.

The UF problem set is constructed to understand and compare the effectiveness of the algorithm in an unconstrained MOO environment. It is clear in Fig. 10 that MOEBS has demonstrated excellent performance on all 3 UF problems, especially in terms of fast convergence and wide coverage. MSSA, MOMVO, and MOEAD all fail to estimate the true Pareto front precisely enough. Their solutions tend to be clustered as well, yielding to poor Pareto front coverage. MOEBS shows a better distribution of solution sets in these tests, which indicates that it can explore the target space more efficiently in an unconstrained environment and achieve superior multi-objective optimization performance.

The WFG test set is crafted to assess an algorithm's ability to handle highly complex and nonlinear problems. For the three selected WFG problems, as illustrated in Fig. 11, MOEBS demonstrates exceptional solution diversity and coverage. Additionally, MOEBS shows superior continuity and uniformity in its solutions compared

Parameters	MOEBS	MOEAD	MSSA	MOMVO
Number of runs	30	30	30	30
Population size (N_p)	100	100	100	100
Maximum number of generations (T)	1000	1000	1000	1000
Archive size (N_r)	100	100	100	100
Other related parameters		Crossover rate, $CR = 0.5$		Maximum of wormhole existence probability,$WEPmax = 1$
		Number of neighbors, $T_n = 15$		Minimum of wormhole existence probability,$WEPmin = 0.2$

Table 1. Experimental setup parameters for optimization algorithms.

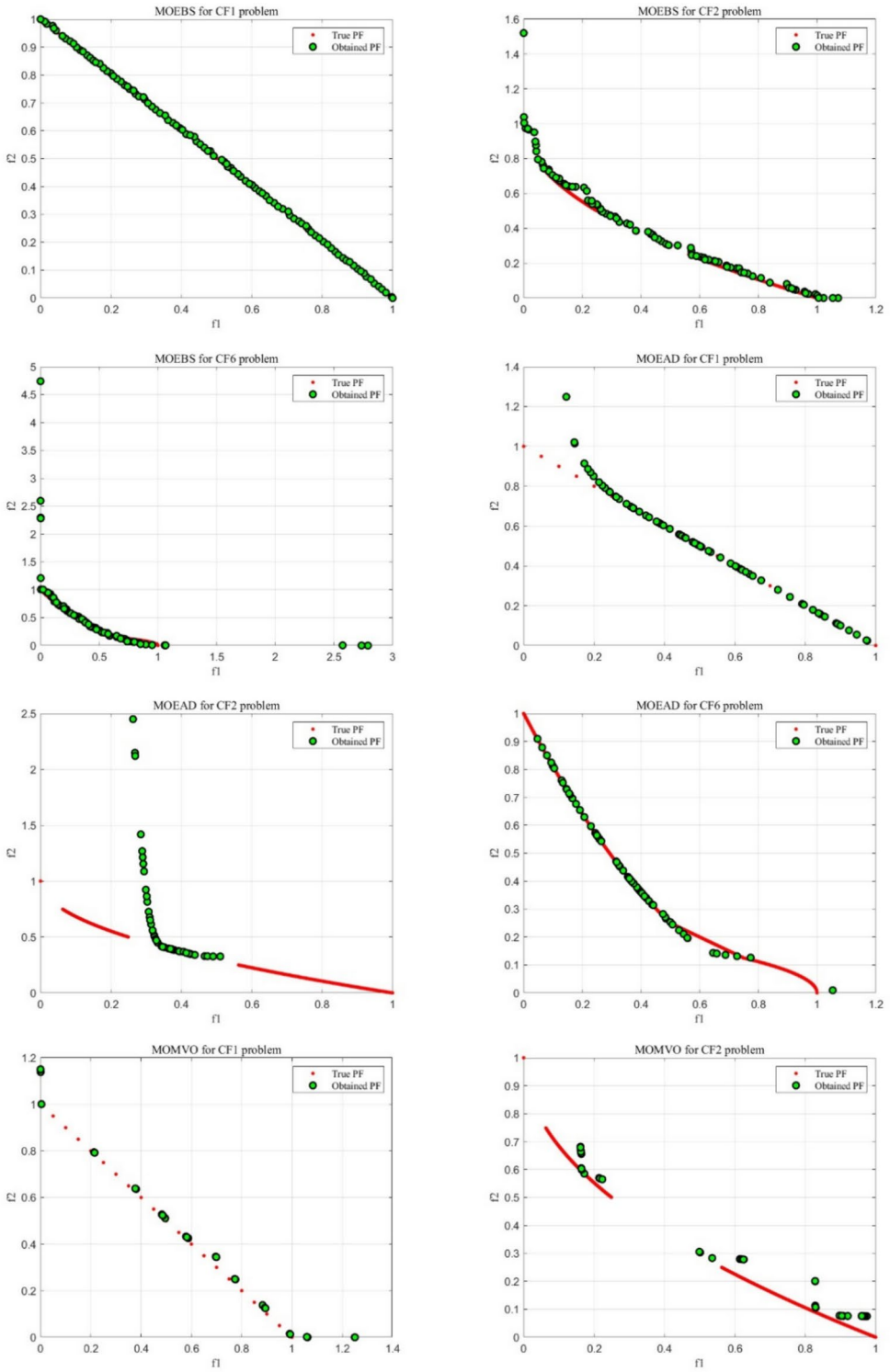

Fig. 8. Results of Pareto fronts generated by all algorithms in CF1, CF2 and CF6.

to the other three competitors. It effectively explores multiple target spaces, avoiding local optima. In contrast, MOMVO's performance is inconsistent, particularly in WFG8, where it lacks the stability shown in the other two problems. Both MOEAD and MSSA exhibit limited exploration in high-dimensional target spaces, with their solution sets focusing on specific front regions.

Figure 12 presents the result on ZDT problem set. Both ZDT1 and ZDT4 are designed with smooth, continuous, and convex Pareto fronts. In both cases, the experimental outcomes demonstrate that MOEBS is able to closely approximate the ideal Pareto front, highlighting its strength when handling convex optimization problems. MOEBS efficiently explores the solution space and generates high-quality solution sets. However,

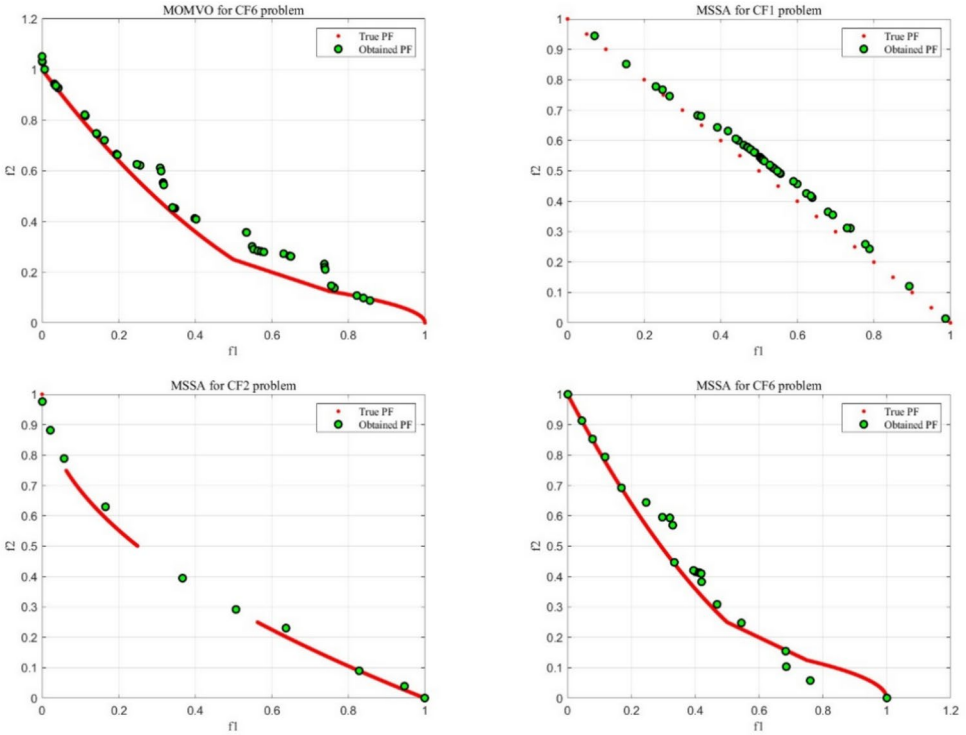

Figure 8. (continued)

MOEAD and MSSA achieve satisfactory results only in one problem, indicating their limited applicability. MOMVO approximates a close Pareto front but is clearly outperformed by MOEBS in ZDT1. Furthermore, MOEBS excels in ZDT3, a problem with a non-continuous Pareto front, where MOEAD and MSSA struggle significantly. These findings demonstrate that MOEBS is not only adept at addressing complex front shapes but also maintains efficient and stable optimization performance for basic convex front problems.

Evaluation of MOO capability of the algorithms based on performance metrics

The test problem sets are regarded as some of the most challenging in the literature, offering a variety of multi-objective search spaces with distinct Pareto optimal fronts, including convex, non-convex, discontinuous, and multi-modal types. To evaluate the effectiveness, we have used Inverted Generational Distance (IGD)[62], Spacing (SP)[63], Generational Distance (GD)[64]and Hypervolume (HV)[65] as metrics for measuring convergence and performance.

Inverted generational distance (IGD)

Inverted generational distance (IGD) is a metric that evaluates the quality of approximations to the Pareto front achieved by MOO algorithms. It is denoted by the following expression:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{17}$$

where n denotes the quantity of elements within PF_{ture} and d_i denotes the Euclidean distance from the solution i within PF_{known} to its closest solution within PF_{ture}. A zero value means that all generated solutions rest exactly on the true Pareto front.

Figure 13 shows a schematic of IGD indicator. The dashed lines represent the Pareto front, the red dots represent the reference points, and the green dots stand for the generated solutions. The arrows show the distance of each generated solution to the closest reference point. IGD measures the ability of the generated solution set to approach the Pareto front by calculating the mean of these distances. The shorter the distance, the less the IGD value, indicating that the nearer the generated solution set is to the Pareto front, the better the algorithm performance.

Figure 14 present the box plot results of four different algorithms (MOMVO, MSSA, MOEAD, MOEBS) on ZTD, DTLZ and other 3 problem sets, analyzed through IGD. MOEBS generally shows the lowest median IGD and a relatively narrow interquartile range (IQR), indicating that it can consistently find solutions closest to the

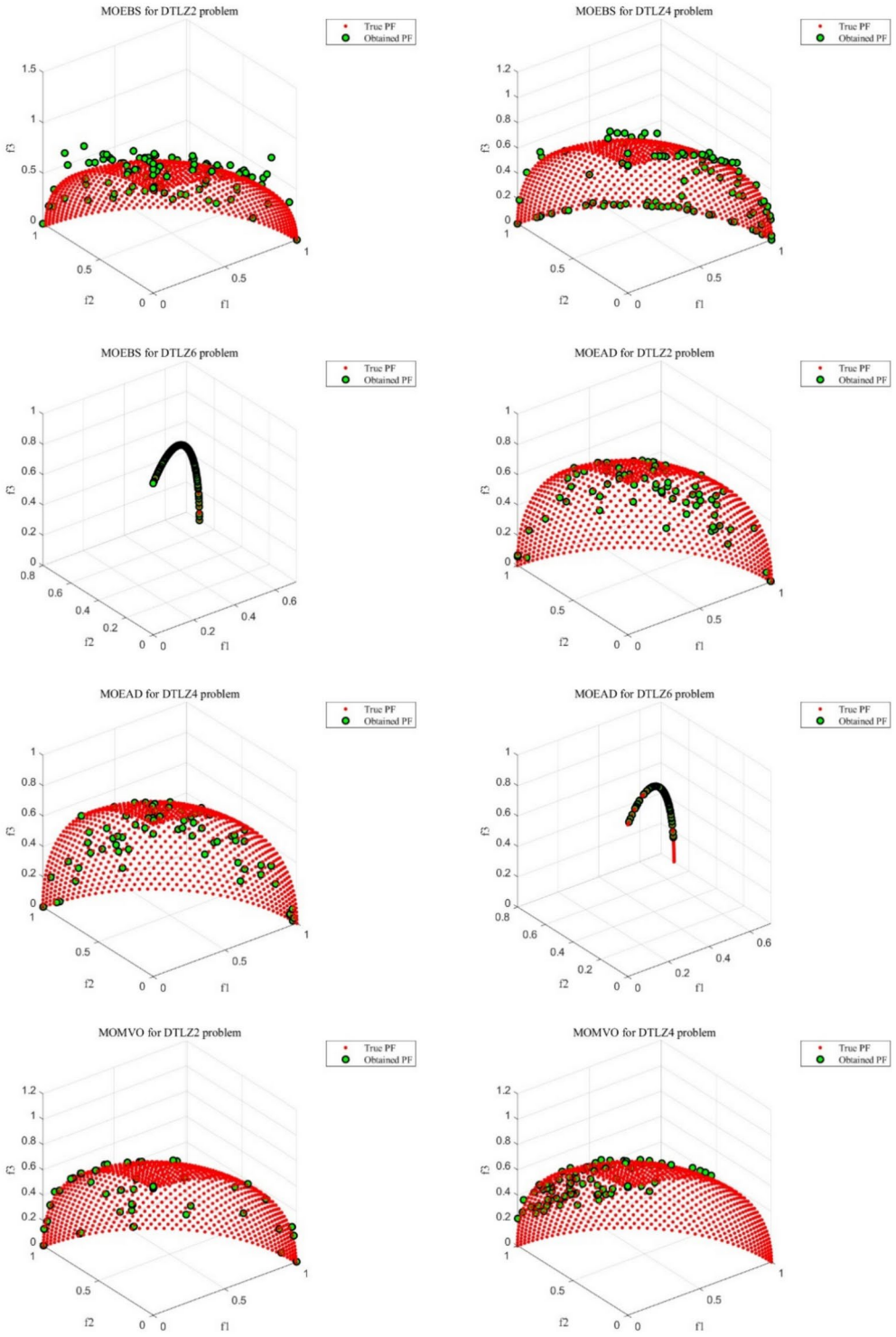

Fig. 9. Results of Pareto fronts generated by all algorithms in DTLZ2, DTLZ4 and DTLZ6.

true Pareto front. MOMVO has a similar performance as MOEBS, while MSSA and MOEAD tend to have either higher median values or larger IQRs, indicating worth performance with inconsistencies.

As shown in Table 2, the IGD calculates the mean distance between the approximate solution set and the real Pareto front. MOEBS performs particularly well on multiple test functions. In the ZDT series of problems, the mean IGD values of MOEBS for ZDT1, ZDT3 and ZDT6 problems are 0.004415, 0.005024 and 0.003504 respectively, which are the lowest values, and the standard deviation is also 0.000150, 0.000173 and 0.000139 respectively. It shows its excellent approximation ability and high stability. In the DTLZ series of problems, the mean IGD of MOEBS for DTLZ3 and DTLZ6 problems is 1.092400 and 0.005199, respectively, which is significantly lower than other algorithms, and the standard deviations are 0.899732 and 0.000359, respectively,

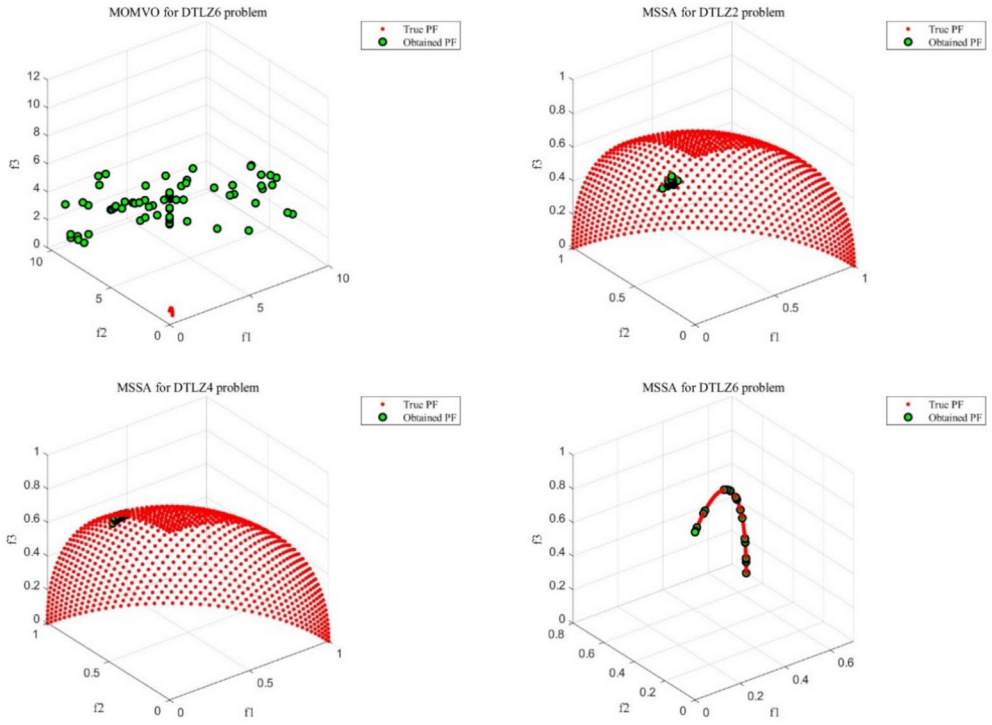

Figure 9. (continued)

which further proves its strong ability and consistency in MOOPs with high dimensions. In addition, in the WFG series of problems, MOEBS also performed well with an average IGD of 0.68266 on the WFG5 problem. Overall, MOEBS achieves the lowest mean on 16 problems sets, while MOMVO, MSSA, and MOEAD achieve the lowest on 9, 10, and 6 correspondingly. For the standard deviation, MOEBS achieves the lowest on 22 problem sets, while MOMVO, MSSA, and MOEAD achieve the lowest on 12, 7, and 0 correspondingly. These data show that MOEBS not only has a significant advantage in approximating the real Pareto front, but also excels in the consistency and stability of its results, making it more competitive in multi-objective optimization tasks.

Spacing (SP)
Spacing is utilized to quantify and assess the coverage. It indicates how uniformly the solutions achieved are distributed along the PF_{known}. It is defined as follows:

$$SP = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2} \tag{18}$$

where N_{pf} in PF_{known} is the quantity of non-dominated solutions and d_i refers to the Euclidean distance from the solution i within PF_{known} to the nearest solution within PF_{ture}. When SP is low, the convergence is better in terms of robustness and reliability.

Figure 15 shows a diagram of the Spacing indicator. The green dot represents the generated solutions, and the black line represents the distance between the generated solutions. The Spacing index measures the distribution uniformity of the solution set by calculating the distance between the generated solution sets. The more uniform the Spacing, the lower the spacing value, denoting that the more uniformly distributed the solution set in the target space and the greater the diversity, the better the algorithm performance.

Figure 16 present the box plot results of the four different algorithms on ZTD, DTLZ and other 3 problem sets, analyzed through Spacing. MOMVO tends to have low median values together with narrower IQRs, indicating great performance. For MOEBS, it generally shows higher spacing value with some level of inconsistency indicated by wider IQRs. This remains as a limitation of our proposed MOEBS and there is still room for improvement. MSSA and MOEAD show varied performance inconsistencies in terms of medians and IQRs.

