Journal of Computers Vol. 34 No. 5, October 2023, pp. 117-133
doi: 10.53106/199115992023103405009

Stud Hybrid Tasmanian Devil - Grey Wolf Optimization: A Novel Bio-Inspired Optimization Algorithm for Learning-to-Rank

Guo-Xi Zhang, Yong-Quan Dong*, Jia-Chen Tan, Hao-Lin Yang

School of Computer Science and Technology, Jiangsu Normal University, Jiangsu, China

{zgx116, tomdyq, 2020200455, yhl}@jsnu.edu.cn

Received 7 October 2022; Revised 4 January 2023; Accepted 14 March 2023

Abstract. Building ranking models used in Information Retrieval (IR) is one of the main problems of learning-to-Rank. Tasmanian Devil Optimization algorithm is a recently proposed meta-heuristic optimization algorithm based on the Tasmanian Devils' predation mechanism. In this paper, grey wolf operator and stud mechanism are introduced into the TDO optimization process to improve the ability of exploitation and achieve better performance. The proposed Stud Hybrid Tasmanian Devil - Grey Wolf Optimization (SHTDO) combines Grey Wolf Optimizer and TDO with a Stud Selection and Crossover Operator, thus helping to enhance the exploration and exploitation as well as the motion under both strategies. Also, to validate the algorithm on the different fields, widely accepted benchmark and dataset was used for testing. The results show that our algorithm obtains higher ranking performance than TDO, which makes it more suitable for solving optimization problems, especially LTR tasks.

Keywords: optimization algorithm, grey wolf operator, tasmanian devil optimization, stud selection and crossover operator, learning-to-rank

1 Introduction

Information retrieval (IR) is the field concerned with the structure, organization, storage, analysis, search and retrieval of information. Information retrieval collects, organizes, and stores all types of information so that users can quickly find and access the information they are interested in. [1] With the development of the Internet, fast and efficient retrieval systems are particularly important. The development of information retrieval applications has led to a high degree of attention being paid to ranking techniques, while the development of ranking techniques has also contributed to the widespread use of information retrieval technology.

The huge amount of data makes it possible to build various heuristic ranking models. Learning to rank (LTR) [2], which uses machine learning algorithms to train a ranking model on a large amount of existing data, is one of the current hot topics of research in the field of information retrieval and machine learning. This model prepares training data and test data consisting of queries and documents, then performs feature extraction and uses the extracted features to train the LTR model. This algorithm can learn the ranking function automatically from the data, saving the effort of manual adjustment.

At this stage, there are various technical routes to solving the LTR problem, one of which is to use an optimization algorithm for direct optimization of the ranking metrics, i.e. to transform the ranking problem into an optimization problem. For optimization problems, the algorithm needs to find the optimal solution to a given problem under complex constraints in a reasonable duration. Evolutionary computational methods are often used to solve these types of optimization problems. In the last few decades, algorithms using meta-heuristic optimization have made great progress, especially in solving many complex optimization problems. Meta-heuristic optimization algorithms deal with real-world optimization problems by simulating biological or physical phenomena. Due to the stochastic nature of meta-heuristic algorithms, it has a strong ability to prevent itself from getting trapped in a local optimum. These algorithms are simple, efficient, robust, and flexible and can be used in a variety of fields. So far, some of the most famous meta-heuristics are particle swarm optimization (PSO) [3], Biogeography-Based Optimization (BBO) [4], artificial bee colony algorithm (ABC) [5], Firefly algorithm (FA) [6], Krill herd (KH) [7], Grey Wolf Optimizer (GWO) [8], Ant-Lion optimizer (ALO) [9], butterfly optimization algorithm (BOA) [10], Beetle Antennae Search Algorithm (BAS) [11], and the recent Tasmanian Devil Optimization (TDO) [12].

* Corresponding Author

The most challenging task encountered in developing meta-heuristics is finding suitable balance between exploration and exploitation. The Tasmanian devil is a carnivorous wild animal that lives on the island of Tasmania. Despite their ability to hunt for prey, they prefer to feed on carrion if it is available. They have two foraging strategies. The Tasmanian devils feed on carrion if they find it. And the second strategy is to hunt and feed by attacking their prey. Tasmanian Devil Optimization algorithm [12] simulates the Tasmanian devils' feeding behavior, one relying on carrion and the other relying on hunting. In foraging, if carrion is present, the Tasmanian devil will feed on it, otherwise the Tasmanian devil will seek out prey and take up hunting to obtain food.

Although TDO is a more advanced optimization algorithm, there are limitations in applying it to the LTR problem. It appears to be weak in coping with optimization problems in high-dimensional search spaces, and it is easy to fall into local optimal solutions. Therefore, the main innovation of this paper is to introduce the gray wolf operator into TDO to improve the accuracy and speed of fitting the LTR problem with the help of its fast convergence property. Meanwhile, the Stud selection crossover operator is used to improve the overall stability of the population through the interaction of other individuals with the optimal ones.

The following parts of this paper is organized as follows: Related works of this paper are discussed in Section 2;An overview of Tasmanian Devil Optimization is presented in Section 3. In addition, the details of the proposed SHTDO algorithm and its application on Learning-to-rank task is provided in Section 4; Subsequently, A series of experiments are performed to demonstrate the performance of SHTDO algorithm in Section 5. Finally, Section 6 sums up present work and makes an outlook for the future work.

2 Related Works

Learning to Rank (LTR) refers to a series of machine learning based ranking algorithms that were originally applied in the field of Information Retrieval (IR), most typically to solve the problem of ranking search results by search engines. [13] It uses machine learning to train a ranking model on a large amount of existing data and then obtain correlations with query terms based on document characteristics. It can automatically learn the ranking function from the data, saving the effort of manual adjustment. The advantage of ranking learning over traditional models is that it can combine and optimize a large number of ranking features and automatically learn the corresponding parameters, resulting in an efficient and more accurate ranking model.

After feature extraction by the feature extractor, the training data is organized in query units to obtain a series of labelled training data sets. For example, for a web or document retrieval problem with N queries and a set of documents for each query, the features $x_i \in R^d$ in the training sample are the query document pairs (q_i, d_i) and the labels y_i are the relevance scores of the document doc_i to the query q_i, e.g. very relevant, relatively relevant, generally relevant or not relevant. The goal of the ranking learning algorithm is to learn a model that can predict the relevance y_j^i from the feature vector x_j^i.

The ranking model used in this paper, f, is a real-valued function, more precisely a document-level function, that describes a linear combination of features in a vector:

$$f(q_i, d_i) = W^T \Phi(q_i, d_i), \tag{1}$$

where W denotes the weight vector, $\Phi(q_i, d_i)$ is feature vector of d_i in query q_i.

LTR algorithms can be classified into three categories according to the way they model the ranking problem: pointwise, pairwase, and listwise methods. The different categories have different input and output spaces and employ different objective loss functions. Among them, the listwise approach considers that each document in the same query is related to each other, and trains the list of documents, labeled with the ranking of the query documents. The listwise approach has a more natural model design for the ranking problem and solves the problem that the ranking should be based on query and position.

One implementation of the listwise approach is to optimize the sorting metrics directly for, say, NDCG. The difficulty with direct optimization of ranking metrics is that metrics such as NDCG are mathematically "Non-Continuous" and "Non-Differentiable" in form. Most of the optimization algorithms are based on continuous and differentiable functions. Therefore, direct optimization is more difficult.

An effective solution to this challenge is to use optimization techniques that can optimize non-smooth objectives, such as Adaboost-based AdaRank [14] and the genetic programming ranking algorithm RankGP [15]. Approaches that directly optimize ranking metrics based on genetic and metaheuristic algorithms are highly rele-

vant work for this study. RankGP is an approach that uses genetic programming to generate ranking functions by analyzing documents for different ranking metrics. RankPSO optimizes NDCG metrics directly using the particle swarm algorithm. The work in this paper is an improvement of a recent optimization algorithm, resulting in a new algorithm. Moreover, it is not only applicable to the LTR problem, but also has a strong solution for other optimization problems, which is the difference of this work.

As one of state-of-the-art meta-heuristic algorithms, TDO has excellent exploration and exploitation capabilities, but has some shortcomings when dealing with LTR tasks. In the first step of the exploration phase, "searching for prey", the location of prey is achieved by randomly selecting other individuals in the population, which is an inefficient and unstable practice and has the possibility of falling into local optimum. Therefore, how to optimize the operator of this process is one of the keys to improve the algorithm to cope with the search ranking problem.

On the other hand, In both exploitation and exploration phases, the process of updating the location is updated independently using random operators and there is no connection between individuals. Therefore, after the execution of the two strategies, the interaction between individuals is executed, and the use of the best individuals in the population to modify the less adapted individuals is beneficial to improve the fitness of the population. These are the key questions to be addressed in this study.

3 Overview of Tasmanian Devil Optimization

3.1 Initialization

TDO randomly generates the initial population based on the constraints of the problem. [12] The members of the TDO population are searchers of the solution space and propose candidates for the problem based on their position in the search space. Thus, mathematically, each member of this kind of group is a vector, whose number of elements is equal to the dimensionality of the problem. Thus, the set of TDO members can be modeled by a matrix in Eq. 1.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times M} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,M} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,M} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,j} & \cdots & x_{N,M} \end{bmatrix}_{N\times M}, \tag{2}$$

where X is the population of search agents, X_i is the i-th candidate solution, $x_{i,j}$ is its potential value for the j-th variable, N is the number of individuals searched for, and M is dimension, i.e. the amount of variables for a given problem.

3.2 Exploration Strategy (Feeding on Carrion)

This stage simulates the Tasmanian devil's preference to feed on carrion rather than hunt, and its location has been updated as follows

$$\begin{aligned} x_{i,j}^{new,S1} &= \begin{cases} x_{i,j} + r\cdot\left(c_{i,j} - I\cdot x_{i,j}\right), & F_C < F_i \\ x_{i,j} + r\cdot\left(x_{i,j} - c_{i,j}\right), & \text{otherwise} \end{cases}, \\ X_i &= \begin{cases} X_i^{new,S1}, & F_i^{new,S1} \\ X_i, & \text{otherwise} \end{cases}, \end{aligned} \tag{3}$$

where $X_i^{new,S1}$ is new position of i-th individual performing exploration strategy; $X_{i,j}^{new,S1}$; F_{c_i}; r is a random number in range [0,1]; i is a random number in {1,2}.

3.3 Exploitation Strategy (Feeding on Hunting)

This stage is divided into finding prey and attacking prey. First, the individual scans the area and selects the prey. In the second stage, it will chase the prey after approaching it, capture it and start feeding. In this strategy, when the i-th individual moving, the positions of the other members are assumed to be those of the prey. The t-th member of the population is randomly selected as prey, where t is a random number in [1, N] and i is its opposite.

$$P_i = X_t,\ i = 1, 2, ..., N,\ t \in \{1, 2, ..., N |\ t \neq i\}, \tag{4}$$

where P_i is the chosen prey of the i-th Tasmanian devil.

The location of individuals in the Prey searching phase is updated as follows

$$\begin{aligned} x_{i,j}^{new,S2} &= \begin{cases} x_{i,j} + r \cdot \left(p_{i,j} - I \cdot x_{i,j} \right), & F_{P_i} < F_i; \\ x_{i,j} + r \cdot \left(x_{i,j} - p_{i,j} \right), & \text{otherwise}, \end{cases} \\ X_i &= \begin{cases} X_i^{new,S2}, & F_i^{new,S2} < F_i; \\ X_i, & \text{otherwise}, \end{cases} \end{aligned} \tag{5}$$

where $X_i^{new,S2}$ is the i-th Tasmanian devil's new status based on exploitation strategy, $X_{i,j}^{new,S2}$ is its value for j-th variable; $F_i^{new,S2}$ is its objective function value, while F_{P_i} is the objective function's value of the prey it has selected.