In Table 3, the SP index measures the diversity and distribution uniformity of the solution set. MOEBS has the lowest SP mean and standard deviation on several test functions, which shows its significant advantages in solving MOOPs. For example, in the ZDT series of problems, the mean SP of MOEBS for ZDT1, ZDT3 and ZDT4 problems is 0.006526, 0.007169 and 0.006058, respectively, which are the lowest, indicating that its solution set is the most uniformly distributed in the target space. Overall, MOEBS achieves the lowest mean on 7 problems

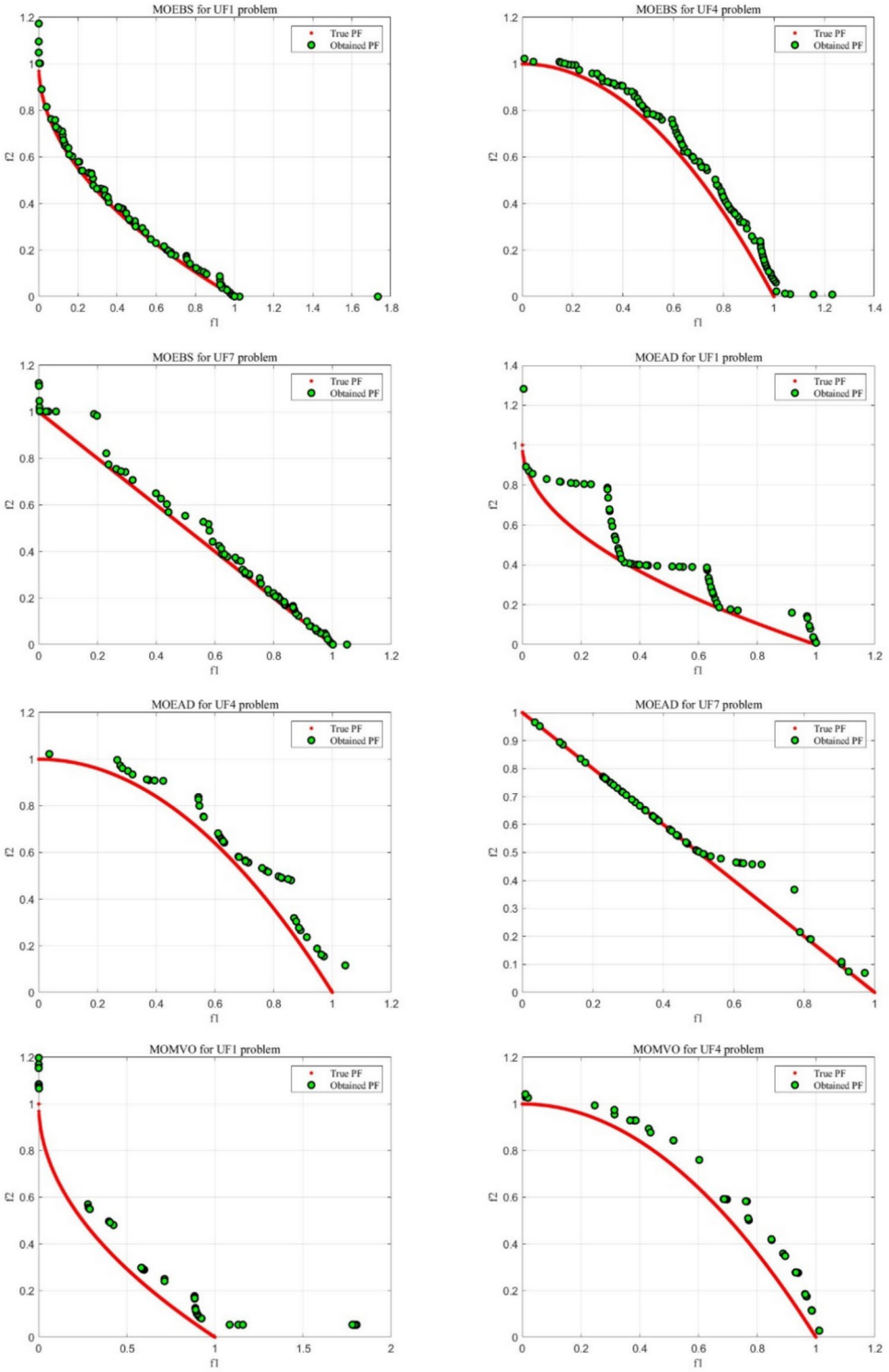

Fig. 10. Results of Pareto fronts generated by all algorithms in UF1, UF4 and UF7.

sets, while MOMVO, MSSA, and MOEAD achieve the lowest on 10, 6, and 18 correspondingly. For the standard deviation, MOEBS achieves the lowest on 9 problem sets, while MOMVO, MSSA, and MOEAD achieve on 13, 4, and 15 correspondingly. Compared with other cutting-edge algorithms, MOEBS has a matching performance in different types and dimensions of problems, fully demonstrating its wide applicability and superiority.

Hypervolume (HV)

HV computes the volume, within the objective space, covered by candidates of set A, mathematically. HV values are then computed as follow:

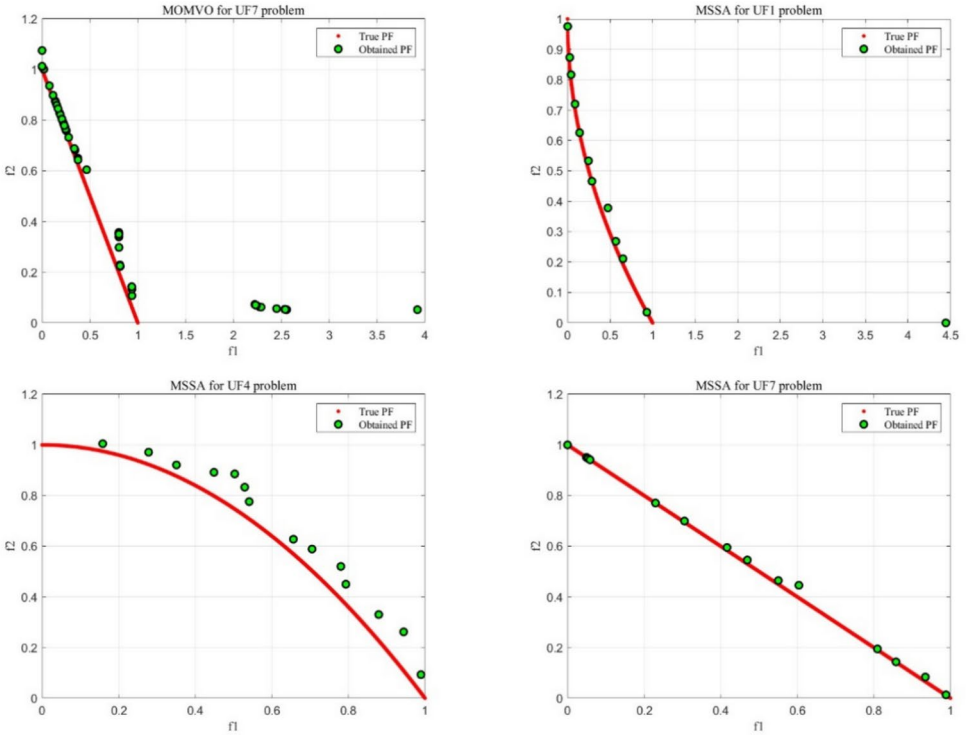

Figure 10. (continued)

$$HV = volume\left(\bigcup_{i=1}^{N_{obt}} V_i\right) \tag{19}$$

where, for every $i \in A$ solution, a v_i hypercube is constructed based on W the reference point and where the i solution stands for the diagonal of the hypercube.

Figure 17 shows a schematic of the HV metrics. The dashed lines stand for the Pareto front, and the reference points are located at the intersection of the two longest perpendicular solid lines in the coordinate axis. The green and blue areas together represent the hypervolume. The green area represents the volume covered from the reference point to each generated solution, while the blue area represents the sum of these volumes to form the total hypervolume. The HV index evaluates algorithm performance by calculating the total hypervolume covered by the generated solution set. The greater the hypervolume, the better the quality of the generated solution set, and the wider the distribution, the better the algorithm performance.

Figure 18 present the box plot results of the 4 algorithms on ZTD, DTLZ and other 3 problem sets, analyzed through HV. MOEBS generally shows much higher median HV values in several plots, sometimes with a few large IQRs, indicating great performance with some inconsistencies. In contrast, all other three algorithms have varied performance, with better performance on some problems and worse performance on the others. This indicates that they might have potentials in specific problems domains, but their versatility is worse than MOEBS. Generally, MOEBS performs the best on HV, indicating the best solution distribution and coverage.

As the Table 4 shows, MOEBS performs particularly well on HV metrics across multiple test functions. Specifically, in the ZDT series of problems, the mean HV values of MOEBS for ZDT1 and ZDT6 problems are 0.719928 and 0.388432 respectively, both of which are the highest values, and the standard deviation is also small, indicating its strong coverage ability and stability. MOEBS performed well with mean HV values of 0.309414 and 0.309425 for WFG4 and WFG5 problems, respectively. In the CF series of problems, the mean HV of MOEBS for CF1 and CF2 problems is 0.571171 and 0.621633, respectively, with a small standard deviation, which further proves its powerful ability in complex constrained multi-objective optimization problems. Overall, MOEBS achieves the highest mean on 12 problems sets, while MOMVO, MSSA, and MOEAD achieve the highest on 14, 7, and 8 correspondingly. For the standard deviation, MOEBS achieves the lowest on 20 problem sets, while MOMVO, MSSA, and MOEAD achieve on 14, 6, and 4 correspondingly. Therefore, MOEBS has a matching performance as its competitors while maintaining the best consistency in terms of multi-objective optimization.

Generational distance (GD)

Distance between the true Pareto front (PF_{ture}) and the achieved Pareto front (PF_{known}) is denoted by generational distance. This metric is mathematically defined as follows:

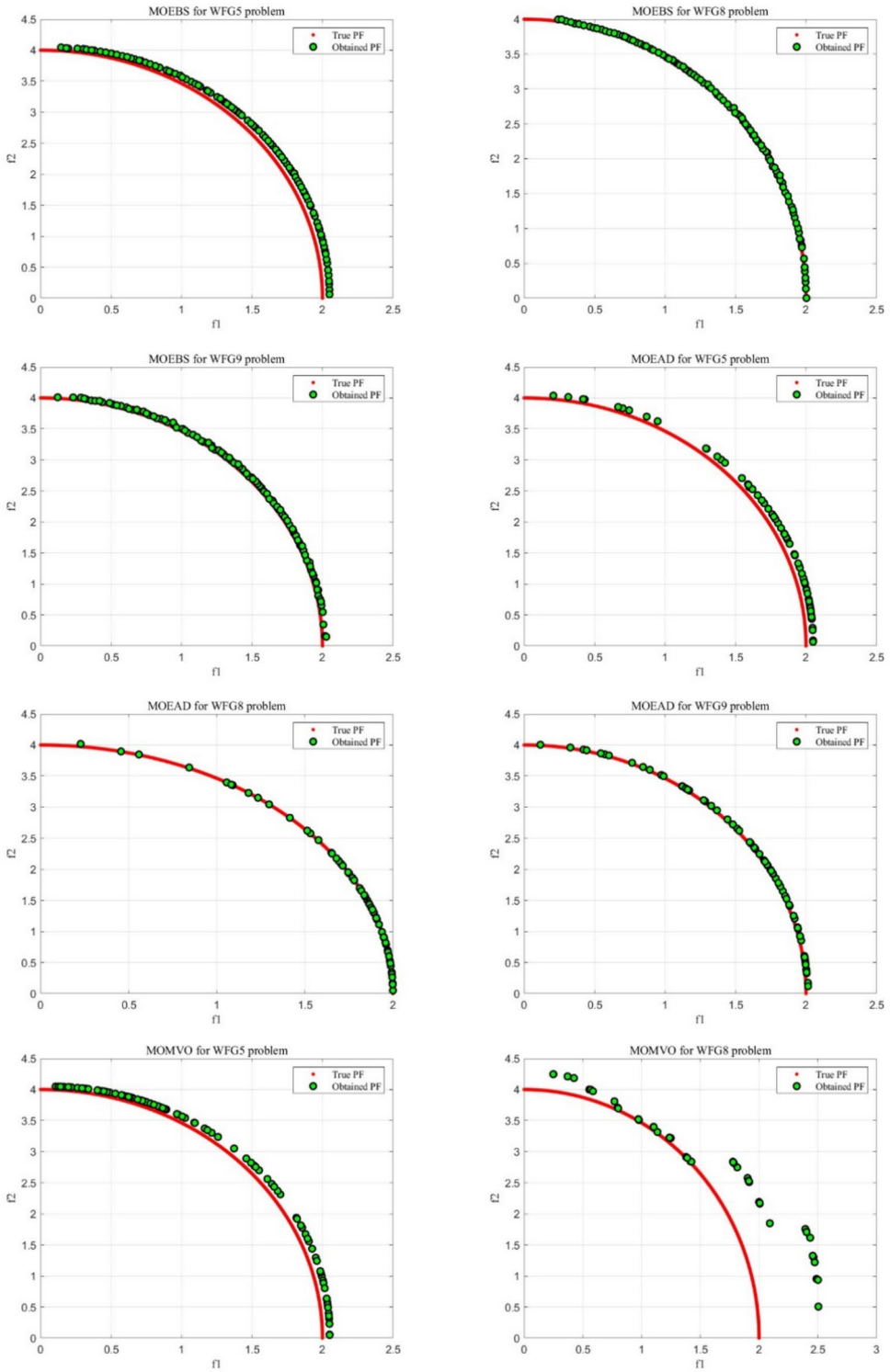

Fig. 11. Results of Pareto fronts produced by the algorithms in WFG5, WFG8 and WFG9.

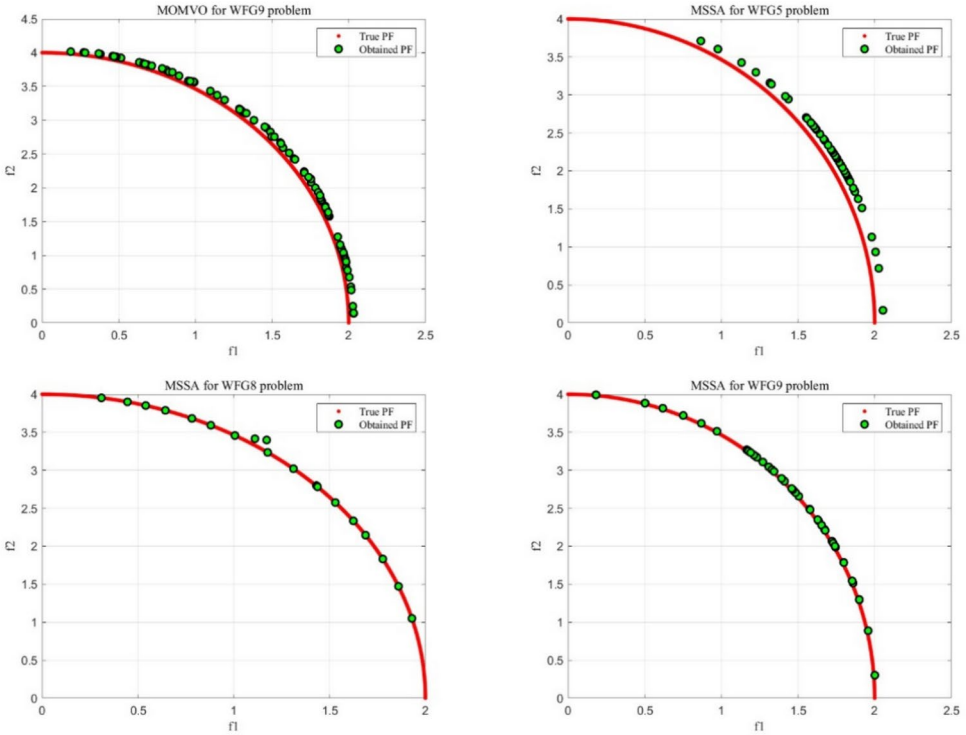

Figure 11. (continued)

$$GD = \frac{\sqrt{\sum_{i=1}^{N_{obt}} dis_i^2}}{N_{obt}} \tag{20}$$

where n_{pf} in PF_{known} represents the quantity of non-dominated solutions and dis_i represents the Euclidean distance from the solution i within PF_{known} to its nearest solution within in PF_{ture}. Notably, a lower GD value refers to better algorithm convergence.

Figure 19 shows a schematic diagram of the GD indicator. The dashed lines represent the Pareto fronts, the red dots represent the reference points, and the green dots stand for the generated solutions. The GD index measures the ability of the solution set to approach the Pareto front by computing the distance from the generated solution set to the closest Pareto front point. This distance between each green point and the nearest red point indicates the degree of deviation between the solution set and the Pareto front. The shorter the distance, the less the GD value, indicating the nearer the generated solution set is to the Pareto front, the better the algorithm performance.

Figure 20 present the box plot results of4 algorithms on ZTD, DTLZ and other 3 problem sets, analyzed through GD. Clearly, MOEBS manages to achieve the lowest median together with a narrow IQR throughout different problems, outperforming the other competitors in terms of effectiveness and consistency. MSSA and MOEAD have huge variations in terms of performance, showing that their abilities to tackle problems with different setup are limited. MOMVO has a similar performance to MOEBS, but with a higher median and wider IQR in general. Overall, MOEBS performs the best in terms of GD.

From Table 5, MOEBS performs well on the GD indicator on several functions, especially on the ZDT3, DTLZ6 and DTLZ7 problems. In the case of ZDT3, the GD average of MOEBS is only 0.000173 and the standard deviation is 0.000013, showing extremely high approximation ability and result consistency. Compared with the GD average of MOMVO, MSSA and MOEAD, which are 0.013133, 0.000724 and 0.000861, respectively. MOEBS was the best performer. For the DTLZ6 problem, the GD average of MOEBS is 0.000048 and the standard deviation is 0.000001, which once again proves its excellent performance and high stability in high-dimensional complex problems. The GD averages of other algorithms are 0.001612 (MOMVO), 0.074797 (MSSA) and 0.000051 (MOEAD), respectively. For the DTLZ7 problem, MOEBS has a GD mean of 0.004806 and a standard deviation of 0.003698, which is slightly higher than the other two functions, but still performs well, compared to MOMVO, MSSA, and MOEAD with GD averages of 0.010121, 1.216168, and 0.189027, respectively. Overall, MOEBS achieves the lowest mean on 8 problems sets, while MOMVO, MSSA, and MOEAD achieve the lowest on 1, 13, and 19 correspondingly. For the standard deviation, MOEBS achieves the lowest on 11 problem sets, while MOMVO, MSSA, and MOEAD achieve on 6, 12, and 12 correspondingly. These results highlight the

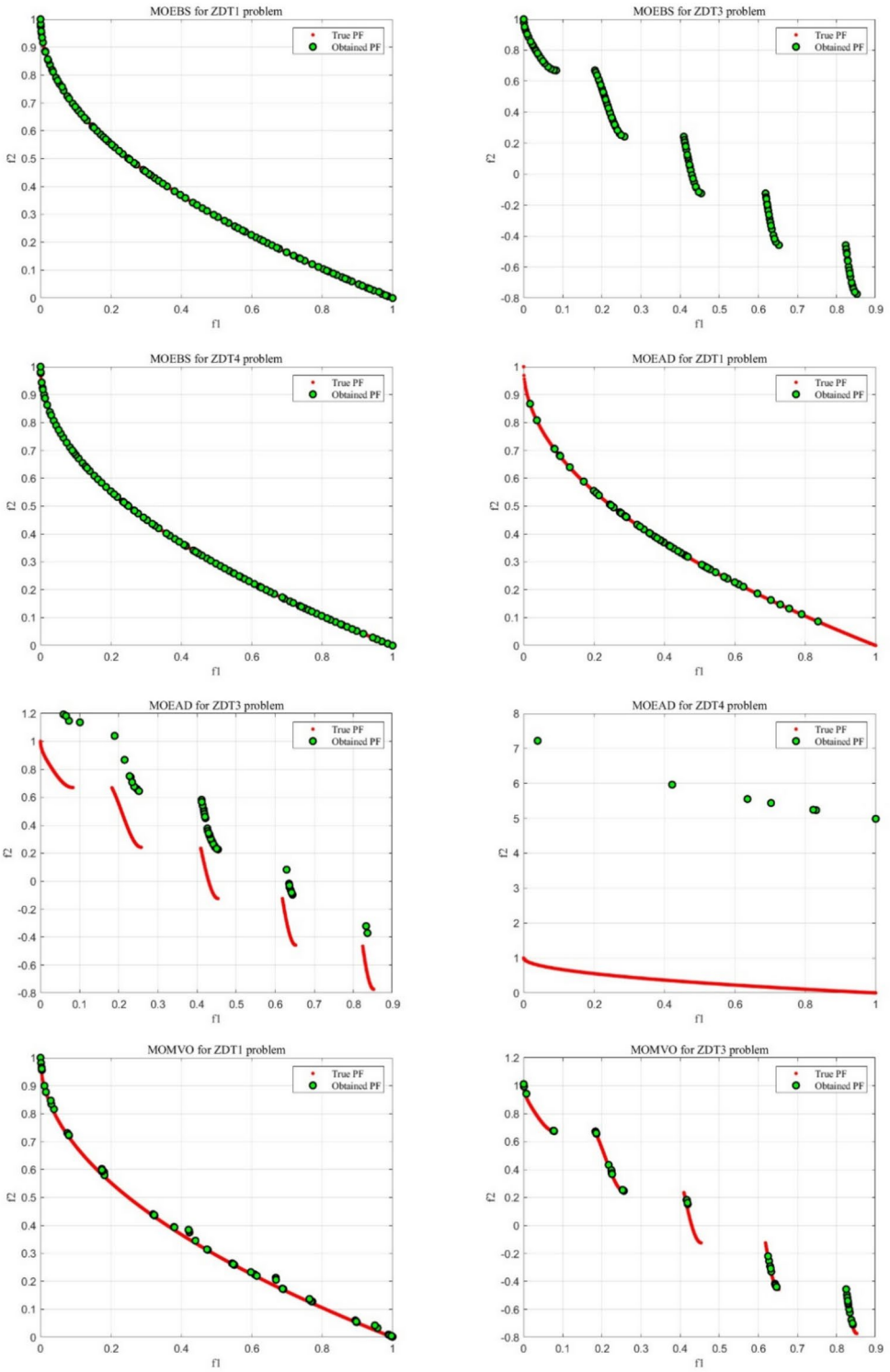

Fig. 12. Results of PF generated by all algorithms on ZDT1, ZDT3 and ZDT4 problems.

remarkable performance of MOEBS matching state-of-the-art algorithms in terms of performance in multi-objective optimization problems.