The location of individuals in the Prey chasing phase has been updated as follows.

$$\begin{aligned} R &= 0.01\left(1 - \frac{t}{T} \right), \\ x_{i,j}^{new} &= x_{i,j} + (2r - 1) \cdot R \cdot x_{i,j}, \\ X_i &= \begin{cases} X_i^{new}, & F_i^{new} < F_i \\ X_i, & \text{otherwise} \end{cases}. \end{aligned} \tag{6}$$

Here, R is neighborhood radius of the attacked location point, t is the iteration counter, T is the maximum iteration number; X_i^{new} is the new status of i-th individual in neighborhood X_i, $x_{i,j}^{new}$ is its value for j-th variable, and F_i^{new} is its objective function value. The overall flowchart of TDO is shown in Fig. 1.

Algorithm 1. Tasmanian devil optimization

Input: Number of the population (N), number of iterations (T), strategy probability (P), optimization problem information and fitness function.

Output: The best solution obtained by TDO for given optimization problem.

Initialization of the position of all search agents and evaluation of the objective function.
for t = 1: T **do**
  for i = 1: N **do**
    if Probability < P **then**
      Exploration strategy (Feeding on carrion)
        Calculate new status and update i-th individual using Eq. 2.
    else
      Exploitation strategy (Feeding on hunting)
      Stage 1: Prey searching
      Select prey for the i-th individual using Eq. 3
      Compute new status and update the i-th individual using Eq. 4
      Stage 2: Prey chasing
      Update neighborhood radius, compute new status and update the i-th individual using Eq. 5.
    end if
  end for
  Evaluate fitness of Tasmanian devils.
  Save the best solution proposed so far.
end for

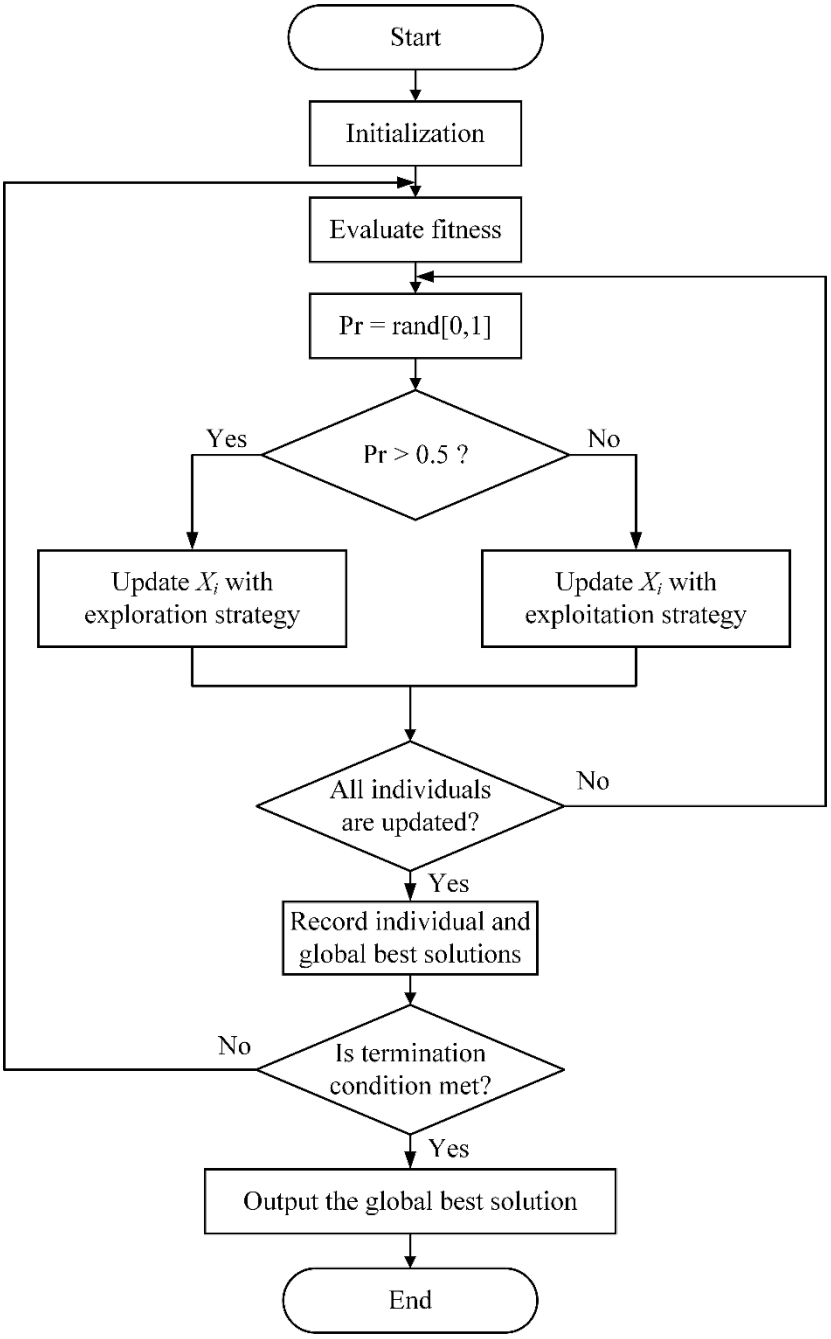


Fig. 1. Flowchart of TDO algorithm

4 Methodology

4.1 The Proposed Stud Hybrid Tasmanian Devil - Grey Wolf Optimization

Overview. The algorithm proposed in this study introduces the stud mechanism with the gray wolf operator into TDO, which will be called Stud Hybrid Tasmanian Devil - Grey Wolf Optimization (SHTDO) for the convenience of description in the following. The algorithm flow is shown in Fig. 2.

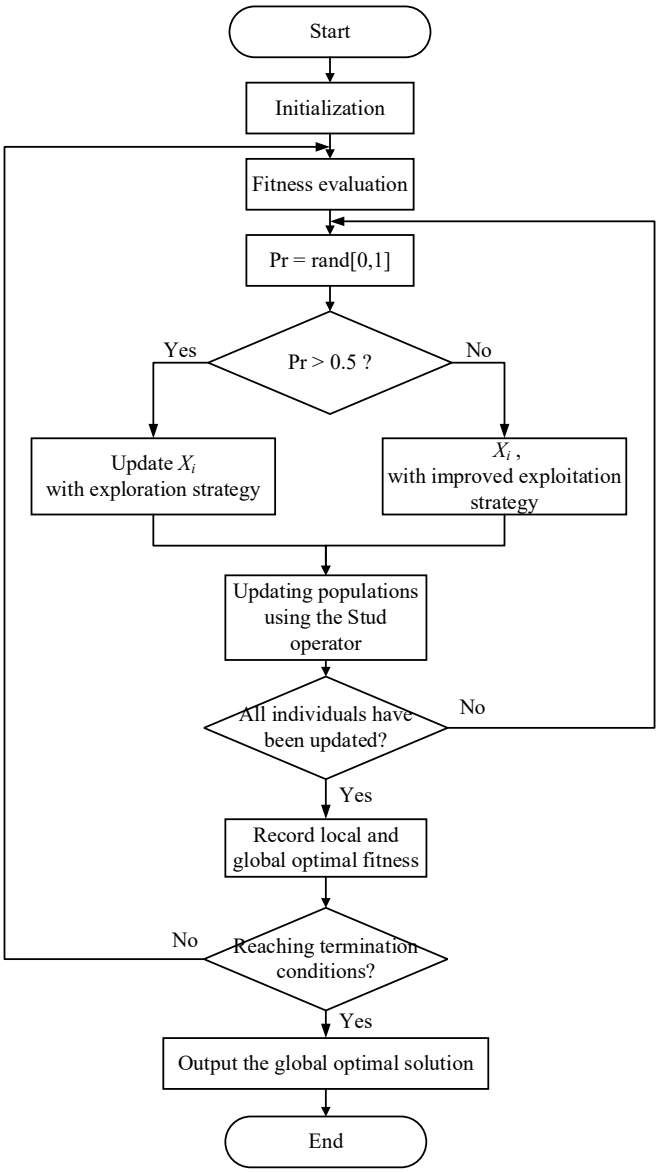


Fig. 2. Flowchart of stud hybrid tasmanian devil – grey wolf optimization algorithm

Grey Wolf Operator. Inspired by the predatory behavior of gray wolves, Mirjalili et al [8] proposed a new optimization algorithm, Grey Wolf Optimizer (GWO), in 2014. GWO achieves the optimization purpose by simulating the predatory behavior of gray wolves and based on the mechanism of wolf group collaboration. GWO algorithm has the features of simple structure, few parameters to be adjusted and easy to implement. GWO algorithm is characterized by simple structure, few parameters to be adjusted, easy implementation, etc. There are convergence factors and information feedback mechanisms that can be adjusted adaptively, and it can achieve a balance between local search and global search, so it has good performance in terms of problem solving accuracy and convergence speed. There are three main steps in individual gray wolf hunting; finding prey, surrounding prey and attacking prey.

This work uses the gray wolf operator to replace the mining strategy part of TDO. The entire population is divided into four classes α, β, γ, ω according to the best adapted individuals, the second best individuals, the third best individuals, and other individuals, and the former class has dominance over the later class. Because the gray wolf ω has the largest proportion of the pack and must be completely subordinate to the gray wolf α, β, γ, the hunting behavior of the pack is mainly guided and directed by the gray wolf α, β, γ.

Let the position of gray wolf α be X_α , the position of gray wolf β be X_β, the position of gray wolf γ be X_γ , and the position of any gray wolf within the whole population be X_i. Here, X_i can be the same as the three head wolves. Then the new position of gray wolf i, guided by the three head wolves, is

$$\begin{aligned} X_{\alpha i} &= X_\alpha - A_1 \cdot D_\alpha, \\ D_\alpha &= \left| C_1 \cdot X_\alpha - X_i \right|, \\ C_1 &= 2r_2, \\ A_1 &= 2a \cdot r_1 - a; \end{aligned} \tag{7}$$

$$\begin{aligned} X_{\beta i} &= X_\beta - A_2 \cdot D_\beta, \\ D_\beta &= \left| C_2 \cdot X_\beta - X_i \right|, \\ C_2 &= 2r_2, \\ A_2 &= 2a \cdot r_1 - a; \end{aligned} \tag{8}$$

$$\begin{aligned} X_{\gamma i} &= X_\gamma - A_3 \cdot D_\gamma, \\ D_\gamma &= \left| C_3 \cdot X_\gamma - X_i \right|, \\ C_3 &= 2r_2, \\ A_3 &= 2a \cdot r_1 - a; \end{aligned} \tag{9}$$

where a decreases linearly from 2 to 0 as the number of iterations increases. r_1 and r_2 are both random numbers between [0,1].

In summary, the formula for the next position of this gray wolf under the simultaneous guidance of the gray wolf α, β, γ is as follows:

$$X_i = \frac{X_{\alpha i} + X_{\beta i} + X_{\delta i}}{3}. \tag{10}$$

In particular, since the higher class has dominance over the lower class, it will not accept the guidance of the head wolf of the lower class when the gray wolf α, β, γ renews its position.

Stud Selection and Crossover Operator. Genetic algorithms have been widely used since their development, and they have proven successful in solving many benchmark and practical engineering problems. The variation and crossover operators in genetic algorithms can be sufficient to solve the problems, but random crossover and random variation are not required. In contrast, Stud Genetic Algorithm (SGA) [16] is a genetic type that uses the optimal genome for crossbreeding in each generation. The idea of SGA is to use the optimal genome to cross with all other genomes to produce new offspring [16]. Inspired by this, Stud selection and crossover operator (SSC) was used in this study to further optimize the overall fitness of the population. The process is shown in *Algorithm 2*.

Algorithm 2. Stud selection and crossover operator

Input: An individual in the populations X_i and its fitness value F_i.
Output: New individual after manipulation X'_i.
Select of the best individuals X_s (stud) from the population for breeding.
Perform single-point crossover to generate new individual X^*_i.
Evaluate the fitness value of new individual F^*_i.
if $F^*_i > F_i$ **then**
  Accept new individuals and replace parental individual.
else
  Abandon the new individual, still use the parent.
end if

4.2 Fitness Function and Predictor for Learning-to-Rank Mission

In this study, we use a linear regression function as a predictor to cope with the LTR task:

$$f(X) = \sum_i w_i \cdot x_i, \tag{11}$$

where w_i is the parameters to be optimized.