Wilcoxon signed-rank test

In this section, we apply the Wilcoxon signed-rank test to review the performance of MOEBS together with the other competing algorithms, and the results are presented in Table 6. When the p-value is less than 0.05, the algorithm is considered to have a significant difference from MOEBS. Conversely, when the p-value is more than 0.05, there is no significant difference between the two algorithms. The symbols '+/=/−' are used to indicate

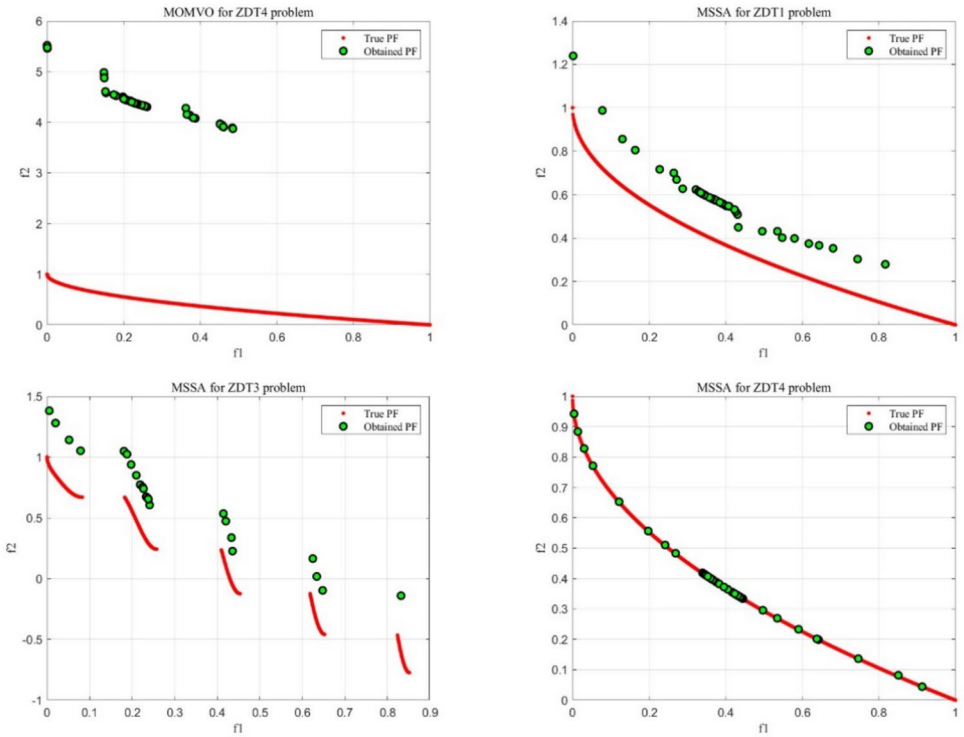

Figure 12. (continued)

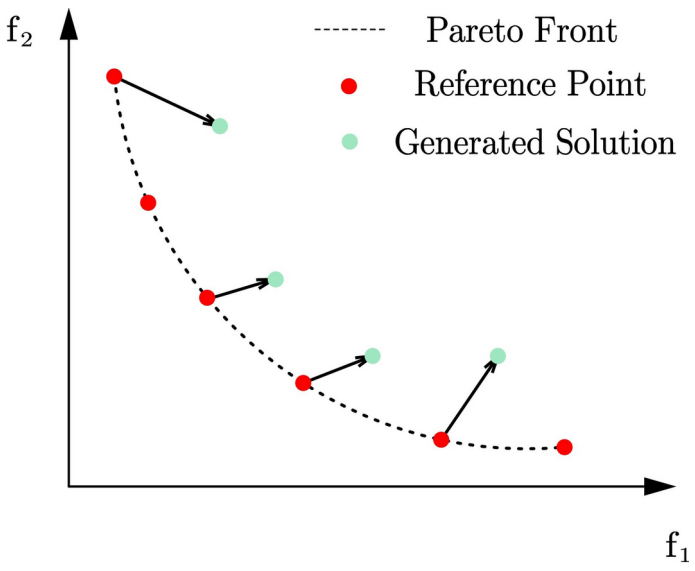

Fig. 13. Illustration of IGD.

whether MOEBS performs better than, similar to, or worse than the competitors. We can see clearly from the table that MOEBS distinguishes from other competitors.

The results of the Wilcoxon signed-rank test in Table 6 indicate that MOEBS significantly outperforms MOMVO, MSSA, and MOEAD in multi-objective optimization. MOEBS demonstrates exceptional performance across IGD, and HV metrics, particularly in ZDT1 and ZDT2 tests, where it often receives a "+" in IGD and HV metrics, reflecting superior convergence and solution diversity, although the numbers of "+" is less than the "-" in SP metrics, presenting . Compared to MOVIVO and MSSA, MOEBS achieves a significant level of $p < 0.05$ in most tests, maintaining stable performance, especially in high-dimensional problems, showcasing strong adaptability and robustness to dimensional changes. This indicates that MOEBS not only excels in accuracy but also consistently provides higher-quality solutions across different dimensions.

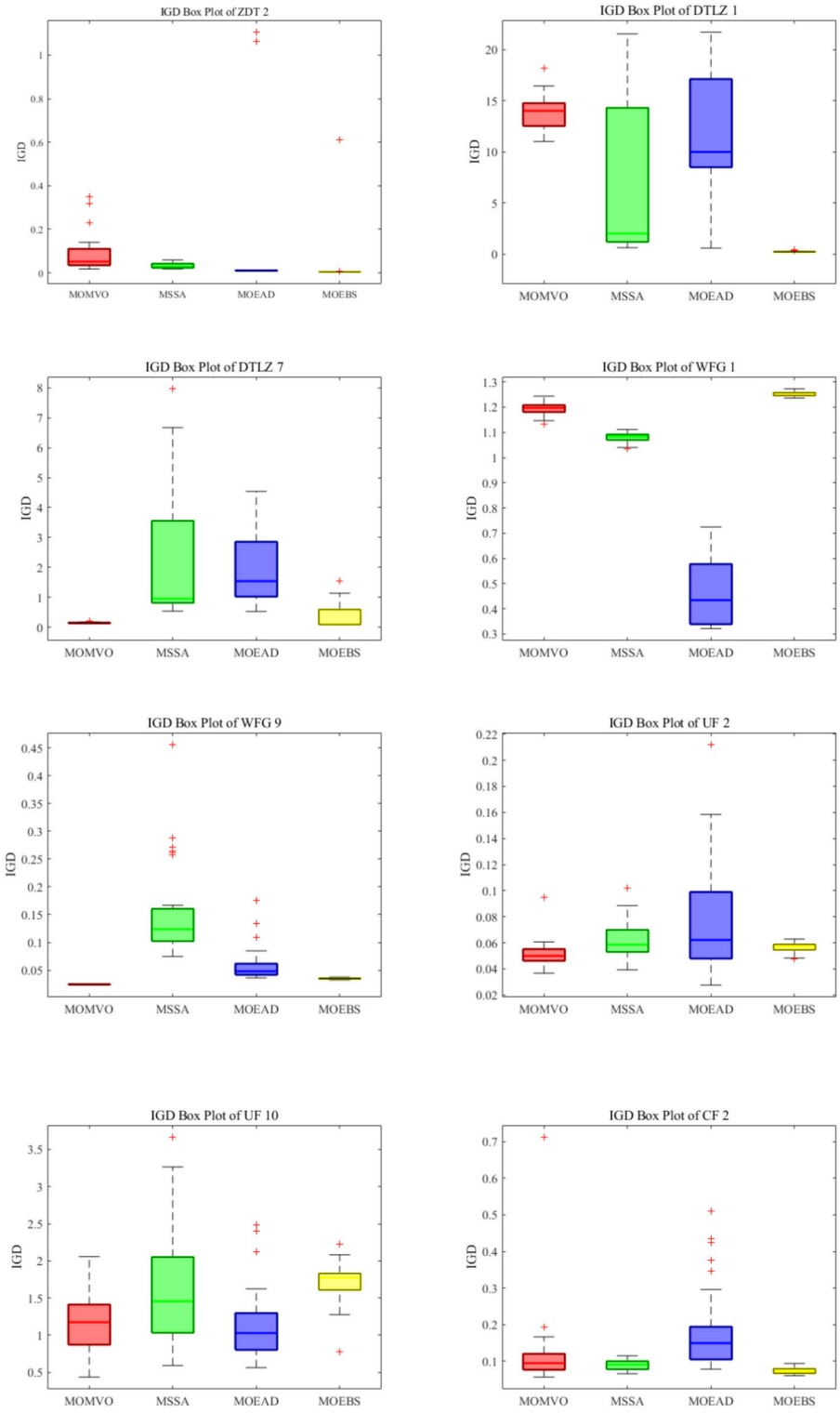

Fig. 14. IGD box plots on ZTD, DTLZ, WFG, UF, CF problem sets.

Review of classical time series models and experiments

To validate the exceptional predictive performance of Transformer in data forecasting, this section utilizes five classic deep learning models (CNN, LSTM, BiLSTM, GRU, and Transformer) to conduct experiments on the closing prices of Amazon, Google, and Uniqlo. The following provides an introduction to these models and details of the experiments.

Function	Index	Algorithms			
		MOMVO	MSSA	MOEAD	MOEBS
ZTD1	Ave	0.034440	0.036145	0.014992	**0.004415**
	Std	0.019453	0.022489	0.003816	**0.000150**
ZTD2	Ave	0.086307	**0.033347**	0.082131	0.085438
	Std	0.082219	**0.012045**	0.272187	0.209363
ZTD3	Ave	0.041801	0.048929	0.160673	**0.005024**
	Std	0.009261	0.018635	0.063397	**0.000173**
ZTD4	Ave	2.841079	**0.030175**	5.565712	0.099167
	Std	1.194689	**0.017705**	4.915656	0.220732
ZTD6	Ave	0.030915	0.038256	0.611892	**0.003504**
	Std	0.045077	0.009824	1.421691	**0.000139**
DTLZ1	Ave	13.900055	6.561541	12.001236	**0.227898**
	Std	1.700935	7.589842	5.696661	**0.057334**
DTLZ2	Ave	0.096030	0.515086	**0.095605**	0.144124
	Std	**0.003717**	0.053672	0.006077	0.005636
DTLZ3	Ave	171.983912	127.297020	99.703416	**1.092400**
	Std	15.512876	49.544354	31.084646	**0.899732**
DTLZ4	Ave	**0.101328**	0.478896	0.273288	0.311863
	Std	**0.016461**	0.075769	0.296550	0.022952
DTLZ5	Ave	**0.016308**	0.076750	0.018721	0.043753
	Std	**0.001453**	0.037157	0.006657	0.005717
DTLZ6	Ave	0.008472	0.229214	0.021451	**0.005199**
	Std	0.001761	0.257678	0.006880	**0.000359**
DTLZ7	Ave	**0.142876**	2.229548	1.975160	0.330376
	Std	**0.020811**	2.292365	1.184277	0.421176
WFG1	Ave	1.194508	1.078181	**0.480049**	1.252531
	Std	0.025156	0.019598	0.142749	**0.008824**
WFG2	Ave	**0.012753**	0.081545	0.230699	0.119857
	Std	**0.000371**	0.019035	0.098845	0.020505
WFG3	Ave	**0.020862**	0.095856	0.053628	0.056992
	Std	**0.000793**	0.044938	0.019707	0.004176
WFG4	Ave	0.071146	0.159762	0.083883	**0.068145**
	Std	**0.001301**	0.041413	0.006926	0.001778
WFG5	Ave	0.070823	0.146389	0.082806	**0.068266**
	Std	**0.001241**	0.024532	0.005708	0.002511
WFG6	Ave	**0.019000**	0.195409	0.092669	0.046923
	Std	**0.000730**	0.085854	0.042623	0.007463
WFG7	Ave	**0.018826**	0.217487	0.046163	0.070100
	Std	**0.000672**	0.121729	0.011614	0.007204
WFG8	Ave	0.288792	0.315038	0.313450	**0.192126**
	Std	**0.015836**	0.041839	0.048335	0.026495
WFG9	Ave	**0.024357**	0.154949	0.058489	0.035007
	Std	**0.000526**	0.083050	0.031103	0.001188
WFG10	Ave	-	-	-	-
	Std	-	-	-	-
UF1	Ave	0.104561	0.086867	0.193462	**0.057798**
	Std	0.030541	0.008830	0.103687	**0.004575**
UF2	Ave	**0.050684**	0.062865	0.079274	0.056547
	Std	0.010447	0.016307	0.043412	**0.003557**
UF3	Ave	0.385014	0.403765	0.521488	**0.379084**
	Std	0.122666	0.208383	0.141231	**0.020868**
UF4	Ave	0.082337	0.083878	0.066995	**0.047180**
	Std	0.016446	0.014167	0.007000	**0.000833**
UF5	Ave	0.780869	**0.523585**	1.382848	0.594331
	Std	0.435155	0.102178	0.436789	**0.066702**

Continued

Function	Index	Algorithms			
		MOMVO	MSSA	MOEAD	MOEBS
UF6	Ave	0.819757	**0.368385**	1.182833	0.663593
	Std	0.614589	**0.055798**	0.464841	0.071375
UF7	Ave	0.229151	**0.062527**	0.278494	0.064269
	Std	0.168087	0.010569	0.195417	**0.006072**
UF8	Ave	0.323449	0.473106	0.287660	**0.219423**
	Std	0.112769	0.218024	0.089306	**0.030612**
UF9	Ave	0.361449	0.414276	0.300519	**0.208904**
	Std	0.295050	0.134307	0.160803	**0.026607**
UF10	Ave	1.189647	1.648003	**1.136031**	1.708986
	Std	0.430803	0.765774	0.487824	**0.263868**
CF1	Ave	0.059137	0.055002	0.028703	**0.009913**
	Std	0.030240	0.024792	0.009816	**0.000847**
CF2	Ave	0.120547	0.089472	0.188416	**0.073170**
	Std	0.116284	0.013314	0.116604	**0.008292**
CF3	Ave	0.804863	**0.448749**	1.558204	1.101709
	Std	0.301899	**0.169498**	0.823029	0.194591
CF4	Ave	0.173440	**0.122328**	0.384216	0.182607
	Std	0.073443	**0.023531**	0.196412	0.029148
CF5	Ave	0.889445	**0.612942**	2.158531	2.227036
	Std	0.605289	**0.238444**	0.983707	0.517105
CF6	Ave	0.171473	**0.063997**	0.244773	0.104902
	Std	0.379792	0.019730	0.140145	**0.016918**
CF7	Ave	0.872307	**0.855281**	2.388138	2.168230
	Std	0.467168	**0.373669**	1.083100	0.439286
CF8	Ave	0.329859	0.931202	**0.255473**	0.401309
	Std	0.156250	0.645356	0.152450	**0.046121**
CF9	Ave	0.316760	0.524537	**0.183937**	0.198070
	Std	0.203406	0.259762	0.081837	**0.026895**
CF10	Ave	1.217310	2.026904	**1.175701**	1.613997
	Std	0.713157	1.221223	0.541634	**0.321126**

Table 2. Statistical results for IGD on ZDT, DTLZ, WFG, UF and CF problems.

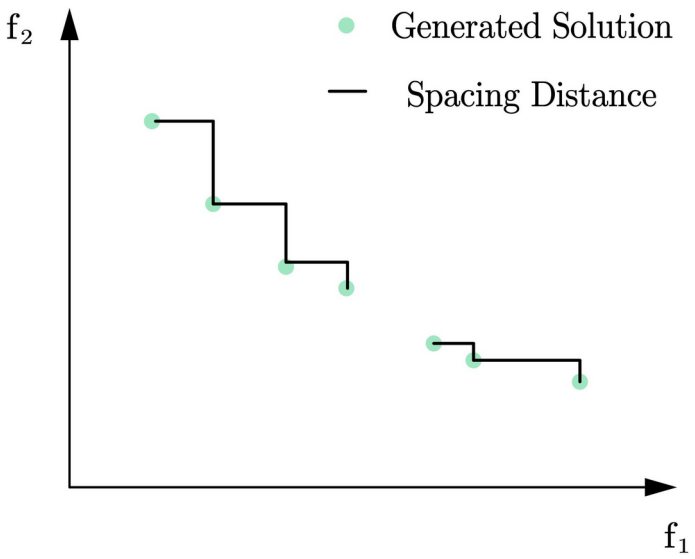

Fig. 15. Illustration of Spacing.

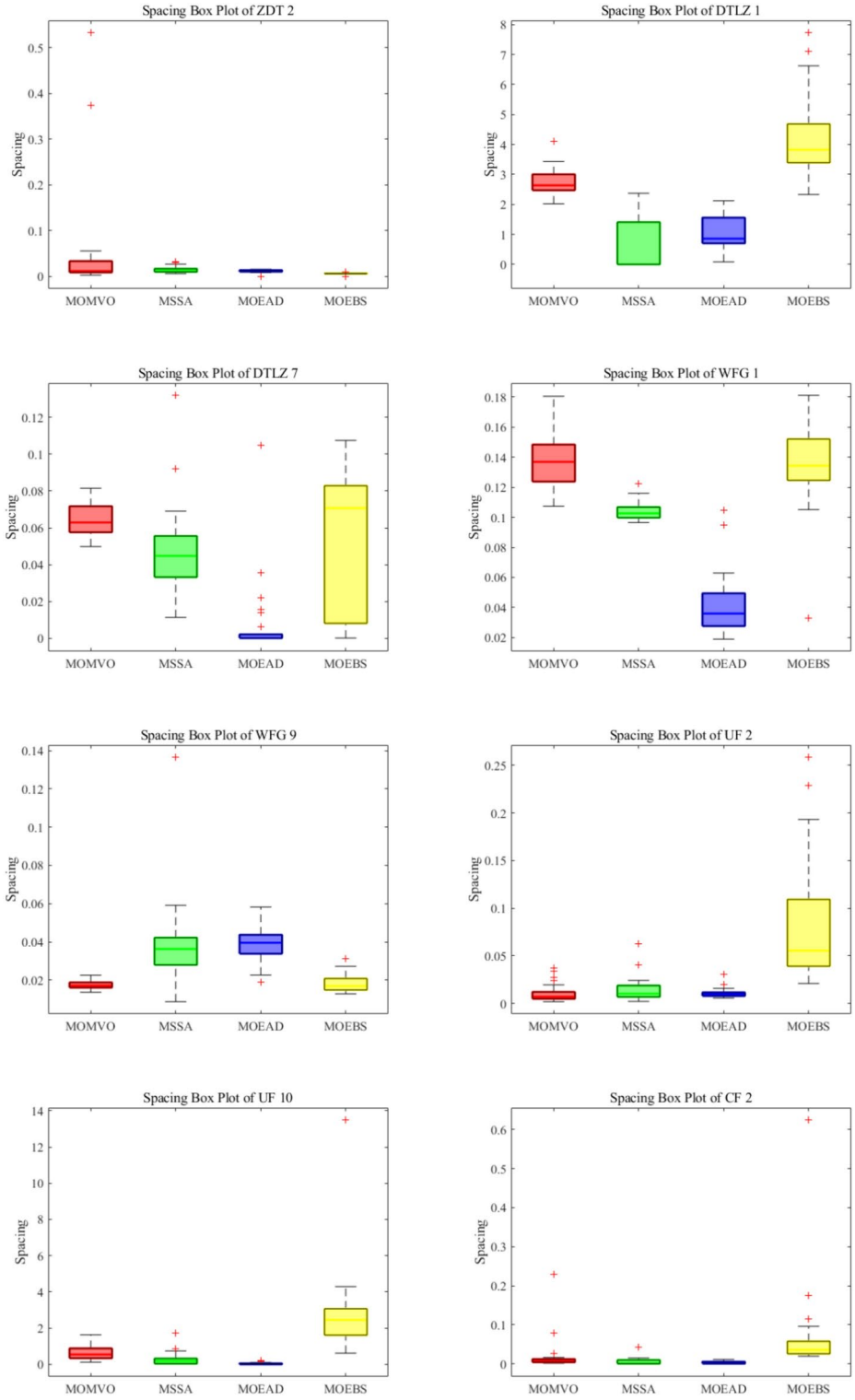

Fig. 16. Spacing box plots on ZTD, DTLZ, WFG, UF, CF problem sets.