A generic function $E(*) \in [0,1]$ is used to represent an evaluation metric. E measures the distance between prediction and true label. Most evaluation metrics return the actual value in [0,1]. Ideally, a ranking model is created that maximizes the accuracy, or equivalently minimizes the loss function, based on the IR metric of the training data, defined as follows:

$$f(X) = \sum_{q=1}^{N} (E(X_q^{ideal}) - E(X_q)), \tag{12}$$

where X_q is the prediction for query q by generated ranking model f.

5 Experiments

5.1 Experimental Settings

All the experiments were performed at MATLAB R2022a in PC with Windows 11 operating system, which has a 3.6 GHz 6-core CPU and 32 GB RAM. To verify the performance of our algorithm in different scenarios, we tested our algorithm with the reference algorithm on the benchmark function, the industrial design optimization problem, and the LTR problem, and compared the results. Specifically, STDO removes the morphology of the gray wolf operator for our algorithm, i.e., the TDO algorithm that introduces only the breeding stock mechanism, which is used here as an ablation reference.

Benchmark Functions. The IEEE Congress on Evolutionary Computation (CEC) is the highest level conference in evolutionary computation worldwide. Most of the papers submitted to this conference test the proposed algorithms using 30 test functions provided by the conference organisers. The CEC'17 benchmark [17], announced at the 2017 conference, is also used in this study as a benchmark test function to evaluate the performance of the algorithms.

CEC'17 benchmark defines 20 basis functions, based on which the above basis functions are transformed through shifting, rotation and combination to generate 30 benchmark test functions, including 4 single-peaked functions, 6 multi-peaked functions, 10 hybrid functions and 10 composite functions. All test functions are minimization problems, their optimal values are 0, the search space are $[-100,100]^D$, where D is the number of dimensions of the search space. In general, the dimension is taken to be one of 10, 30, 50 or 100.

Industrial Design Optimization Problems. Many problems in industrial design can be abstracted into optimization problems. This study evaluates the performance of the algorithm in practical applications by optimizing three engineering design optimization problems: welded beam design, pressure vessel design, and telescopic spring design.

1) Optimization of welded beam design

The welded beam design is a minimization problem whose main objective is to reduce the manufacturing cost of the welded beam [18]. A schematic representation of this problem is shown in Fig. 3.

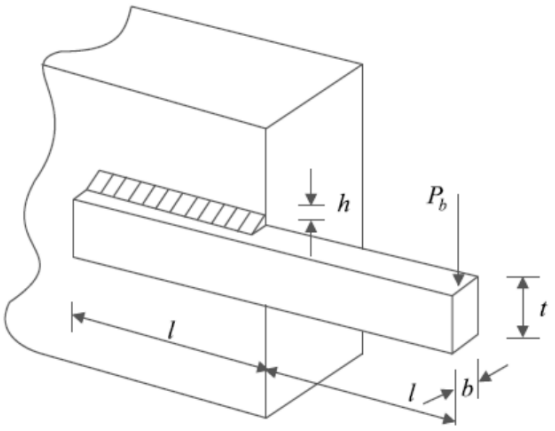


Fig. 3. Schematic diagram of the welded beam design optimization problem

The formal description of this problem is:

$$\begin{aligned} &X=\left[x_1,x_2,x_3,x_4\right]=[h,l,t,b],\\ &\text{Minimize } f(x)=1.10471x_1^2x_2+0.04811x_3x_4\left(14.0+x_2\right),\\ &s.t.\begin{cases} g_1(x)=\tau(x)-13600\le 0,\\ g_2(x)=\sigma(x)-30000\le 0,\\ g_3(x)=x_1-x_4\le 0,\\ g_4(x)=0.10471x_1^2+0.04811x_3x_4\left(14+x_2\right)-5.0\le 0. \end{cases} \end{aligned} \tag{13}$$

2) Optimization of pressure vessel design

Pressure vessel design is a minimization problem whose main objective is to reduce the total cost of material, welding and forming of cylindrical vessels [19]. A schematic representation of this problem is shown in Fig. 4.

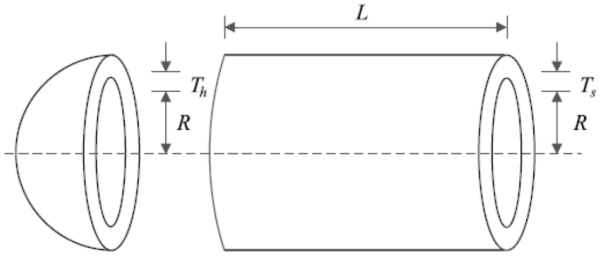


Fig. 4. Schematic diagram of pressure vessel design Optimization problem

The formal description of this problem is:

$$
\begin{aligned}
&X=\left[x_1,x_2,x_3,x_4\right]=\left[T_s,T_h,R,L\right],\\
&\text{Minimize}\, f(x)=0.6224x_1x_3x_4+1.778x_2x_3^2+3.1661x_1^2x_4+19.84x_1^2x_3,\\
&s.t.\begin{cases}
g_1(x)=-x_1+0.0193x_3\le 0,\\
g_2(x)=-x_2+0.00954x_3\le 0,\\
g_3(x)=-\pi x_3^2x_4-\dfrac{4}{3}\pi x_3^3+1296000\le 0,\\
g_4(x)=x_4-240\le 0,\\
0\le x_1,x_2\le 100,\quad 10\le x_3,x_4\le 200.
\end{cases}
\end{aligned}
\tag{14}
$$

3) Optimization of the telescopic spring design

The telescopic spring design is also a minimization problem whose main objective is to reduce the weight of the telescopic spring [18]. A schematic representation of this problem is shown in Fig. 5.