Overview of classical time series models

CNN (Convolutional Neural Network)

CNN extracts local features from input data through convolution operations, where convolutional filters scan the input to identify local patterns, such as edges in images or local patterns in time series. After convolution, pooling layers reduce feature dimensions, improving computational efficiency and reducing overfitting. For time series data, CNN excels at efficiently capturing short-term dependencies and local patterns, making it suitable for tasks like signal processing or preprocessing speech data[41].

Function	Index	Algorithms			
		MOMVO	MSSA	MOEAD	MOEBS
ZTD1	Ave	0.026459	0.011731	0.013979	**0.006526**
	Std	0.083189	0.003861	0.004078	**0.000589**
ZTD2	Ave	0.049249	0.014362	0.011457	**0.005759**
	Std	0.117064	0.006560	0.003621	**0.002442**
ZTD3	Ave	0.108414	0.012892	0.014752	**0.007169**
	Std	0.193046	0.004540	0.007811	**0.000661**
ZTD4	Ave	1.048411	0.012032	0.105549	**0.006058**
	Std	2.843320	0.003810	0.429861	**0.003506**
ZTD6	Ave	0.044969	0.013930	**0.010304**	0.251907
	Std	0.041661	**0.004551**	0.004606	0.260172
DTLZ1	Ave	2.726574	**0.591521**	1.045438	4.199137
	Std	**0.456379**	0.807114	0.539217	1.343409
DTLZ2	Ave	0.068741	**0.012420**	0.067143	0.076190
	Std	0.012837	0.009472	0.007988	**0.007696**
DTLZ3	Ave	23.178768	**4.356206**	6.195430	28.465642
	Std	3.647969	6.338016	**2.380846**	8.496647
DTLZ4	Ave	0.063679	**0.023347**	0.048773	0.117965
	Std	**0.010617**	0.017199	0.027854	0.034067
DTLZ5	Ave	0.020626	0.014997	**0.014925**	0.036219
	Std	0.013399	0.007854	**0.003137**	0.014184
DTLZ6	Ave	0.011793	0.041339	0.015963	**0.008058**
	Std	0.012261	0.037144	0.004450	**0.000883**
DTLZ7	Ave	0.064497	0.046943	**0.006793**	0.056736
	Std	**0.008312**	0.026421	0.020207	0.035528
WFG1	Ave	0.138023	0.103604	**0.041242**	0.135123
	Std	0.018245	**0.005639**	0.020383	0.026284
WFG2	Ave	**0.013139**	0.027049	0.050835	0.121488
	Std	**0.001234**	0.011530	0.026267	0.047342
WFG3	Ave	**0.015524**	0.034422	0.045398	0.040556
	Std	**0.001380**	0.011515	0.017320	0.007244
WFG4	Ave	**0.018550**	0.032411	0.034418	0.021353
	Std	**0.001265**	0.017734	0.008099	0.002749
WFG5	Ave	**0.018758**	0.028893	0.040085	0.020727
	Std	**0.001819**	0.014732	0.009440	0.003985
WFG6	Ave	**0.019684**	0.038801	0.042859	0.035936
	Std	**0.001994**	0.014583	0.010125	0.011437
WFG7	Ave	**0.019608**	0.035109	0.041497	0.049148
	Std	**0.002232**	0.019626	0.012306	0.013961
WFG8	Ave	**0.031513**	0.035221	0.042582	0.104775
	Std	0.026971	0.015369	**0.009984**	0.044722
WFG9	Ave	**0.017388**	0.038483	0.039209	0.018282
	Std	**0.001930**	0.021641	0.009116	0.004334
WFG10	Ave	**0.024650**	0.037757	0.059506	0.038616
	Std	0.008543	0.014741	0.033912	**0.005922**
UF1	Ave	0.007880	0.004532	**0.001702**	0.044787
	Std	0.005658	0.007875	**0.002221**	0.046068
UF2	Ave	**0.010643**	0.014314	0.010836	0.086456
	Std	0.009050	0.012210	**0.004727**	0.066344
UF3	Ave	0.029637	0.002103	**0.001735**	0.029597
	Std	0.010066	0.005219	**0.003132**	0.060371
UF4	Ave	0.010837	0.009884	0.012177	**0.007954**
	Std	0.004303	0.010191	0.005080	**0.002267**
UF5	Ave	0.034639	0.002677	**0.002387**	0.366548
	Std	0.037212	0.010935	**0.004784**	0.296823

Continued

Function	Index	Algorithms			
		MOMVO	MSSA	MOEAD	MOEBS
UF6	Ave	0.040039	**0.000334**	0.002625	0.265813
	Std	**0.000000**	0.001168	0.007402	0.169029
UF7	Ave	0.007608	0.010135	**0.005436**	0.051190
	Std	**0.006473**	0.007378	0.007234	0.030607
UF8	Ave	0.260435	0.150251	**0.051048**	1.044351
	Std	0.152174	0.236871	**0.023417**	0.475604
UF9	Ave	0.213146	0.087874	**0.043289**	0.899606
	Std	0.146353	0.090730	**0.028973**	0.477132
UF10	Ave	0.627649	0.282037	**0.041046**	2.768668
	Std	0.385007	0.355436	**0.053757**	2.233361
CF1	Ave	0.029862	0.011287	0.016632	**0.004730**
	Std	0.035021	0.004153	0.009126	**0.000427**
CF2	Ave	0.018650	0.005358	**0.002823**	0.064963
	Std	0.045332	0.008679	**0.002844**	0.110851
CF3	Ave	0.199220	**0.000242**	0.002150	0.622956
	Std	**0.000000**	0.000744	0.004646	0.569364
CF4	Ave	0.012645	0.003991	**0.003735**	0.437485
	Std	0.016190	**0.006786**	0.009463	0.494674
CF5	Ave	-	**0.002202**	0.003613	1.441032
	Std	-	**0.004987**	0.010211	1.551803
CF6	Ave	0.019671	0.009398	**0.007727**	0.389627
	Std	0.013569	**0.005310**	0.011447	0.351685
CF7	Ave	0.284863	0.006070	**0.004083**	1.164972
	Std	0.307611	0.013523	**0.009015**	1.374945
CF8	Ave	0.295503	0.184859	**0.057122**	2.864021
	Std	0.241285	0.129685	**0.060418**	1.250903
CF9	Ave	0.229793	0.158875	**0.051472**	1.047098
	Std	0.211594	0.212562	**0.031089**	0.642722
CF10	Ave	0.695424	0.495872	**0.042921**	2.720145
	Std	0.484528	0.629724	**0.053518**	1.791807

Table 3. Statistical results for Spacing on ZDT, DTLZ, WFG, UF and CF problems.

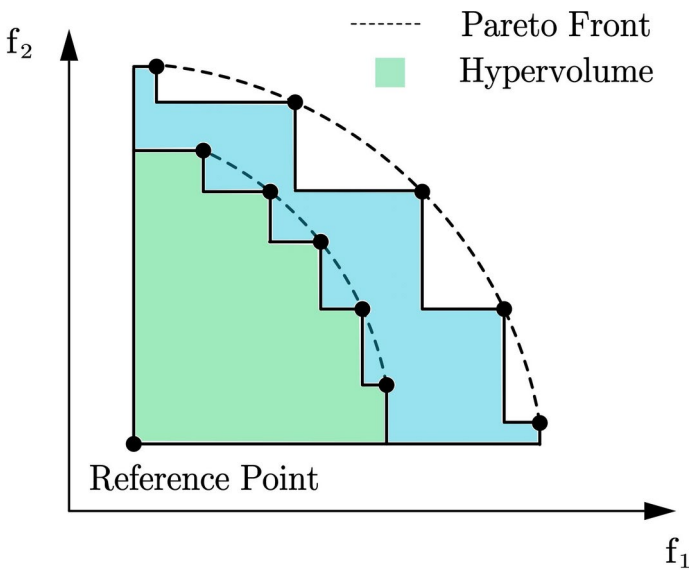

Fig. 17. Illustration of Hypervolume.

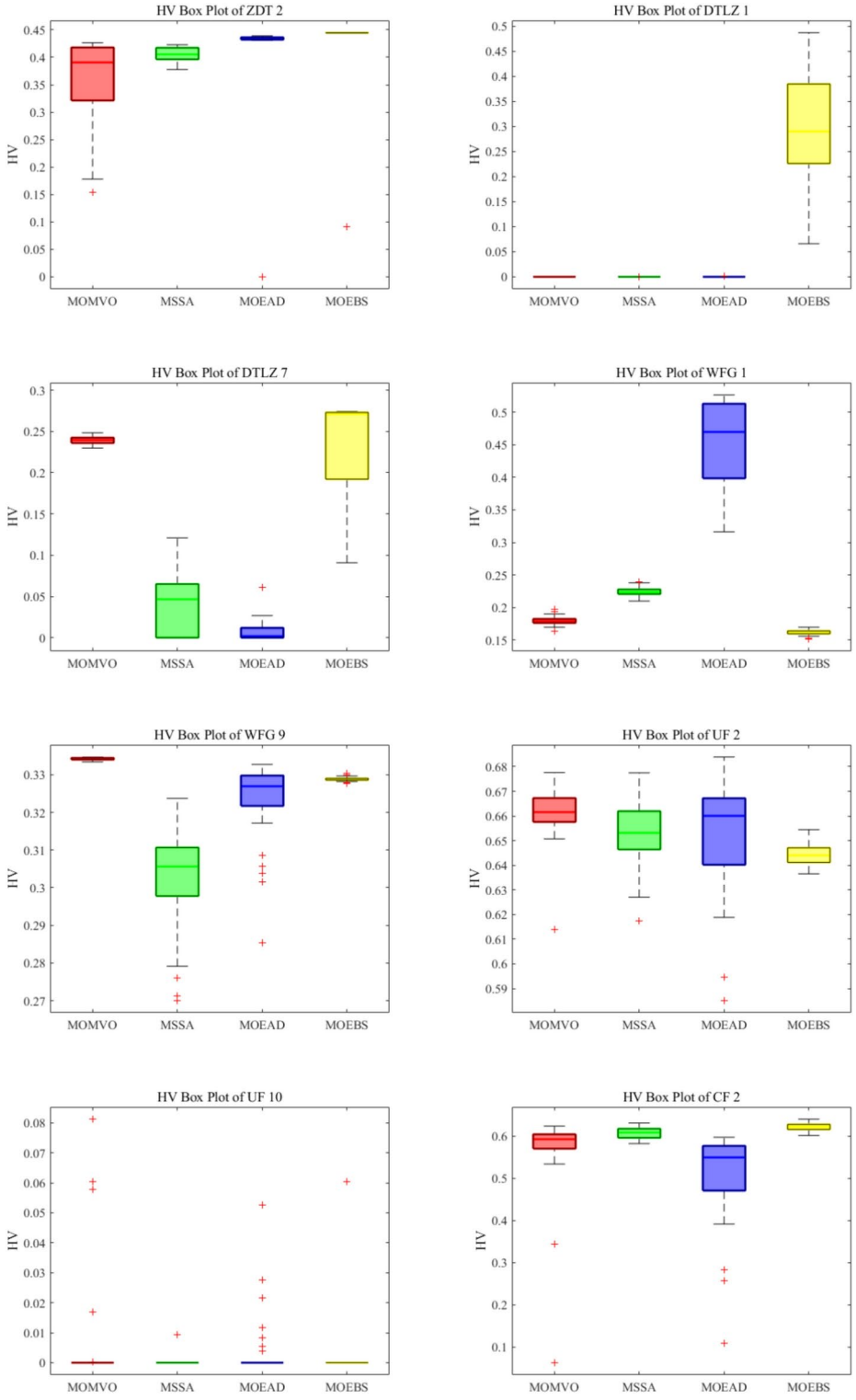

Fig. 18. HV box plots on ZTD, DTLZ, WFG, UF, CF problem sets.

LSTM (Long Short-Term Memory)

LSTM is a specialized type of Recurrent Neural Network (RNN) that introduces memory cells and gating mechanisms (input gate, forget gate, output gate) to address the vanishing and exploding gradient problems in RNNs when handling long sequences. LSTM can dynamically "remember" or "forget" information based on the context of the time series. Its advantage for time series data lies in its ability to capture long-term dependencies, making it particularly suitable for tasks with long-term trends or complex dynamics, such as financial forecasting or weather prediction[42].

Function	Index	Algorithms			
		MOMVO	MSSA	MOEAD	MOEBS
ZTD1	Ave	0.686230	0.685062	0.704805	**0.719928**
	Std	0.015423	0.013548	0.006465	**0.000160**
ZTD2	Ave	0.362063	**0.405689**	0.405681	0.397436
	Std	0.072847	**0.012457**	0.110297	0.122285
ZTD3	Ave	0.591822	0.607131	**0.630965**	0.599689
	Std	0.025364	0.031764	0.116692	**0.000049**
ZTD4	Ave	0.018066	**0.691777**	0.014366	0.635326
	Std	0.098954	**0.011229**	0.045645	0.181386
ZTD6	Ave	0.366671	0.354074	0.280317	**0.388432**
	Std	0.033737	0.010542	0.152333	**0.000174**
DTLZ1	Ave	0.000000	0.000019	0.000030	**0.289930**
	Std	**0.000000**	0.000106	0.000164	0.113692
DTLZ2	Ave	0.467605	0.155744	**0.482366**	0.370952
	Std	**0.006111**	0.030871	0.009680	0.014304
DTLZ3	Ave	0.000000	0.000000	0.000000	**0.044892**
	Std	**0.000000**	**0.000000**	**0.000000**	0.052638
DTLZ4	Ave	**0.471956**	0.172413	0.405899	0.246670
	Std	**0.009078**	0.068464	0.135200	0.024137
DTLZ5	Ave	0.188552	0.149717	**0.190727**	0.158261
	Std	**0.002727**	0.021678	0.003426	0.009824
DTLZ6	Ave	0.198429	0.120278	0.190753	**0.199921**
	Std	0.000918	0.052493	0.002664	**0.000104**
DTLZ7	Ave	**0.239070**	0.043963	0.008068	0.236801
	Std	**0.004652**	0.034243	0.013323	0.057025
WFG1	Ave	0.179734	0.224835	**0.449657**	0.162316
	Std	0.006643	0.006617	0.065032	**0.004276**
WFG2	Ave	**0.632009**	0.610690	0.582897	0.598928
	Std	**0.000184**	0.011784	0.038040	0.003971
WFG3	Ave	**0.575435**	0.545642	0.562098	0.551651
	Std	**0.000473**	0.014269	0.011420	0.002478
WFG4	Ave	0.307218	0.276108	0.299295	**0.309444**
	Std	**0.000813**	0.011274	0.002869	0.001946
WFG5	Ave	0.307347	0.277814	0.299437	**0.309425**
	Std	**0.000842**	0.008277	0.002638	0.002195
WFG6	Ave	**0.342728**	0.299335	0.298503	0.332103
	Std	**0.000387**	0.015668	0.026818	0.001968
WFG7	Ave	**0.342917**	0.301640	0.333019	0.319961
	Std	**0.000396**	0.020208	0.004920	0.003141
WFG8	Ave	0.230610	0.218296	0.223111	**0.275202**
	Std	**0.001283**	0.004593	0.011911	0.003871
WFG9	Ave	**0.334209**	0.301786	0.322683	0.328840
	Std	**0.000288**	0.014623	0.011093	0.000614
WFG10	Ave	-	-	-	-
	Std	-	-	-	-
UF1	Ave	0.579140	0.601722	0.501502	**0.637751**
	Std	0.039443	0.012769	0.102160	**6.39E-03**
UF2	Ave	**0.660648**	0.652072	0.651649	0.644338
	Std	0.010882	0.013527	0.024424	**0.004493**
UF3	Ave	**0.277772**	0.263218	0.154334	0.184904
	Std	0.095712	0.131883	0.095158	**0.021954**
UF4	Ave	0.329686	0.326198	0.351198	**3.81E-01**
	Std	0.024990	0.010472	0.007553	**9.36E-04**
UF5	Ave	**0.064055**	0.040627	0.001787	0.034291
	Std	0.091509	0.036415	**0.007969**	0.016986

Continued

Function	Index	Algorithms			
		MOMVO	MSSA	MOEAD	MOEBS
UF6	Ave	**0.085111**	0.063468	0.020479	0.005043
	Std	0.096838	0.047577	0.047438	**0.013174**
UF7	Ave	0.365735	**0.499108**	0.338771	0.490795
	Std	0.126134	0.013651	0.142285	**0.008748**
UF8	Ave	0.211063	0.137184	**0.348176**	0.267942
	Std	0.074527	0.083924	0.053842	**0.024516**
UF9	Ave	0.388752	0.289066	**0.533800**	0.523274
	Std	0.143226	0.117238	0.141379	**0.025166**
UF10	Ave	**0.007220**	0.000311	0.004367	0.002012
	Std	0.020604	**0.001704**	0.011246	0.011022
CF1	Ave	0.504159	0.515580	0.544397	**0.571171**
	Std	0.038217	0.019913	0.012924	**0.000927**
CF2	Ave	0.561729	0.606408	0.503415	**0.621633**
	Std	0.106605	0.012827	0.113904	**0.010027**
CF3	Ave	0.008409	**0.055941**	0.001530	0.000000
	Std	0.020207	0.048566	0.008381	**0.000000**
CF4	Ave	0.329668	**0.386241**	0.216169	0.299731
	Std	0.085657	0.038672	0.132168	**0.027712**
CF5	Ave	0.053203	**0.083845**	0.000000	0.000000
	Std	0.087696	0.102681	**0.000000**	**0.000000**
CF6	Ave	0.557032	**0.637813**	0.477376	0.556746
	Std	0.112673	**0.017710**	0.130093	0.019984
CF7	Ave	**0.115535**	0.081801	0.000105	0.000270
	Std	0.142142	0.129936	**0.000577**	0.001480
CF8	Ave	0.214723	0.035500	**0.341027**	0.107834
	Std	0.070768	0.046677	0.101179	**0.026536**
CF9	Ave	0.216850	0.124963	**0.355863**	0.272182
	Std	0.070791	0.073344	0.059448	**0.033606**
CF10	Ave	**0.018230**	0.000084	0.007331	0.000466
	Std	0.040314	**0.000459**	0.021554	0.002551

Table 4. Statistical results for HV on ZDT, DTLZ, WFG, UF and CF problems.

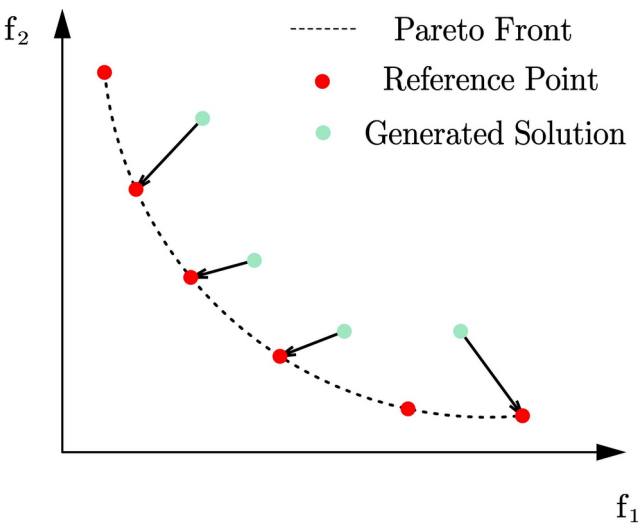

Fig. 19. Illustration of GD.