Fig. 5. Schematic diagram of telescopic spring design Optimization problem

The formal description of this problem is as follows.

$$
\begin{aligned}
&X=\left[x_1,x_2,x_3\right]=[d,D,P],\\
&\text{Minimize}\, f(x)=\left(x_3+2\right)x_2x_1^2,\\
&s.t.\begin{cases}
g_1(x)=1-\dfrac{x_2^3x_3}{71785x_1^4}\le 0,\\
g_2(x)=\dfrac{4x_2^2-x_1x_2}{12566\left(x_2x_1^3\right)}+\dfrac{1}{5108x_1^2}-1\le 0,\\
g_3(x)=1-\dfrac{140.45x_1}{x_2^2x_3}\le 0,\\
g_4(x)=\dfrac{x_1+x_2}{1.5}-1\le 0,\\
0.05\le x_1\le 2,\quad 0.25\le x_2\le 1.3,\quad 2\le x_3\le 15.
\end{cases}
\end{aligned}
\tag{15}
$$

LTR Datasets. The datasets used in the experiments were Microsoft MSLR-WEB10K and MSLR-WEB30K provided by Qin and Liu [20], containing a total of 10,000 and 30,000 query terms. This dataset consists of queries and a series of document feature vectors, each sample representing a query-document pair with a label for relevance evaluation, with labels ranging from 0 (not relevant) to 4 (fully relevant) representing the degree of relevance of documents and queries from low to high, derived from statistical parsing of log data from the Microsoft Bing search engine. Each query sample is a 136-dimensional feature vector containing query information, document features and interaction features, in the form of both discrete categorical features and continuous dense features such as BM25, which represent the basic features commonly used in the research field of IR. The dataset is divided into five subsets *S*1, *S*2, *S*3, *S*4, *S*5, which are divided into five folds according to different

training, validation and test sets, with a division ratio of 3:1:1. In the experiments conducted in this paper, the training, validation and test sets are divided into {(*S*1, *S*2, *S*3), (*S*4), (*S*5)} without loss of generality.

Evaluation Metrics of LTR Problem.

(1) Mean Average Precision (MAP)

In multi-categorization, MAP is used to evaluate the performance of a model for all categories, while ensuring that each category is given equal weight without bias relative to metrics such as Precision, Recall, and AUC. This metric can give more weight to errors that are at the top of the results list. Conversely, it gives less weight to errors that appear further down the list. This is consistent with the need to display as many relevant entries as possible at the top of the search results list.

Precision@k (P@k) is the proportion of relevant items in the top K search results.

$$P@k = \frac{\#\text{relevant items in top k results}}{k}. \tag{16}$$

Average precision@k (AP@k) is the sum of P@K for different k values divided by the total number of relevant items in the previous k results.

$$AP@k = \frac{1}{r_k}\sum_{1}^{k} P@i \cdot rel_i, \tag{17}$$

where r_k means number of relevant items in first k results, rel_i means the relevance score of i-th item.

Based on the above arithmetic, the Mean Average Precision@k (MAP@k) measures the mean value of average Precision@K over all queries for the entire dataset.

(2) Normalized Discounted Cumulative Gain (NDCG)

This metric is usually used to measure and evaluate search result algorithms. It is based on two ideas: results with high relevance influence the final metric score more than results with average relevance, and the higher the metric will be when there are results with high relevance appearing further up the list.

NDCG calculates cumulative gain (CG) first

$$CG_p = \sum_{i=1}^{p} rel_i, \tag{18}$$

where rel_i means the relevance score of i-th document. This is followed by the calculation of the discounted value of the cumulative gain (Discounted CG, DCG), which is designed so that the higher ranked results have a greater impact on the final score, i.e. the further down the ranking the document is, the lower the value.

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(1+i)}. \tag{19}$$

The number of search results returned varies for different search terms, and the DCG is a cumulative value that is not comparable for the results of two different search sorts. Therefore further normalization is required, i.e.

$$NDCG@k = \frac{DCG@k}{IDCG@k}, \tag{20}$$

where IDCG is the maximum DCG value with ideal conditions.

$$IDCG@k \quad \sum_{i\ 1}^{\|REL} \frac{2^{rel_i} \quad 1}{\log_2(i \quad 1)}, \tag{21}$$

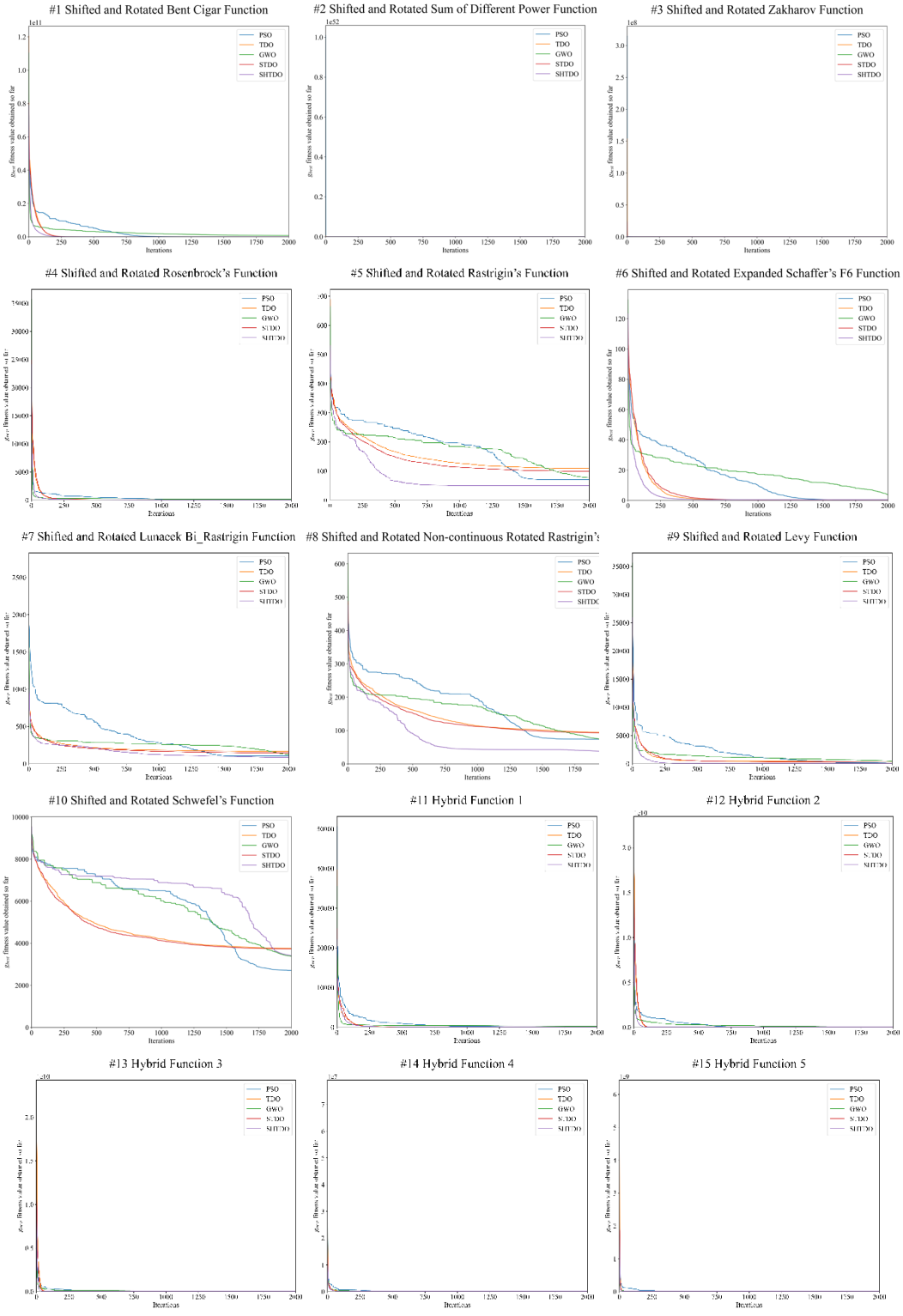
where ||*REL*|| denotes the set of the top *k* results in descending order of relevance, which is equivalent to ranking the results in the best way.