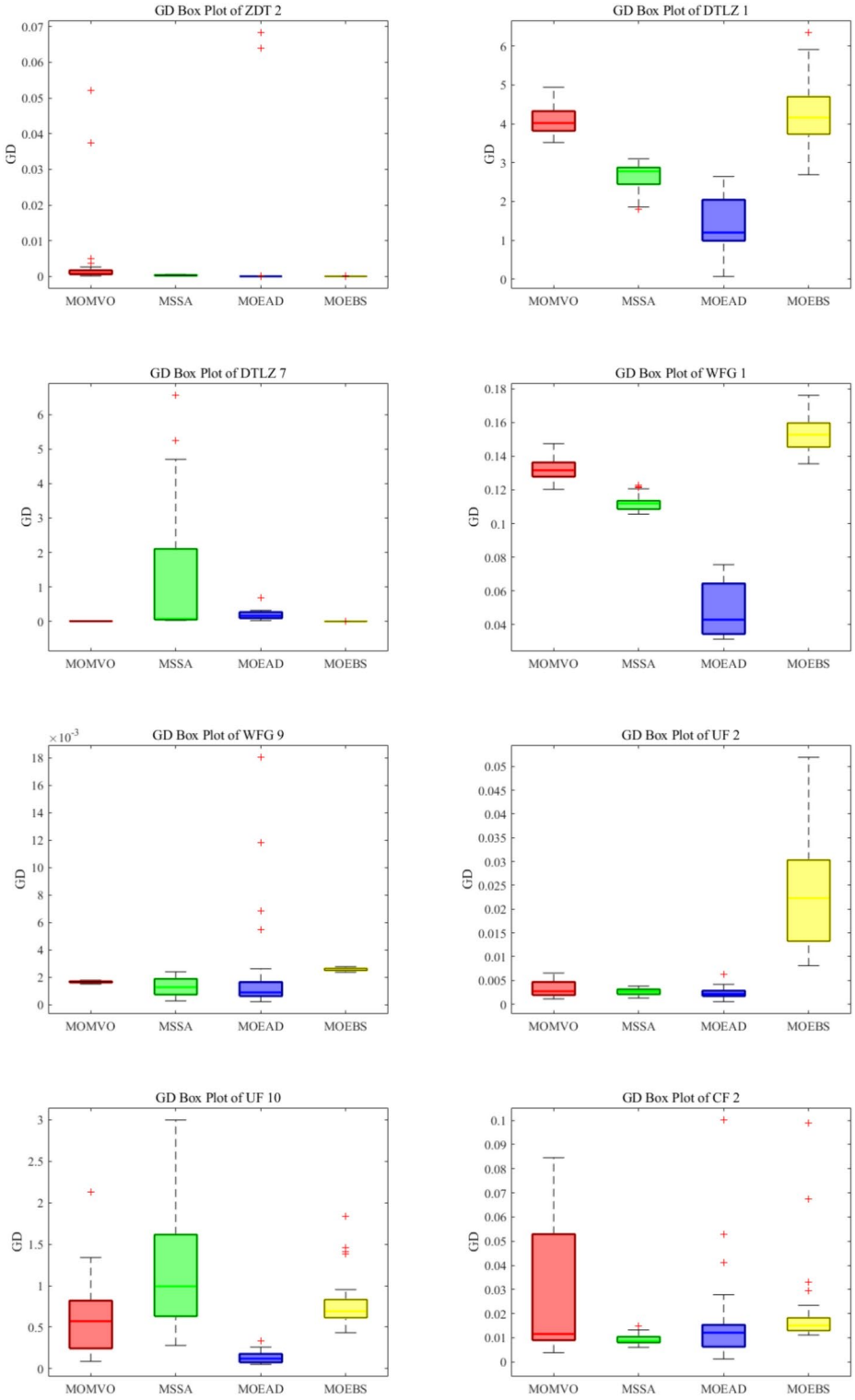

Fig. 20. GD box plot ZTD, DTLZ, WFG, UF, CF problem sets.

BILSTM (Bidirectional Long Short-Term Memory)
BiLSTM extends LSTM by considering both forward and backward information in a time sequence. It consists of two LSTM networks: one processes the sequence in the forward direction, and the other processes it in the backward direction, combining the outputs of both. For time series data, its advantage lies in leveraging both past and future information, making it ideal for tasks that require global context, such as speech recognition, sentiment analysis, or natural language processing[43].

Function	Index	Algorithms			
		MOMVO	MSSA	MOEAD	MOEBS
ZTD1	Ave	0.007311	0.000579	**0.000100**	0.000128
	Std	0.026964	0.000166	0.000304	**0.000038**
ZTD2	Ave	Inf	0.000317	0.004460	**0.000046**
	Std	-	0.000144	0.016777	**0.000023**
ZTD3	Ave	0.013133	0.000724	0.008361	**0.000173**
	Std	0.023095	0.000269	0.008672	**0.000013**
ZTD4	Ave	0.376748	**0.000101**	0.696499	0.003504
	Std	0.300608	**0.000081**	1.169920	0.009007
ZTD6	Ave	Inf	**0.021720**	0.057958	0.040231
	Std	-	**0.012892**	0.133683	0.038146
DTLZ1	Ave	4.088347	2.653486	**1.432663**	4.240839
	Std	0.367466	**0.329354**	0.686569	0.771908
DTLZ2	Ave	0.010576	0.002551	**0.001763**	0.024176
	Std	0.002362	0.000475	**0.000087**	0.003108
DTLZ3	Ave	31.323024	19.027741	**9.977502**	28.272374
	Std	2.804625	**1.285552**	3.113486	3.486306
DTLZ4	Ave	0.008240	0.007635	**0.002136**	0.023931
	Std	0.002107	0.006506	**0.001940**	0.003726
DTLZ5	Ave	0.002935	0.000273	**0.000053**	0.010469
	Std	0.001400	0.000262	**0.000004**	0.003642
DTLZ6	Ave	0.001612	0.074797	0.000051	**0.000048**
	Std	0.002353	0.060465	0.000003	**0.000001**
DTLZ7	Ave	0.010121	1.216168	0.189027	**0.004806**
	Std	**0.000883**	2.039051	0.133524	0.003698
WFG1	Ave	0.131864	0.112087	**0.047422**	0.153184
	Std	0.006688	**0.004636**	0.014953	0.009925
WFG2	Ave	**0.000618**	0.001364	0.005343	0.009001
	Std	**0.000046**	0.001726	0.007162	0.001923
WFG3	Ave	0.001659	**0.000517**	0.000728	0.005880
	Std	**0.000126**	0.000309	0.001948	0.000753
WFG4	Ave	0.006828	0.008514	**0.006090**	0.006386
	Std	0.000101	0.002145	0.000068	**0.000050**
WFG5	Ave	0.006808	0.008424	**0.006086**	0.006392
	Std	0.000089	0.002403	0.000077	**0.000045**
WFG6	Ave	0.001227	**0.000433**	0.006594	0.003003
	Std	**0.000117**	0.000248	0.004902	0.000585
WFG7	Ave	0.001190	**0.000294**	0.000413	0.005927
	Std	**0.000127**	0.000163	0.001361	0.000936
WFG8	Ave	0.031038	0.031160	0.036433	**0.018614**
	Std	0.003522	0.005538	0.005916	**0.002140**
WFG9	Ave	0.001659	**0.001278**	0.002247	0.002576
	Std	**0.000068**	0.000634	0.003821	0.000101
WFG10	Ave	-	-	-	-
	Std	-	-	-	-
UF1	Ave	Inf	0.006713	**0.005502**	0.012740
	Std	-	**0.002388**	0.004687	0.008327
UF2	Ave	0.003262	0.002633	**0.002277**	0.023603
	Std	0.001611	**0.000633**	0.001172	0.012259
UF3	Ave	Inf	0.022878	0.050354	**0.003661**
	Std	-	0.024477	0.019531	**0.007510**
UF4	Ave	Inf	0.007294	0.006311	**0.005266**
	Std	-	0.001150	0.000761	**0.000163**
UF5	Ave	Inf	**0.076117**	0.195456	0.253426
	Std	-	**0.051837**	0.143913	0.131954

Continued

Function	Index	Algorithms			
		MOMVO	MSSA	MOEAD	MOEBS
UF6	Ave	Inf	**0.037821**	0.140064	0.224622
	Std	-	**0.033419**	0.082148	0.057784
UF7	Ave	Inf	**0.004329**	0.007595	0.014005
	Std	-	**0.002386**	0.008498	0.005142
UF8	Ave	0.202417	0.189590	**0.013053**	0.158710
	Std	0.181711	0.154291	**0.014698**	0.058996
UF9	Ave	Inf	0.213821	**0.012957**	0.139524
	Std	-	0.179060	**0.007860**	0.049219
UF10	Ave	0.603707	1.202785	**0.130300**	0.796571
	Std	0.428549	0.789015	**0.066732**	0.320097
CF1	Ave	Inf	0.003505	0.004002	**0.002310**
	Std	-	0.000663	0.001418	**0.000045**
CF2	Ave	Inf	**0.009171**	0.016391	0.020758
	Std	-	**0.002102**	0.019414	0.018169
CF3	Ave	Inf	0.189425	**0.184857**	0.405579
	Std	-	0.170433	0.158937	**0.143713**
CF4	Ave	Inf	**0.021024**	0.023065	0.144272
	Std	-	0.027338	**0.022112**	0.099387
CF5	Ave	Inf	**0.145800**	0.304618	1.102925
	Std	-	**0.148872**	0.210324	0.529500
CF6	Ave	Inf	0.011936	**0.007172**	0.115879
	Std	-	0.014226	**0.008278**	0.078761
CF7	Ave	Inf	**0.212504**	0.262469	1.040095
	Std	-	0.179434	**0.129160**	0.631390
CF8	Ave	0.242208	0.310384	**0.015480**	0.600634
	Std	0.166704	0.176335	**0.016849**	0.157663
CF9	Ave	Inf	0.186359	**0.010750**	0.168844
	Std	-	0.165256	**0.007744**	0.076517
CF10	Ave	0.700196	1.157418	**0.138718**	0.872052
	Std	0.479630	0.719235	**0.081883**	0.253779

Table 5. Statistical results for GD on ZDT, DTLZ, WFG, UF and CF problems.

GRU

GRU is a simplified variant of LSTM that uses gating mechanisms to control the flow of information without a separate memory cell. It has two main gates: the update gate, which determines how much of the past information to retain, and the reset gate, which decides how much of the previous information to forget. Compared to LSTM, GRU has fewer parameters, making it more computationally efficient while maintaining similar performance for many tasks. For time series data, GRU's advantage lies in its ability to capture temporal dependencies with less computational complexity, making it suitable for tasks like stock price prediction, anomaly detection, or time series classification[66].

Transformer

Transformers rely on the self-attention mechanism to model relationships between elements in a sequence without relying on recurrence or convolution. Through a global attention mechanism, Transformer computes dependencies across all elements of the input sequence and use multi-head attention to capture features from different subspaces. For time series data, their advantage is the ability to efficiently capture both global and local temporal dependencies while supporting parallel computation, making them suitable for handling long sequences, such as power load forecasting or multivariate time series analysis[45].

Experimental methods and data

To verify the overall performance of the Transformer, the experiment selected stock closing price data from Amazon, Google, and Uniqlo. Meanwhile, Google, and Uniqlo were selected for the prediction experiments because they represent different types of markets: Amazon and Google are globally recognized tech companies, while Uniqlo is an international retail brand. Using stock price data from these companies helps evaluate the model's applicability and generalization ability across different market conditions. It then compared five deep learning models: CNN, LSTM, BiLSTM, GRU, and Transformer. Using these three representative datasets, the experiment evaluated how each model performed under different market dynamics. The three stock datasets can be accessed in the supplementary information files.

	IGD			SP			HV			GD		
MOEBS VS	MOMVO	MSSA	MOEAD	MOMVO	MSSA	MOEAD	MOMVO	MSSA	MOEAD	MOMVO	MSSA	MOEAD
ZTD1	+	+	+	=	+	+	+	+	+	=	+	=
ZTD2	=	=	=	+	+	+	=	=	=	+	+	=
ZTD3	+	+	+	+	+	+	=	=	=	+	+	+
ZTD4	+	=	+	+	+	=	+	=	+	+	-	+
ZTD6	+	+	+	-	-	-	+	+	+	+	-	=
DTLZ1	+	+	+	-	-	-	+	+	+	=	-	-
DTLZ2	-	+	-	-	-	-	-	+	-	-	-	-
DTLZ3	+	+	+	-	-	-	+	+	+	+	-	-
DTLZ4	-	+	=	-	-	-	-	+	-	-	-	-
DTLZ5	-	+	-	-	-	-	-	=	-	-	-	-
DTLZ6	+	+	+	=	+	+	+	+	+	+	+	+
DTLZ7	-	+	+	=	=	-	=	+	+	+	+	+
WFG1	-	-	-	=	-	-	-	-	-	-	-	-
WFG2	-	-	+	-	-	-	-	-	+	-	-	-
WFG3	-	+	=	-	-	=	-	+	-	-	-	-
WFG4	+	+	+	-	+	+	+	+	+	+	+	-
WFG5	+	+	+	-	+	+	+	+	+	+	+	-
WFG6	-	+	+	-	=	+	-	+	+	-	-	+
WFG7	-	+	-	-	-	-	-	+	-	-	-	-
WFG8	+	+	+	-	-	-	+	+	+	+	+	+
WFG9	-	+	+	=	+	+	-	+	+	-	-	=
WFG10	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
UF1	+	+	+	-	-	-	+	+	+	+	-	-
UF2	-	+	+	-	-	-	-	-	=	-	-	-
UF3	=	=	+	=	-	-	-	-	=	+	+	+
UF4	+	+	+	+	=	+	+	+	+	+	+	+
UF5	+	-	+	-	-	-	=	=	+	+	-	=
UF6	=	-	+	=	-	-	-	-	=	+	-	-
UF7	+	=	+	-	-	-	+	-	+	+	-	-
UF8	+	+	+	-	-	-	+	+	-	=	=	-
UF9	+	+	+	-	-	-	+	+	=	+	+	-
UF10	-	=	-	-	-	-	=	=	=	=	+	-
CF1	+	+	+	+	+	+	+	+	+	+	+	+
CF2	+	+	+	=	-	-	+	+	+	+	-	=
CF3	-	-	+	=	-	-	-	-	=	+	-	-
CF4	=	-	+	-	-	-	=	-	+	+	-	-
CF5	-	-	=	+	-	-	-	-	-	+	-	-
CF6	=	-	+	-	-	-	=	-	+	+	-	-
CF7	-	-	=	=	-	-	-	-	=	+	-	-
CF8	-	+	-	-	-	-	-	+	-	-	-	-
CF9	+	+	=	-	-	-	+	+	-	+	=	-
CF10	-	=	-	-	-	-	-	=	=	=	+	-
Overall	19/17/5	26/9/6	28/7/6	6/25/10	9/29/3	10/29/2	17/17/7	23/11/7	21/10/10	25/11/5	14/25/2	9/26/6

Table 6. Wilcoxon signed-rank test statistical results.

Data description of Amazon stock closing price

Amazon operates in the technology and e-commerce sectors, with highly volatile stock prices influenced by market demand, technological advancements, and global economic fluctuations. The dataset is in a time series format, with the first column being the date and the second column the closing price, reflecting price changes from historical to recent times.

Data description of Google stock closing price

Google is a leading global internet technology company, with its stock prices affected by technological innovations, advertising revenues, and regulatory policies. While the price trends are relatively stable, there can be significant fluctuations during specific periods. The dataset is also in a time series format, with two columns: date and closing price, showing price trends at different time points.

Data description of Uniqlo stock closing price
As an apparel retail company, Uniqlo's stock prices are greatly influenced by seasonal sales, market demand, and international trade conditions, showing cyclical variations. The Uniqlo dataset is also in a time series format, with two columns: date and closing price, helping assess the model's ability to predict cyclical and trend patterns.

The experimental methods and performance valuation metrics
In this experiment, five machine learning models—CNN, LSTM, BiLSTM, GRU, and Transformer—were applied to three datasets: Amazon, Google, and Uniqlo. Each model was trained and tested on the same data, using an 8/2 split between the training and testing sets to ensure consistent evaluation across all models and datasets. The key metrics for evaluating the accuracy of stock market prediction models typically include R^2 (coefficient of determination), RMSE (Root Mean Squared Error), and RPD (Ratio of Performance to Deviation). These metrics help quantify prediction errors and assess the model's fit and stability. The performance of each model was evaluated using three key metrics:

Root Mean Square Error (RMSE) is a common metric for the difference between the prediction of a model versus the observation. It is the square root of the mean of the squared differences between prediction and actual observation. It offers a way to evaluate the magnitude of prediction errors and is particularly sensitive to large errors. The RMSE on the training set reflects the fit degree of the model on the training data.

$$RMSE = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(Y_i - Y_{pred,i}\right)^2} \tag{21}$$

where M stands for the quantity of samples within the training data used to compute the mean of squared differences between the actual values. Y_i and the predicted values $Y_{ptrd,i}$ for each sample, indicating the model's accuracy on the training data.

Ratio of Performance to Deviation (RPD): RPD is the ratio of the standard deviation of the observed values to the RMSE of the predictions. This metric is used to assess the quality of predictive models, with higher values indicating better model performance. For RPD, we evaluate each candidate based on its performance on the test sample only, which can be calculated as below:

$$RPD = \frac{\sigma_Y}{\sqrt{\frac{1}{N}\sum\limits_{i=1}^{N}\left(Y_i - Y_{pred,i}\right)^2}} \tag{22}$$

where N represents the quantity of samples within the test set, and the denominator is the RMSE in the test set, while the numerator $\sigma_{Y_{test}}$ stands for the standard deviation of the test data. In addition, this metric on test sample is positively correlated to the RMSE, hence, a supplementary metric indicating the optimization effectiveness and the predictive power of the model.

Coefficient of Determination (R^2): R^2 denotes the proportion of variance within the dependent variable that can be explained by the independent variables. Its value typically ranges between 0 and 1, with higher values signifying that a greater portion of the variance is captured and explained, indicating better predictive power. For R^2 on the test set, the expression is formulated as below:

$$R^2 = 1 - \frac{\sum\limits_{i=1}^{N}\left(Y_i - \hat{Y}_i\right)^2}{\sum\limits_{i=1}^{N}\left(Y_i - \bar{Y}\right)^2} \tag{23}$$

where $Y_{test,i}$ is the actual value of i^{th} observation in the test sample, $\widehat{Y}_{test,i}$ is the predicted value of i^{th} observation in the test sample, $\overline{Y}_{test}$ is the mean of the observed values in the test set, and N represents the quantity of samples within the test set.

Experimental results and analysis

The comparison results of 5 classical time series models in Amazon
As shown in Fig. 21, the comparison results of five models on the Amazon dataset are presented, with evaluation metrics including RMSE, R^2, and RPD. The results indicate that the Transformer outperforms other models significantly on the test set, particularly in terms of prediction accuracy and fitting capability. It has the lowest RMSE on the test set, indicating higher prediction accuracy; an R^2 close to 1, suggesting better trend capture; and a higher RPD, reflecting stronger stability and generalization. In contrast, LSTM, GRU, and BiLSTM exhibit higher RMSEs, lower (or even negative) R^2 values on the test set, indicating limited fitting and prediction capabilities for stock price fluctuations. Additionally, other models have lower RPDs, suggesting that their

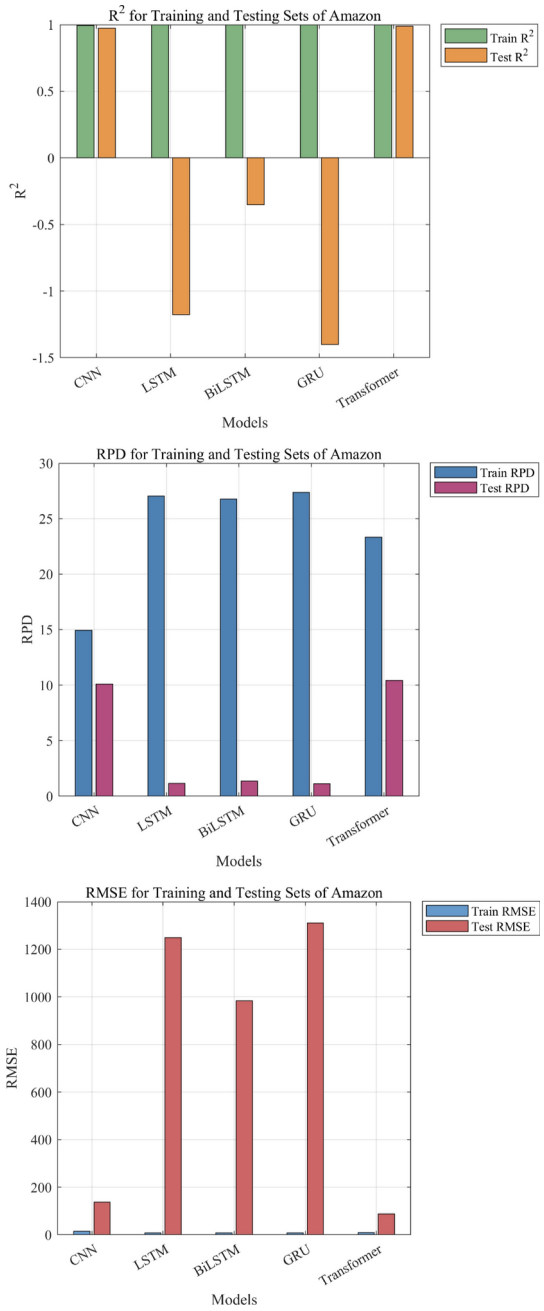

Fig. 21. Comparison of five model results on Amazon dataset.

stability and robustness are inferior to those of the Transformer. Therefore, the Transformer shows the best overall performance on the Amazon dataset.

Table 7 the comparison results of five models (CNN, LSTM, BiLSTM, GRU, and Transformer) on the Amazon stock dataset. On the training set, BiLSTM and GRU achieved lower RMSEs (7.6044 and 7.4646, respectively), which indicates a high degree of accuracy in fitting Amazon's stock price trends. However, all models display very high R_2 values (greater than 0.99) on the training set, suggesting potential overfitting, particularly for LSTM, BiLSTM, and GRU, which generally perform well on training but show limitations on the test set.

On the test set, the Transformer stands out with an RMSE of 88.2008 and the highest R_2 of 0.98914, indicating that it generalizes better to unseen data and can more accurately predict stock price trends in Amazon's highly volatile market. This suggests that the Transformer's architecture, with its multi-head self-attention mechanism, effectively captures long-term dependencies and complex time series patterns, allowing it to better handle the variability inherent in stock price movements.

In contrast, models like LSTM, BiLSTM, and GRU perform poorly on the test set, with negative R_2 values (-1.1789, -0.35191, and -1.4, respectively), highlighting a tendency toward overfitting. These negative R_2

Model	Training set			Test set		
	RMSE	R^2	RPD	RMSE	R^2	RPD
CNN	14.4468	0.99484	14.9258	136.2487	0.97407	10.0916
LSTM	**7.4371**	**0.99863**	27.0368	1249.0474	−1.1789	1.166
BiLSTM	7.6044	0.99857	26.7637	983.8625	−0.35191	1.3677
GRU	7.4646	0.99862	**27.3561**	1310.8926	−1.4	1.1248
Transformer	8.8969	0.99804	23.3262	**88.2008**	**0.98914**	**10.427**

Table 7. The comparison results of 5 models in Amazon.

values (−1.1789, −0.35191, and −1.4, respectively), highlighting a tendency toward overfitting. These negative R_2(0.97407), indicating limited capability in handling the complex dynamics of Amazon's stock prices.

Overall, the results underscore the Transformer's advantage in capturing long-term trends and intricate dependencies within time series data, making it a robust choice for stock prediction tasks in volatile sectors such as tech and e-commerce. This robustness is especially critical for industries like Amazon, where price movements are influenced by a combination of market dynamics, investor sentiment, and broader economic trends.

The comparison results of 5 classical time series models in Google

In Fig. 22, the results indicate that the Transformer outperforms other models on the test set, particularly in terms of prediction accuracy and robustness. It has a lower RMSE, indicating smaller prediction errors; an R^2 close to 1, demonstrating excellent data fitting capability; and a higher RPD, reflecting its advantages in stability and generalization. In contrast, LSTM and BiLSTM exhibit higher RMSEs and significantly reduced R^2 values on the test set, suggesting greater errors and overfitting when capturing Google's stock price fluctuations. Additionally, the GRU's performance on the test set is not as strong as the Transformer's, while the CNN performs the worst, with the highest RMSE and lowest R^2, showing weaker adaptability to complex time series. Therefore, the Transformer demonstrates the best overall performance on the Google dataset.

Table 8 shows that the Transformer performed the best on the training set, with the lowest RMSE (19.3469), the highest R^2 (0.9899), and an RPD of 10.1062. This indicates that the Transformer excels at capturing long-term trends and patterns in historical data. In comparison, the RMSEs of LSTM and BiLSTM are 19.6892 and 20.7555, respectively. While they also demonstrate strong fitting capabilities, their RPDs are slightly lower than the Transformer's, reflecting a relative weakness in extracting local features.

On the test set, the Transformer continues to exhibit excellent generalization ability, with a significantly reduced RMSE of 34.1159, an R^2 of 0.98845, and an RPD of 9.3472, indicating high accuracy in predicting unseen data. In contrast, the test performance of LSTM and BiLSTM declines sharply, with RMSEs of 241.7605 and 229.149, and R^2 values of only 0.42 and 0.47893, respectively, suggesting overfitting issues when handling the short-term fluctuations of Google's stock price. This result reveals the limitations of traditional recurrent neural networks in handling complex volatility, especially when facing irregular changes in time series data.

Moreover, like other major corporations, Google's stock price is generally stable but can experience sharp fluctuations during specific market events or policy changes, posing challenges for model adaptability. Compared to CNN, LSTM, and GRU, the Transformer's multi-head self-attention mechanism more comprehensively captures both global features and local variations within the time series, making it more stable when dealing with both short- and long-term fluctuations in Google's stock data. As a result, the Transformer demonstrates greater robustness and predictive capability in this highly volatile market environment.

The comparison results of 5 classical time series models in Uniqlo

In Fig. 23, the comparison results of five models on the Uniqlo dataset are presented, with evaluation metrics including RMSE, R^2, and RPD. The results show that the Transformer outperforms other models on the test set, particularly in terms of prediction accuracy and robustness. Its RMSE on the test set is relatively low, indicating smaller prediction errors; the R^2 is close to 1, demonstrating good fitting capability; and the RPD is higher, reflecting stability and generalization in predictions. In contrast, LSTM, GRU, and BiLSTM generally have higher RMSEs and slightly lower R^2 on the test set, indicating some errors when handling short-term stock price fluctuations. Additionally, CNN performs the worst on the test set, with the highest RMSE and lowest R^2, showing weaker prediction capability for Uniqlo's stock prices. Therefore, the Transformer exhibits the best overall performance on the Uniqlo dataset.

Table 9 shows the comparison results of five models (CNN, LSTM, BiLSTM, GRU, and Transformer) on the Uniqlo stock dataset. RMSE (Root Mean Square Error) measures the difference between the predicted and actual values, R^2 (Coefficient of Determination) evaluates how well the model fits the data variation, and RPD (Relative Percent Difference) reflects the stability of the model's prediction capabilities. On the training set, the Transformer performs the best, with the lowest RMSE (833.9759), the highest R^2 (0.99485), and an RPD of 13.9406, indicating that it captures more trends and patterns in Uniqlo's historical data with strong accuracy and stability. In comparison, LSTM and BiLSTM also perform well, but their RPDs are slightly lower, suggesting a weaker ability to capture local features.

On the test set, the Transformer remains ahead, with an RMSE of 990.1383, an R^2 of 0.95272, and an RPD of 4.599, indicating good generalization and prediction stability when dealing with unseen data. LSTM and GRU also achieve R^2 values of 0.94264 and 0.94424, respectively, showing decent accuracy, but their RMSEs

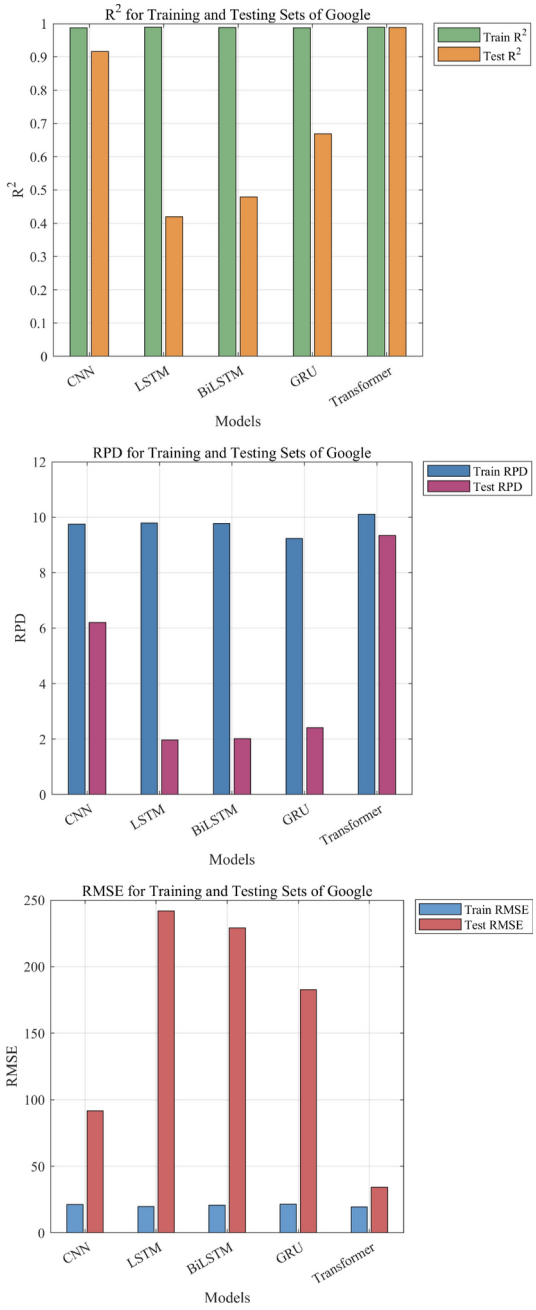

Fig. 22. Comparison of five model results on Google dataset.

Model	Training set			Test set		
	RMSE	R^2	RPD	RMSE	R^2	RPD
CNN	21.1861	0.98789	9.7586	91.7208	0.91652	6.2039
LSTM	19.6892	0.98954	9.7938	241.7605	0.42	1.9687
BiLSTM	20.755	0.98838	9.7837	229.149	0.47893	2.0146
GRU	21.5261	0.9875	9.2429	182.7175	0.6687	2.4149
Transformer	**19.3469**	**0.9899**	**10.1062**	**34.1159**	**0.98845**	**9.3472**

Table 8. The comparison results of 5 models in Google.

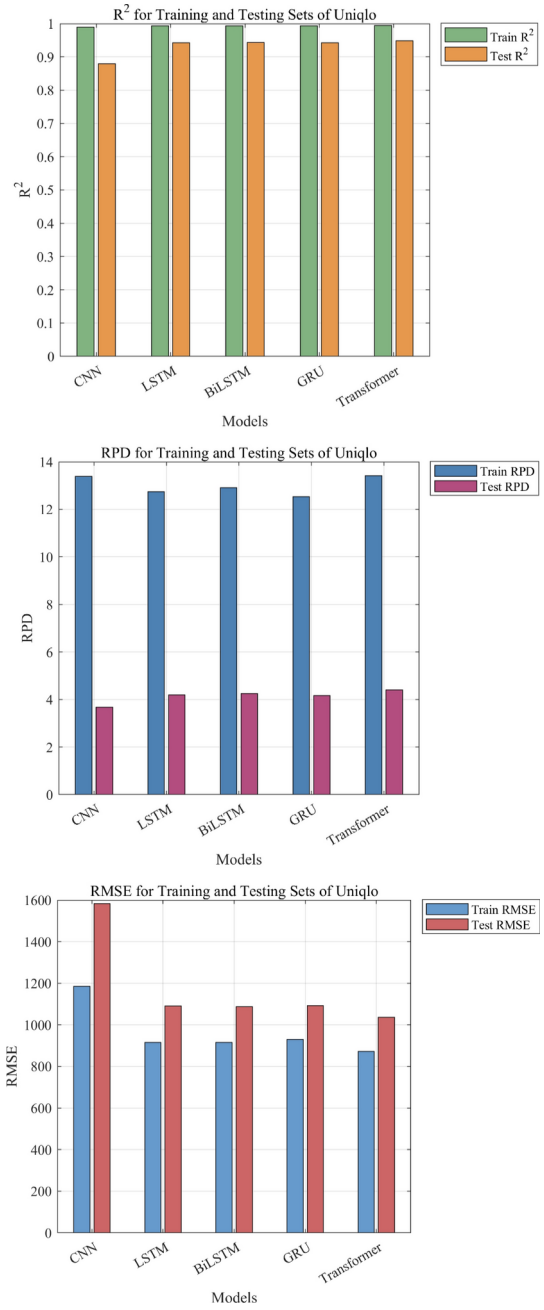

Fig. 23. Comparison of five model results on Uniqlo dataset.

Model	Training set			Test set		
	RMSE	R^2	RPD	RMSE	R^2	RPD
CNN	1185.6085	0.98959	13.3879	1582.3979	0.87924	3.6708
LSTM	915.8231	0.99379	12.7475	1090.5964	0.94264	4.1922
BiLSTM	915.9382	0.99379	12.9147	1088.0242	0.94291	4.2479
GRU	929.8816	0.9936	12.528	1092.4503	0.94244	4.1686
Transformer	**833.9759**	**0.99485**	**13.9406**	**990.1383**	**0.95272**	**4.599**

Table 9. The comparison results of 5 models in Uniqlo.

(1090.5964 and 1092.4503) and RPDs (4.1922 and 4.1686) are slightly higher than those of the Transformer, indicating slightly lower accuracy in handling short-term fluctuations. The Transformer demonstrates the most robust and accurate performance on the Uniqlo stock dataset. Its multi-head self-attention mechanism allows it to comprehensively capture both global features and local fluctuations in time series, showing stronger adaptability when dealing with complex market dynamics. While LSTM and GRU have strengths in capturing long- and short-term dependencies, their generalization capabilities are somewhat limited under complex market conditions. CNN, on the other hand, struggles with capturing long-term dependencies in time series, resulting in weaker performance.

Although the results of Transformer and the other four models are among the best, there is still room for improvement. For instance, optimizing and fine-tuning the hyperparameters of Transformer could significantly enhance its predictive performance. Section "Hyperparameter optimization experiments for the Transformer model" will provide a detailed comparison of the prediction results using different hyperparameter optimization methods, exploring further opportunities to improve the model's effectiveness.

Hyperparameter optimization experiments for the Transformer model

Hyperparameter optimization is crucial for deep learning models because the right hyperparameters can significantly improve the model's performance and training efficiency. Parameters such as learning rate, regularization coefficients, and model architecture directly affect the model's fitting ability and generalization capacity. To verify the powerful optimization performance of MOEBS, this section conducted hyperparameter optimization experiments on the Transformer model across three datasets: Amazon, Google, and Uniqlo, focusing on three key metrics: R^2, RMSE, and RPD. The experiments employed various methods, including Random Search (RS), Grid Search (GS), Bayesian Optimization (BO), as well as multi-objective optimization methods such as MOEBS-Transformer, MOMVO-Transformer, MOEAD-Transformer, and MSSA-Transformer. The results demonstrate that MOEBS significantly enhances the prediction accuracy and robustness of the Transformer model across all metrics. The key hyperparameters optimized in this study include:

1) **Number of attention heads** $(\mathrm{n_{head}})$: Defines the number of hidden units in the self-attention layers of the Transformer. More hidden units allow the model to learn more complex patterns, but it increases computational cost.
2) **Initial Learning Rate** (η_0) : The learning rate controls how much the model's weights are updated with respect to the loss gradient during training. The learning rate is crucial for model convergence.
3) **Regularization coefficient** (λ_{L2}) : L2 regularization coefficient is applied to prevent over- fitting by penalizing large weights.

Random search (RS) and its experimental procedures

Random Search (RS) is a widely used hyperparameter optimization technique due to its simplicity and effectiveness. In Random Search, hyperparameter values are randomly sampled from predefined ranges for a fixed number of iterations. For each trial, a model is trained using the sampled hyperparameters, and its performance is evaluated based on a chosen metric. The best performing set of hyperparameters is then selected.

In this section, we discuss how Random Search is applied to optimize the Transformer model for time series forecasting. Specifically, we focus on three critical hyperparameters: the number of hidden units, the initial learning rate, and the L2 regularization strength. The random search process consists of the following steps:

1) **Hyperparameter Sampling:** For each trial, the number of hidden units, the initial learning rate, and the L2 regularization strength are sampled randomly from their respective ranges.
2) **Model Training:** For each sampled hyperparameter set, a Transformer model is constructed and trained on the time series training data. The training uses the Adam optimizer with a maximum of 200 epochs and a mini-batch size of 256. The model is optimized to minimize the mean squared error (MSE) between the predicted and true values.
3) **Model Evaluation:** After training, the model is evaluated on a test set, and the root mean square error (RMSE) is calculated. The RMSE is used as the primary metric to assess the model's performance. The model with the lowest RMSE on the test set is selected as the best model.

Grid Search (GS) and its experimental procedures

Grid Search (GS) is a systematic hyperparameter optimization method that evaluates all possible combinations of hyperparameters within predefined ranges While Grid Search is computationally expensive compared to Random Search, it guarantees that all hyperparameter combinations are tested, ensuring the global optimum is found within the predefined grid.

In this section, we apply Grid Search to optimize the Transformer model for time series forecasting. The hyperparameters tuned include the number of hidden units, the initial learning rate, and L2 regularization strength. Grid Search proceeds as follows:

1) **Hyperparameter Grid Definition:** The hyperparameter grid is defined by specifying ranges for the number of attention heads, the initial learning rate, and regularization coefficient. For each combination of these values, a model is trained and evaluated.
2) **Model Training:** For each sampled hyperparameter set, a Transformer model is constructed and trained on the time series training data. The training uses the Adam optimizer with a maximum of 200 epochs and a mini-batch size of 256. The model is optimized to minimize the mean squared error (MSE) between the predicted and true values.

3) **Model Evaluation:** After training, the model is evaluated on a test set, and the root mean square error (RMSE) is calculated. The RMSE is used as the primary metric to assess the model's performance. The model with the lowest RMSE on the test set is selected as the best model.

Bayesian optimization (BO) and its experimental procedures

Bayesian optimization is an iterative strategy used for optimizing expensive objective functions. It is particularly effective for problems where the objective function is noisy, non-convex, or lacks an explicit mathematical form, such as hyperparameter tuning in machine learning models. The optimization procedure is based on building a probabilistic surrogate model of the objective function and using an acquisition function to guide the search for the optimal parameters.

Surrogate model

At the core of Bayesian optimization is the surrogate model, typically a Gaussian Process (GP). The GP is used to model the distribution of the objective function $f(\theta)$ based on observed data. Formally, given a set of observed evaluations $D = \{(\theta_1, f(\theta_1)), \ldots, (\theta_n, f(\theta_n))\}$, the GP models the objective function $f(\theta)$ as a sample from a multivariate normal distribution:

$$f(\theta)\ \mathcal{GP}(\mu(\theta), k(\theta, \theta')) \tag{24}$$

where, $\mu(\theta)$ is the mean function, representing the expected value of the objective function at pointθ, (θ, θ') is the covariance (kernel) function, which quantifies the correlation between the objective values at points θ and θ' The kernel function $k(\theta, \theta')$ plays a critical role in determining how smoothly the objective function is modeled and the degree to which information about $f(\theta)$ at one point affects the predictions at nearby points.