5.2 Experimental Results and Analysis

Performance Comparison of Benchmark Functions. The test results of SHTDO and the comparison models on CEC'17 are shown in the Table 1. It can be seen that SHTDO leads the other comparison models on 22 different benchmark functions within a limited number of iterations. For the benchmark functions that are not leading, the difference between SHTDO and the best performing algorithm is mostly within 1 order of magnitude, which shows that our algorithm has a strong ability to handle multiple complex optimization problems.

Table 1. Performance comparison of different algorithms on the CEC'17 benchmark function

F	PSO	GWO	TDO	STDO	**SHTDO**
F_1	4.87×10^{3}	8.07×10^{8}	6.92×10^{3}	3.44×10^{3}	**1.53×10^{3}**
F_2	1.45×10^{17}	9.94×10^{26}	3.08×10^{9}	**1.33×10^{4}**	3.34×10^{4}
F_3	8.81×10^{3}	2.70×10^{4}	1.90×10^{4}	1.99×10^{4}	**1.18×10^{3}**
F_4	1.24×10^{2}	1.46×10^{2}	9.48×10^{1}	9.93×10^{1}	**7.12×10^{1}**
F_5	7.08×10^{1}	7.68×10^{1}	1.08×10^{2}	1.08×10^{2}	**3.94×10^{1}**
F_6	**7.99×10^{-3}**	3.59×10^{0}	3.40×10^{-1}	7.34×10^{-1}	1.41×10^{-1}
F_7	1.03×10^{2}	1.24×10^{2}	1.64×10^{2}	1.50×10^{2}	**7.21×10^{1}**
F_8	7.38×10^{1}	7.20×10^{1}	9.48×10^{1}	1.07×10^{2}	**4.14×10^{1}**
F_9	1.11×10^{2}	4.79×10^{2}	3.68×10^{2}	1.31×10^{2}	**1.21×10^{1}**
F_{10}	2.71×10^{3}	3.38×10^{3}	3.76×10^{3}	3.84×10^{3}	**2.62×10^{3}**
F_{11}	1.01×10^{2}	2.91×10^{2}	8.27×10^{1}	6.62×10^{1}	**6.21×10^{1}**
F_{12}	9.77×10^{5}	3.10×10^{7}	5.32×10^{4}	4.72×10^{4}	**3.84×10^{4}**
F_{13}	**7.11×10^{3}**	9.69×10^{5}	1.17×10^{4}	1.34×10^{4}	7.68×10^{3}
F_{14}	2.38×10^{4}	4.45×10^{4}	**1.06×10^{3}**	1.50×10^{3}	2.47×10^{3}
F_{15}	1.47×10^{4}	3.96×10^{5}	1.24×10^{3}	5.46×10^{2}	**4.52×10^{2}**
F_{16}	7.24×10^{2}	7.49×10^{2}	7.02×10^{2}	6.69×10^{2}	**1.58×10^{2}**
F_{17}	2.63×10^{2}	2.30×10^{2}	1.72×10^{2}	1.42×10^{2}	**6.78×10^{1}**
F_{18}	1.42×10^{5}	4.88×10^{5}	**5.08×10^{4}**	6.61×10^{4}	7.81×10^{4}
F_{19}	2.04×10^{4}	1.21×10^{5}	3.53×10^{3}	4.92×10^{3}	**2.08×10^{3}**
F_{20}	2.73×10^{2}	3.26×10^{2}	2.58×10^{2}	2.58×10^{2}	**1.31×10^{2}**
F_{21}	2.68×10^{2}	2.69×10^{2}	2.80×10^{2}	2.75×10^{2}	**2.28×10^{2}**
F_{22}	3.38×10^{3}	3.04×10^{3}	1.02×10^{2}	1.01×10^{2}	**1.00×10^{2}**
F_{23}	4.10×10^{2}	4.23×10^{2}	4.35×10^{2}	4.20×10^{2}	**3.90×10^{2}**
F_{24}	5.11×10^{2}	5.24×10^{2}	4.73×10^{2}	4.74×10^{2}	**4.51×10^{2}**
F_{25}	4.08×10^{2}	4.57×10^{2}	3.95×10^{2}	3.94×10^{2}	**3.87×10^{2}**
F_{26}	1.77×10^{3}	1.89×10^{3}	4.51×10^{2}	**2.80×10^{2}**	1.34×10^{3}
F_{27}	5.32×10^{2}	5.35×10^{2}	5.19×10^{2}	5.16×10^{2}	**5.09×10^{2}**
F_{28}	4.65×10^{2}	5.68×10^{2}	4.26×10^{2}	**3.99×10^{2}**	4.00×10^{2}
F_{29}	6.61×10^{2}	7.62×10^{2}	7.34×10^{2}	7.31×10^{2}	**5.37×10^{2}**
F_{30}	9.42×10^{3}	4.96×10^{6}	**3.66×10^{3}**	4.97×10^{3}	4.23×10^{3}