Posterior distribution

As new points θ are evaluated and their corresponding objective values $f(\theta)$ are observed, the GP is updated. The posterior distribution of $f(\theta)$,conditioned on the observed data D, provides both a mean estimate $\mu_n(\theta)$ and an uncertainty measure $\sigma_n^2(\theta)$ for the objective function at any new point θ :

$$f\ (\theta)|\ D\ \mathcal{N}(\mu_n(\theta), \sigma_n^2(\theta)) \tag{25}$$

where the posterior mean $\mu_n(\theta)$ represents the best estimate of the objective function at θ given the data, and the posterior variance $\sigma_n^2(\theta)$ reflects the uncertainty about $f(\theta)$ in regions of the parameter space that have not been well-explored.

Acquisition function

To decide where to evaluate the objective function next, Bayesian optimization uses an acquisition function $a\left(\theta; \widetilde{D}\right)$. The acquisition function balances two objectives :

1) Exploration: Focusing on regions where the uncertainty $\sigma_n(\theta)$ is high.
2) Exploitation: Focusing on regions where the mean estimate $\mu_n(\theta)$ is low, suggesting that good solutions may be found. A common acquisition function is the Expected Improvement (EI), which evaluates the expected amount by which a new evaluation at θ will improve upon the current best objective value $f(\theta^*)$:

$$EI(\theta) = E[\max(0, f(\theta^*) - f(\theta))] = \sigma_n(\theta)\,(z\Phi(z) + \phi(z)) \tag{26}$$

where:

$$z = \frac{f(\theta^*) - \mu_n(\theta)}{\sigma_n(\theta)},$$

where $\Phi(z)$ is the cumulative distribution function of the standard normal distribution. $\phi(z)$ is the probability density function of the standard normal distribution. $\sigma_n(\theta)$ is the posterior standard deviation at θ, $f(\theta^*)$ is the current best observed value of the objective function. The next evaluation point $\theta_n + 1$ is chosen to maximize the acquisition function:

$$q_{n+1} = \arg\max_q a(q; D) \tag{27}$$

Bayesian optimization procedure

The Bayesian optimization process proceeds iteratively as follows:

1) **Initialization:** Evaluate the objective function $f(\theta)$ at a small set of initial points $\theta_1, \ldots, \theta_n$.
2) **Fit the surrogate mModel:** Using the observed data $D = \{(\theta_i, f(\theta_i))\}_{i=1}^n$, fit a Gaussian Process to model the objective function.
3) **Maximize acquisition function:** Identify the next point $\theta_n + 1$ by maximizing the acquisition function $a(\theta; D)$.

4) **Evaluate the Objective Function:** Compute $f(\theta_{n+1})$ and update the data set $D = D \cup \{(\theta_{n+1}, f(\theta_{n+1}))\}$.
5) **Update the surrogate model**: Update the Gaussian Process with the new data and repeat the process until a stopping criterion is met (e.g., a fixed number of iterations or a convergence threshold).

Experimental optimization of the transformer model using four multi-objective algorithms

This section introduces the architecture of MOEBS-Transformer and compares it with three other Transformer models that utilize multi-objective optimization algorithms for hyperparameter tuning: MOMVO-Transformer, MOEAD-Transformer, and MSSA-Transformer. Additionally, it presents the objective functions employed in the optimization process.

Optimization algorithms for comparison

- Transformer with hyperparameters optimized by MOMVO (MOMVO-Transformer)
- Transformer with hyperparameters optimized by MOMVO (MOEAD-Transformer)
- Transformer with hyperparameters optimized by MSSA (MSSA-Transformer)

Furthermore, MOEBS differs from MOMVO , MSSA, and MOEAD in several key aspects. MOEBS has unique advantages in multi-objective optimization, particularly in its global search capability and dynamic adaptability. While MOMVO, MSSA, and MOEAD are also powerful multi-objective optimization algorithms, they differ in exploration abilities and convergence speed. MOEBS excels in maintaining a better balance between multiple objectives and preserving the non-dominated solution set in multi-objective optimization.

Multi-objective algorithms-transformer model optimization objective functions
Here, we briefly introduce the three-objective function MOEBS is targeting on:

Root Mean Square Error (RMSE) is a common metric for the difference between the prediction of a model versus the observation. It is the square root of the mean of the squared differences between prediction and actual observation. It offers a way to evaluate the magnitude of prediction errors and is particularly sensitive to large errors. The RMSE on the training set reflects the fit degree of the model on the training data.

$$RMSE_{train} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(Y_{train,i} - Y_{pred,train,i}\right)^2} \tag{28}$$

where M stands for the quantity of samples within the training data used to compute the mean of squared differences between the actual values. $Y_{train,i}$ and the predicted values $Y_{ptrd,train,i}$ for each sample, indicating the model's accuracy on the training data.

The RMSE on the test set measures the model's capability of generalization to new data. A good model not only needs to perform well on training sets, but more importantly, it needs to be able to perform consistently on previously unseen data.

$$RMSE_{test} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(Y_{test,i} - Y_{pred,test,i}\right)^2} \tag{29}$$

where N represents the quantity of samples within the test set, and the formula uses $Y_{train,i}$ and $Y_{pred,train,i}$ to compute the mean value of squared discrepancies between the actual and predicted values for each sample in the test set, assessing the model's capability of generalization.

The Variance of the test error refers to the consistency of the model's effectiveness over multiple tests. The less the variance, the more stable the output of the model, and generally the stronger the generalization ability

$$Var_{test} = Var(Y_{test} - Y_{pred,test}) \tag{30}$$

where Var_{test} utilizes $Y_{pred,train,i}$ and $Y_{pred,test,i}$ containing all actual and predicted values for the test set, to calculate the variance of the test prediction errors, which measures the stability and reliability of the model's performance across different sets of test data.

To ensure the model's generalization capability, we ultimately selected the solution from the Pareto front with the lowest test set RMSE. The corresponding parameters of this solution are considered the optimal hyperparameters identified by the multi-objective optimization algorithm. This selection approach balances the model's performance across different objectives, ensuring stability and predictive accuracy in practical applications.

Systematic analysis of the comparison results for different hyperparameter optimization methods

To begin with, the MOEBS-Transformer architecture is proposed, incorporating multi-objective optimization into the hyperparameter optimization process of the Transformer model.The framework for MOEBS-Transformer is shown in Fig. 24 To validate the optimization performance of MOEBS, this chapter presents a systematic analysis

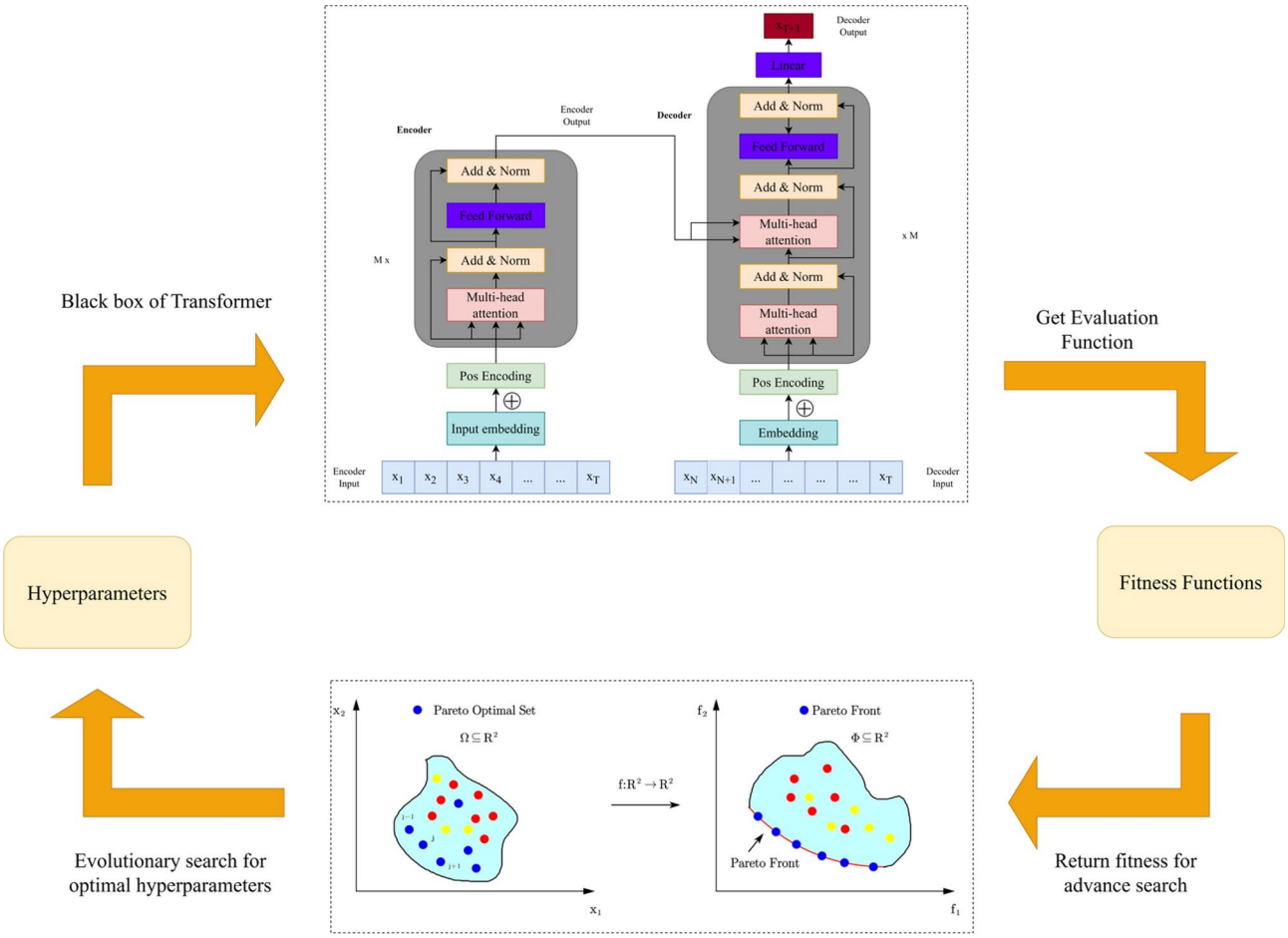

Fig. 24. The framework of MOEBS-Transformer.

of the results obtained from different hyperparameter optimization methods applied to the Transformer model. The analysis centers around three core metrics—R^2, RMSE, and RPD—evaluated across three dataset dimensions (The detailed explanation of these three-evaluation metrics is presented in Sect.5.3). The models under evaluation include the MOEBS-Transformer, which incorporates multi-objective optimization, followed by the MOMVO-Transformer (MOMVO-optimized Transformer), MSSA-Transformer (MSSA-optimized Transformer), and MOEAD-Transformer (MOEAD-optimized Transformer). Additionally, Transformer models optimized using traditional methods, including RS-Transformer (Random Search-optimized Transformer), GS-Transformer (Grid Search-optimized Transformer), and BO-Transformer (Bayesian Optimization-optimized Transformer), are also evaluated. This in-depth comparison aims to determine the effectiveness of each optimization method in enhancing the Transformer's accuracy, robustness, and generalization, providing insights into their performance across various datasets.

The comparison results of transformer models optimized by different algorithms in Amazon dataset

As shown in Fig. 25, the performance of the Transformer model optimized by different algorithms on the Amazon dataset varies significantly. For the R^2 metric, the MOEBS-Transformer achieves the highest fit in both the training and test sets, demonstrating the best fitting capability. MOEAD-Transformer and MOMVO-Transformer also perform well in R^2, just slightly below MOEBS-Transformer, indicating that multi-objective optimization algorithms effectively enhance model fitting performance. In contrast, GS-Transformer and RS-Transformer exhibit noticeably lower R^2, especially on the test set, indicating their inability to effectively capture the global features of the data. Additionally, in terms of RMSE, the MOEBS-Transformer has the lowest test set RMSE (52.5328), highlighting its strong ability to improve model prediction accuracy. MOMVO-Transformer and MOEAD-Transformer also show lower RMSE, while GS-Transformer and RS-Transformer have higher RMSE, particularly with GS-Transformer reaching 346.4594 on the test set, reflecting the limitations of traditional methods when handling complex datasets.

From Table 10, the MOEBS-Transformer also demonstrates a significant advantage in RPD, achieving 16.1724 on the test set, indicating excellent stability and generalization. BO-Transformer performs well in RPD, achieving 36.2845 on the training set, reflecting the potential of Bayesian optimization in enhancing model stability. MOMVO-Transformer and MOEAD-Transformer also perform well in RPD on the test set, though still slightly below MOEBS-Transformer. On the other hand, GS-Transformer and RS-Transformer have the

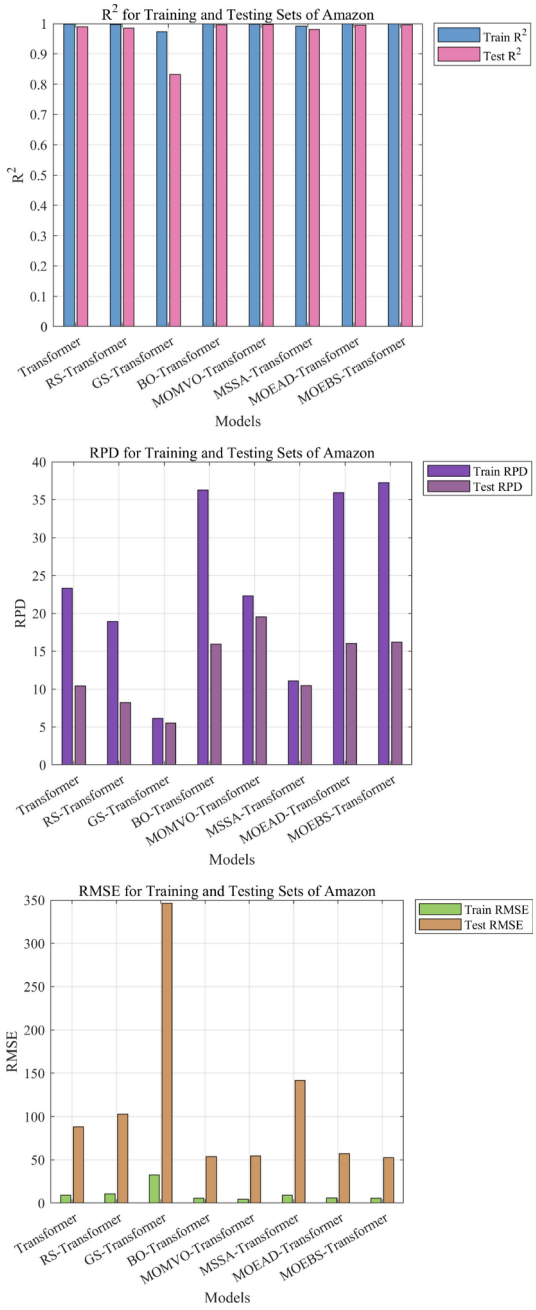

Fig. 25. Comparison of transformer models optimized by different algorithms on Amazon dataset.

lowest RPD, with test set RPDs of 5.5287 and 8.2235, respectively, indicating a lack of robustness in handling data fluctuations and maintaining prediction stability. In summary, the MOEBS-Transformer outperforms other optimization methods in all three key metrics (R^2, RMSE, RPD), validating its strong performance in hyperparameter optimization. Overall, multi-objective optimization algorithms perform better than Random Search, Grid Search, and Bayesian Optimization across all metrics, demonstrating their advantage in handling complex datasets and multi-dimensional optimization tasks.

The comparison results of transformer models optimized by different algorithms in Google

As shown in Fig. 26, the performance of the Transformer model optimized by different algorithms on the training and test sets of the Google dataset shows clear differences. On the training set, the MOEBS-Transformer achieves the lowest RMSE (15.7253), along with high R^2 and RPD, at 0.9909 and 10.5716, respectively, demonstrating MOEBS's effectiveness in improving the model's fitting capability and stability. MOMVO-Transformer also performs well on the training set, with an R^2 of 0.9907, and its RMSE and RPD are only slightly behind MOEBS-Transformer. Bayesian Optimization (BO-Transformer) also shows good results, achieving an R^2 of 0.9896, with

Model	Training set			Test set		
	RMSE	R^2	RPD	RMSE	R^2	RPD
Transformer	8.8969	0.9980	23.3262	88.2008	0.9891	10.4270
RS-Transformer	10.6295	0.9972	18.9148	102.9107	0.9852	8.2235
GS-Transformer	32.7169	0.9735	6.1445	346.4594	0.8324	5.5287
BO-Transformer	5.5452	0.9992	36.2845	53.7565	0.9960	15.9220
MOMVO-Transformer	**4.4441**	0.9979	22.3255	54.5659	**0.9972**	**19.5469**
MSSA-Transformer	8.9905	0.9915	11.0998	141.7431	0.9811	10.4672
MOEAD-Transformer	5.8889	0.9991	35.9243	57.4243	0.9954	16.0326
MOEBS-Transformer	5.4766	**0.9993**	**37.2649**	**52.5328**	0.9962	16.1724

Table 10. Performance comparison of Transformer models optimized by different algorithms on Amazon dataset.

relatively low RMSE and RPD. In contrast, RS-Transformer and GS-Transformer have higher RMSE and RPD, indicating weaker optimization effects, particularly in terms of effectively fitting the data on the training set.

From Table 11, it can be seen that on the test set, MOEBS-Transformer continues to perform best, with an RMSE of 33.9834, an R^2 of 0.9911, and an RPD of 10.5880, demonstrating excellent prediction accuracy and robustness on unseen data. MOMVO-Transformer and BO-Transformer also perform relatively well, with test set RMSEs of 36.4335 and 34.6023, respectively, and R^2 values close to 0.99. In contrast, RS-Transformer and GS-Transformer perform poorly on the test set, particularly GS-Transformer, which has an RMSE of 136.7710 and an R^2 of only 0.8144, indicating the limitations of traditional optimization methods in handling high-dimensional and complex data. Overall, the MOEBS-Transformer shows the best performance on both the training and test sets, while multi-objective optimization algorithms generally outperform random search and grid search in enhancing model fitting capability, prediction accuracy, and generalization.

The comparison results of transformer models optimized by different algorithms in Uniqlo

As presented in Fig. 27, the performance of the Transformer model optimized by different algorithms on the Uniqlo dataset varies across the training and test sets. As a globally recognized apparel brand, Uniqlo's stock price data is influenced by market consumption trends, seasonal fluctuations, and other factors, exhibiting high volatility. On the training set, the MOEBS-Transformer performs the best, with the lowest RMSE (838.9390), an R^2 of 0.9948, and an RPD of 13.9732, demonstrating its outstanding fitting capability in complex market data. MSSA-Transformer and MOEAD-Transformer also show good results in terms of R^2 and RPD, but their RMSEs are slightly higher than that of MOEBS-Transformer. In contrast, the GS-Transformer has a significantly higher RMSE (2087.9704) on the training set, indicating the limitations of traditional methods in optimizing high-dimensional financial time series data.

According to the results in Table 12, MOEBS-Transformer continues to excel on the test set, with an RMSE reduced to 996.7573, an R^2 of 0.9521, and an RPD of 4.6117, highlighting its strong generalization capability in handling highly uncertain test data. MOMVO-Transformer and MSSA-Transformer also achieve relatively high R^2 values on the test set (0.9482 and 0.9519, respectively), but their RMSE and RPD metrics are slightly inferior to those of MOEBS-Transformer. On the other hand, RS-Transformer and GS-Transformer perform poorly on the test set, especially GS-Transformer, with an RMSE of 2180.3976 and an R^2 of only 0.7707, reflecting its limitations in capturing complex fluctuation patterns. Overall, MOEBS-Transformer exhibits stronger adaptability and robustness in handling Uniqlo's highly volatile data, while multi-objective optimization algorithms, in general, are more effective than traditional methods in dealing with such financial time series data.

Conclusion and future work

This paper introduces the multi-objective extension of the Escaping Bird Search algorithm, namely MOEBS (Multi-Objective Escaping Bird Search Optimization). By employing advanced optimization strategies, MOEBS achieves a precise balance in the distribution of Pareto front solutions (PFS), significantly enhancing coverage. A thorough evaluation based on standard benchmark datasets such as ZDT, DTLZ, WFG, UF, and CF demonstrates MOEBS's robustness and effectiveness in addressing various multi-objective optimization challenges. Notably, the resulting Pareto front solutions exhibit a broad distribution. Analysis of evaluation metrics, including Hypervolume (HV), Generational Distance (GD), Inverted Generational Distance (IGD), and Spacing, confirms MOEBS's advantages and its ability to balance exploration and exploitation.In this study, we first compared five deep learning models—CNN, LSTM, BiLSTM, GRU, and Transformer—to establish the advantages of the Transformer model across three datasets: Google, Amazon, and Uniqlo. To address the limitations in hyperparameter settings, we proposed the MOEBS-Transformer model. To validate the performance of the MOEBS-Transformer, we compared it with three Transformer models that utilized multi-objective optimization algorithms for hyperparameter tuning (MOMVO-Transformer, MOEAD-Transformer, and MSSA-Transformer). Additionally, we conducted comparisons with Transformer models optimized using traditional methods such as Random Search (RS), Grid Search (GS), and Bayesian Optimization (BO) (RS-Transformer, GS-Transformer, and BO-Transformer).Furthermore, we conducted comparative experiments

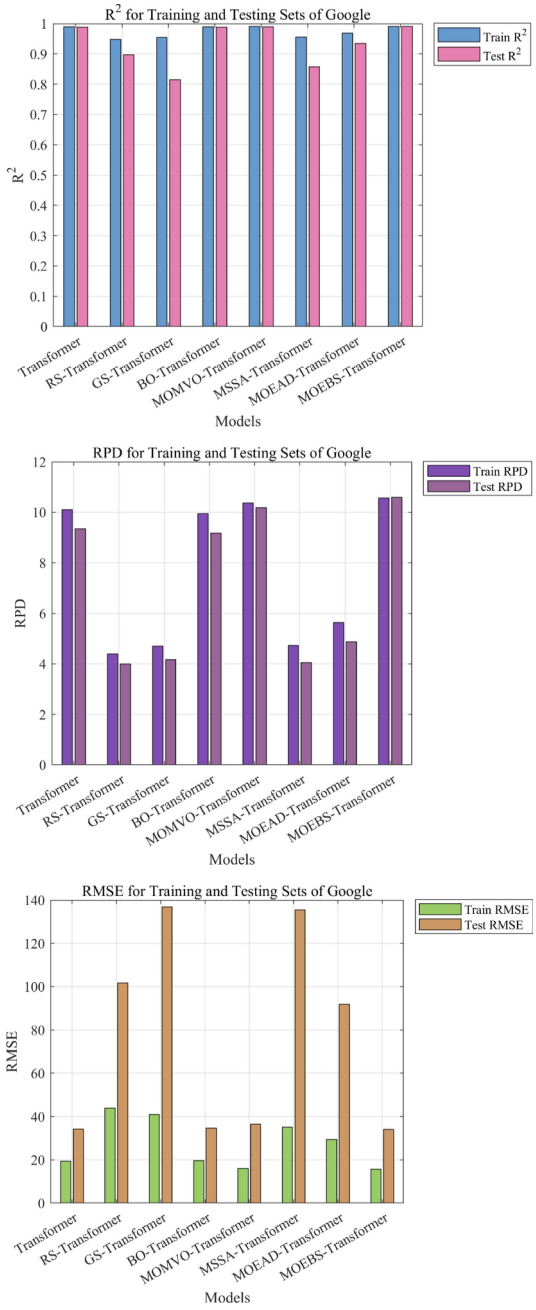

Fig. 26. Comparison of transformer models optimized by different algorithms on Google dataset.

on three metrics—R^2, RMSE, and RPD—across the three datasets, which thoroughly verify the optimization effectiveness and broad applicability of MOEBS-Transformer. However, MOEBS does have limitations; its performance in terms of Spacing indicates a lack of uniformity in solution distribution, suggesting potential areas for improvement. Moreover, the MOEBS-Transformer still has room for enhancement, as it does not outperform all state-of-the-art algorithms across every metric, such as RMSE, highlighting the need to improve its capacity to learn from training data and generalize to unseen test data.

Future enhancements can be achieved by optimizing computational efficiency and extending the Transformer model to a more complicated structure. For instance, distributed computing or parallel algorithms can be introduced to enhance MOEBS's computational efficiency, enabling it to handle larger-scale and higher-dimensional optimization problems. Additionally, incorporating time series analysis into the prediction model can enhance performance due to the inherent temporal dependencies and patterns in stock prices. The current research methodology can be extended to other financial assets, such as bonds, futures, and forex. The same models can be applied to optimize hyperparameters for these assets and use deep learning techniques for market prediction. While different financial assets exhibit different volatilities and behavior patterns, multi-objective optimization algorithms like MOEBS have broad applicability and can be adjusted and optimized for different

Model	Training set			Test set		
	RMSE	R^2	RPD	RMSE	R^2	RPD
Transformer	19.3469	0.9899	10.1062	34.1159	0.9885	9.3472
RS-Transformer	43.8561	0.9481	4.3997	101.7357	0.8973	4.0019
GS-Transformer	40.9845	0.9547	4.7014	136.7710	0.8144	4.1669
BO-Transformer	19.6255	0.9896	9.9478	34.6023	0.9881	9.1806
MOMVO-Transformer	15.9330	0.9907	10.3718	36.4335	0.9897	10.1880
MSSA-Transformer	35.0417	0.9549	4.7216	135.5162	0.8579	4.0564
MOEAD-Transformer	29.3011	0.9685	5.6410	91.8614	0.9347	4.8722
MOEBS-Transformer	**15.7253**	**0.9909**	**10.5716**	**33.9834**	**0.9911**	**10.5880**

Table 11. Performance comparison of Transformer models optimized by different algorithms on Google dataset.

markets and asset types. Additionally, Time series techniques allow the model to account for trends, seasonal effects, and cyclical behaviors, which are often present in financial data, thereby providing a more accurate and robust forecast. This approach leverages historical data to identify and exploit regularities over time, improving the model's ability to predict future price movements. These improvements will make MOEBS-Transformer a more powerful tool in the field of stock price prediction. We can also explore cross-domain applications in time series analysis, such as weather forecasting, climate forecasting, economic forecasting, retail forecasting, etc., to further improve MOEBS-Transformer's versatility30.

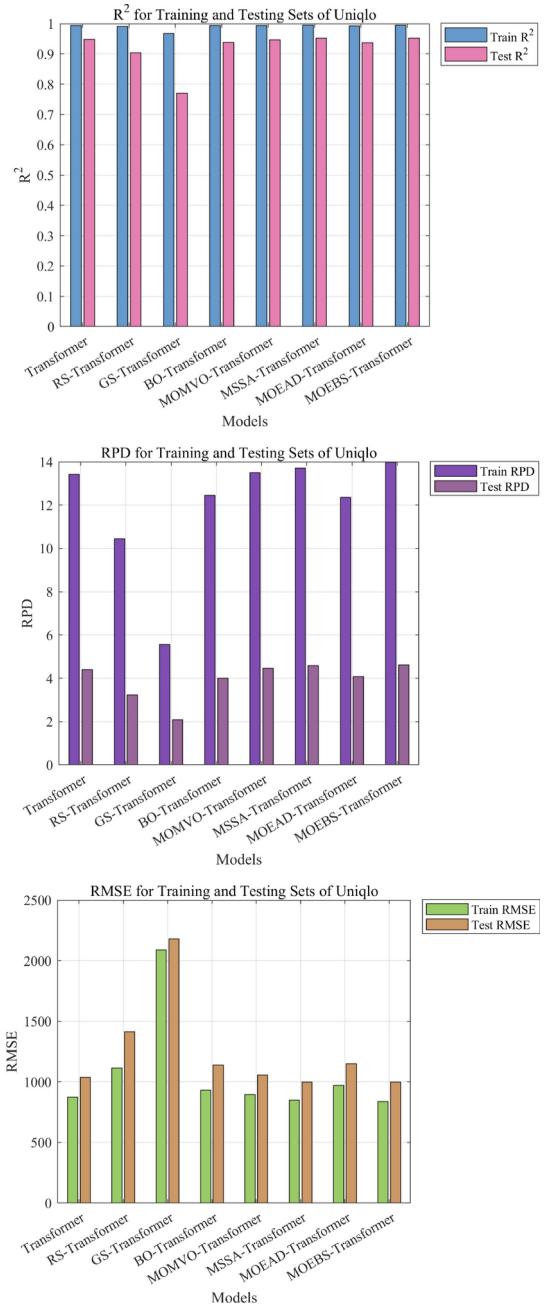

Fig. 27. Comparison of transformer models optimized by different algorithms on Uniqlo dataset.

Model	Training set			Test set		
	RMSE	R^2	RPD	RMSE	R^2	RPD
Transformer	873.0160	0.9944	13.4201	1036.6555	0.9482	4.4047
RS-Transformer	1113.0439	0.9908	10.4418	1413.2333	0.9037	3.2234
GS-Transformer	2087.9704	0.9677	5.5660	2180.3976	0.7707	2.0887
BO-Transformer	933.2676	0.9936	12.4520	1137.5240	0.9376	4.0062
MOMVO-Transformer	894.8251	0.9941	13.4962	1055.8910	0.9462	4.4673
MSSA-Transformer	850.4012	0.9946	13.7194	998.3982	0.9519	4.5856
MOEAD-Transformer	970.7575	0.9930	12.3596	1150.1705	0.9362	4.0845
MOEBS-Transformer	**838.9390**	**0.9948**	**13.9732**	**996.7573**	**0.9521**	**4.6117**

Table 12. Performance comparison of Transformer models optimized by different algorithms on Uniqlo dataset.

Data availability

All data sources and experimental results are provided within the manuscript or supplementary information files. For any further inquiries, please contact the corresponding author.

Appendix

See Tables 2,3,4,5,6

Received: 27 August 2024; Accepted: 31 January 2025
Published online: 17 February 2025

References

1. Yang, M., Zhang, J. & Bai, Z. A new approach to system design optimization of underwater gliders. *IEEE/ASME Trans. Mechatron.* **27**(5), 3494-505 (2022).
2. Sharma, S., Shukla, S. & Singh, V. mLBOA: A modified butterfly optimization algorithm with lagrange interpolation for global optimization. *J. Bionic. Eng.* **19**(4), 1161-176 (2022).
3. Jiang, F., Wang, L. & Bai, L. An improved whale algorithm and its application in truss optimization. *J. Bionic. Eng.* **18**(3), 721-32 (2021).
4. Barboza, F., Kimura, H. & Altman, E. Machine learning models and bankruptcy prediction. *Expert. Syst. Appl.* **83**, 405-17 (2017).
5. Son, H., Lee, S. & Kim, C. Data analytic approach for bankruptcy prediction. *Expert. Syst.Appl.* **138**, 112816 (2019).
6. Storn, R. & Price, K. Differential evolution A simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11**(4), 341-59 (1997).
7. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **12**(6), 702-13 (2008).
8. Beheshti, Z. & Shamsuddin, S. M. H. A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl.* **5**(1), 1-5 (2013).
9. Heidari, A. A., Abbaspour, R. A. & Chen, H. Efficient boosted grey wolf optimizers for global search and kernel extreme learning machine training. *Appl. Soft Comput.* **81**, 105521 (2019).
10. Mirjalili, S., Lewis, A. & Yang, X. Multi-objective multi-verse optimization: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **27**(2), 495-13 (2016).
11. Suthaharan, S., 2016 Support Vector Machine. In *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning,* 207-35 (2016).
12. Liu, Y., Li, Z. & Zheng, X. Chaotic simulated annealing multi-verse optimization enhanced kernel extreme learning machine for medical diagnosis. *Comput. Biol. Med.* **144**, 105356 (2022).
13. Emam, M. M., Houssein, E. H. & Ghoniem, R. M. A modified reptile search algorithm for global optimization and image segmentation: Case study brain MRI images. *Comput. Biol. Med.* **152**, 106404 (2023).
14. Das, S., Sahu, T. P. & Janghel, R. R. Stock market forecasting using intrinsic time-scale decomposition in fusion with cluster based modified CSA optimized ELM. *J. King Saud Univ.-Comp. Inform. Sci.* **34**(10), 8777-793 (2022).
15. Das, S. et al. Effective forecasting of stock market price by using extreme learning machine optimized by PSO-based group oriented crow search algorithm. *Neural Comput. Appl.* **34**(1), 555-1 (2022).
16. Mirjalili, S., Mirjalili, S. M. & Lewis, A. Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **27**(2), 495-13 (2016).
17. Chen, J., Zhu, F. & Zheng, X. Multi-threshold image segmentation based on an improved differential evolution: Case study of thyroid papillary carcinoma. *Biomed. Signal Process. Control* **85**, 104893 (2023).
18. Das, S., Sahu, T. P. & Janghel, R. R. Oil and gold price prediction using optimized fuzzy inference system based extreme learning machine. *Resour. Policy* **79**, 103109 (2022).
19. Vaswani, A., N. Shazeer, & N. Parmar. *Attention is All You Need*. (2017).
20. Zhou, H. M., Zhang, Y. & Liu, S. A modified particle swarm optimization algorithm for a batch-processing machine scheduling problem with arbitrary release times and non-identical job sizes. *Comput. Ind. Eng.* **123**, 67-1 (2018).
21. Devlin, J., et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. **1** 4171-186 (2019).
22. Chen, H., Ghoniem, R. M. & Abbaspour, R. A. An efficient double adaptive random spare reinforced whale optimization algorithm.https://doi.org/10.1016/j.eswa.2019.113018 (2023).
23. Deb, K. *Multi-Objective Optimization using Evolutionary Algorithms* (Wiley, 2002).
24. Chen, H., Wang, J. & Shi, Y. Simulated annealing-based dynamic step shuffled frog leaping algorithm: Optimal performance design and feature selection. *Neurocomputing* **503**, 325-62 (2024).

25. Das, S., Sahu, T. P. & Janghel, R. R. PSO-based group-oriented crow search algorithm (PGCSA). *Eng. Comput.* **38**(2), 545-71 (2021).
26. Ouyang, K. et al. Escape: An optimization method based on crowd evacuation behaviors. *Artif. Intell. Rev.***58**(1), 19 (2024).
27. Fu, S. et al. Red-billed blue magpie optimizer: a novel metaheuristic algorithm for 2D/3D UAV path planning and engineering design problems.*Artif. Intell. Rev.***57**(6), 1-9 (2024).
28. Ouyang, K. and S. Fu, Graph Neural Networks Are Evolutionary Algorithms. arXiv:2412.17629 (2024).
29. Glover, F. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13**, 533-49 (1986).
30. Faramarzi, A., Heidarinejad, M. & Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. https://doi.org/10.1016/j.knosys.2019.105190 (2020).
31. Fu, Y. et al. Secretary bird optimization algorithm: A new metaheuristic for solving global optimization problems. *Artif. Intell. Rev.* **57**(5), 123 (2024).
32. Pareto, V. *Manual of Political Economy* (Macmillan, 1906).
33. Kennedy, J. and R.C. Eberhart. Particle swarm optimization. (1995).
34. Deb, K. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182-97 (2002).
35. Zhou, Z. et al. A comprehensive review on evolutionary multi-objective optimization (EMO): Basic concepts, recent advances, and future directions. *Complex. Intell. Syst.* **8**(4), 3299-332 (2011).
36. Ghorbani, N. & Babaei, E. Exchange market algorithm. *Appl. Soft Comput.* **19**, 177-87 (2014).
37. Satapathy, S. & Naik, A. Social group optimization (SGO): a new population evolutionary optimization technique. *Complex. Intell. Syst.* **2**(3), 173-03 (2016).
38. Bergstra, J. & Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **13**(2), 281-05 (2012).
39. Hutter, F., H.H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy*, January 17-1. Selected Papers 5. (Springer, 2011).
40. Jones, D. R., Schonlau, M. & Welch, W. J. Efficient global optimization of expensive black-box functions. *J. Global Optim.* **13**, 455-92 (1998).
41. LeCun, Y. et al. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278-324 (1998).
42. Hochreiter, S. Long Short-term Memory. https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
43. Graves, A. and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM networks. In *Proc. 2005 IEEE International Joint Conference on Neural Networks* IEEE (2005).
44. Cho, K., Learning phrase representations using RNN encoder-decoder for statistical machine translation.arXiv:1406.1078(2014).
45. Vaswani, A., *Attention is all you need.* Advances in Neural Information Processing Systems (2017).
46. Macrotrends. *Alphabet - 20 Year Stock Price History | GOOGL*. 2024 August 26 2024 Available from: https://www.macrotrends.net/stocks/charts/GOOGL/alphabet/stock-price-history (2024)
47. Mirjalili, S., Lewis, A. & Yang, X. S. Grey wolf optimizer: Theory, literature review, and application in computational fluid dynamics problems. *Adv. Eng. Softw.* **69**, 46-1 (2016).
48. Chen, H., Wang, Z. & Shi, Y. Multi-threshold image segmentation based on an improved differential evolution: Case study of thyroid papillary carcinoma. *Biomed. Signal Process. Control* **85**, 104893 (2024).
49. Xiao, H., Zhang, H. & Zhao, L. Transformers in medical image segmentation: A review. *Biomed. Signal Process. Control* **84**, 104791 (2023).
50. Heidari, A. A. et al. Harris hawks optimization: Algorithm and applications. *Futur. Gener. Comput. Syst.* **97**, 849-72 (2019).
51. Liu, J. et al. Chaotic simulated annealing multi-verse optimization enhanced kernel extreme learning machine for medical diagnosis. *Comput. Biol. Med.* **144**, 105356 (2023).
52. Zhou, Y., Lan, J. & Li, W. Prediction of stock market prices based on combination of random forest and RBF neural network. *J. Comput. Des. Eng.* **8**(1), 95-06 (2021).
53. Mirjalili, S. et al. A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **114**, 163-91 (2017).
54. Azizi, M., Talatahari, S. & Gandomi, A. H. Fire hawk optimizer: A novel metaheuristic algorithm. *Artif. Intell. Rev.* **56**(1), 287-63 (2023).
55. Pourpanah, F. et al. A review of artificial fish swarm algorithms: Recent advances and applications. *Artif. Intell. Rev.* **56**(3), 1867-903 (2023).
56. Houssein, E. H., Chen, H. & Al-Atabany, W. Accurate multilevel thresholding image segmentation via oppositional Snake optimization algorithm: Real cases with liver disease. *Comput. Biol. Med.* **169**, 107922 (2023).
57. Meng, Y. et al. Dynamics of collective cooperation under personalised strategy updates. *Nat. Commun.* **15**(1), 3125 (2024).
58. Rahman, C. M. Group learning algorithm: A new metaheuristic algorithm. *Neural Comput. Appl.* **35**(19), 14013-028 (2023).
59. Hanif, M. & Mohammad, N. Metaheuristic algorithms for elevator group control system: A holistic review. *Soft Comput.* **27**(21), 15905-5936 (2023).
60. Huband, S. et al. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evolut. Comput.* **10**(5), 477-06 (2006).
61. Zhang, Q., et al., Multiobjective optimization test instances for the CEC 2009 special session and competition. (2008).
62. Knowles, J.D., L. Thiele, and E. Zitzler, A tutorial on the performance assessment of stochastic multiobjective optimizers. Tik report **214** (2006).
63. Han, H., Lu, W. & Qiao, J. An adaptive multiobjective particle swarm optimization based on multiple adaptive methods. *IEEE Trans. Cybern.* **47**(9), 2754-767 (2017).
64. Van Veldhuizen, D. A. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations* (Air Force Institute of Technology, 1999).
65. Zitzler, E. & Thiele, L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evolut. Comput.* **3**(4), 257-71 (1999).
66. Cho, K., et al., Learning phrase representations using RNN encoder-decoder for statistical machine translation arXiv:1406.1078(2014).

Author contributions

D.W. and K.O. designed the methodology and drafted the programming code. Z.W. curated the experimental data. D.W. conducted the experiments and led in writing the main manuscript. K.O. led in analyzing the experimental results. M.Q., J.Y., and J.Y. assisted in analyzing the experimental results. Z.W. and X.S. assisted in writing the main manuscript and led in providing visualization. M.Q. and Y.J. assisted in providing visualization. Z.W. and X.S. enhanced the programming code. X.S. and K.O. led in reviewing the manuscript and all the other authors assisted in reviewing the manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-88883-8.

Correspondence and requests for materials should be addressed to X.S. or K.O.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.