#1 Shifted and Rotated Bent Cigar Function
#2 Shifted and Rotated Sum of Different Power Function
#3 Shifted and Rotated Zakharov Function
#4 Shifted and Rotated Rosenbrock's Function
#5 Shifted and Rotated Rastrigin's Function
#6 Shifted and Rotated Expanded Schaffer's F6 Function
#7 Shifted and Rotated Lunacek Bi_Rastrigin Function
#8 Shifted and Rotated Non-continuous Rotated Rastrigin's
#9 Shifted and Rotated Levy Function
#10 Shifted and Rotated Schwefel's Function
#11 Hybrid Function 1
#12 Hybrid Function 2
#13 Hybrid Function 3
#14 Hybrid Function 4
#15 Hybrid Function 5
PSO
TDO
GWO
STDO
SHTDO
Iterations

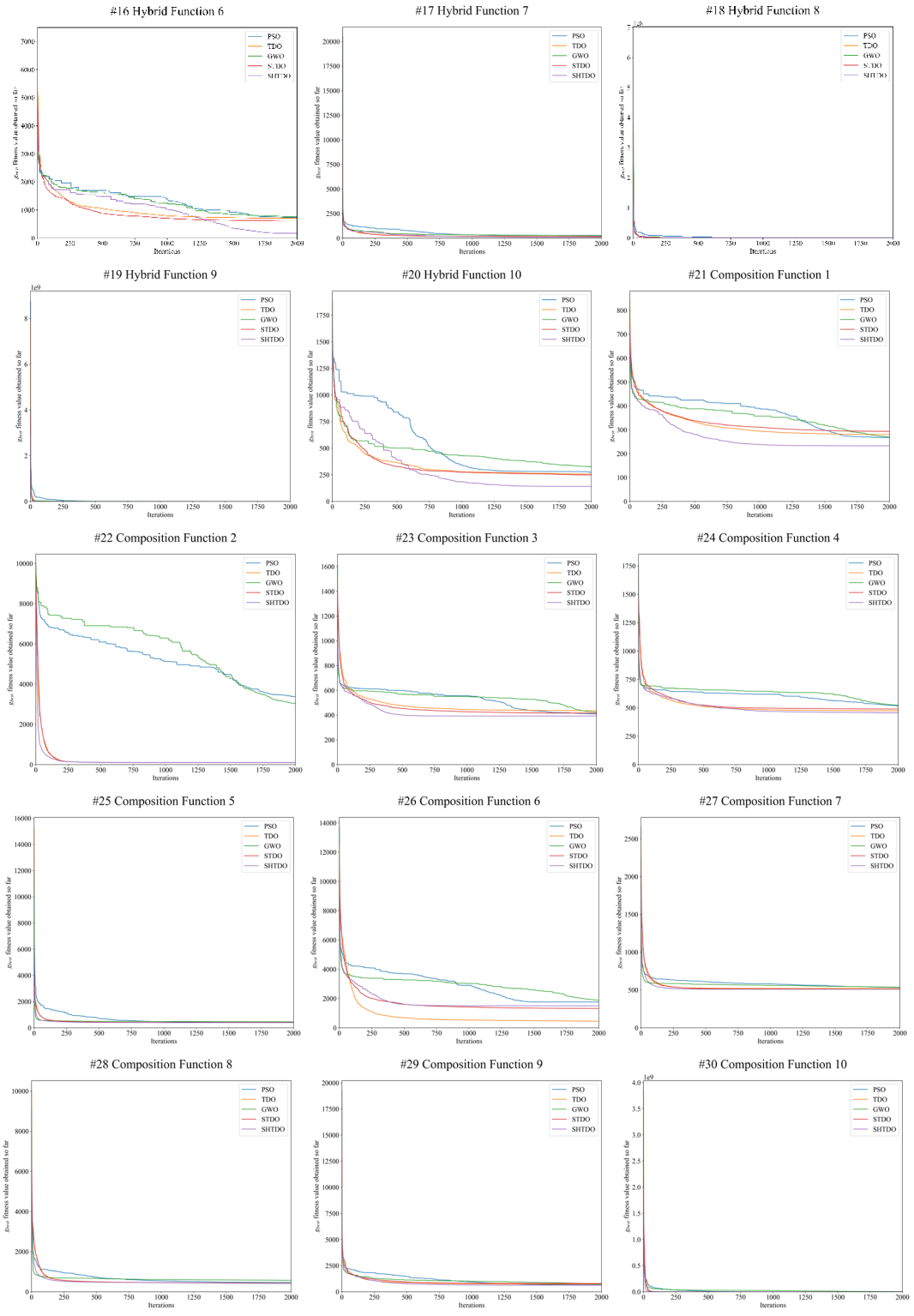


Fig. 6. Convergence trends of different algorithms on the CEC'17 benchmark function

As shown in the Fig. 6, our algorithm is able to converge quickly at an early stage, further illustrating its optimization-seeking capability when coping with optimization problems.

Performance Comparison of Industrial Design Problems. In this section, the performance of our algorithm is compared with other optimization algorithms for solving industrial model problems, and all the parameters of the algorithm are kept the same as the settings of the previous experiment. The experimental results are shown in Table 2.

Table 2. Performance comparison of different algorithms for industrial design optimization problems

	PSO	GWO	TDO	STDO	SHTDO
Welded beam design	1.5070	1.5070	1.5070	1.5070	1.5070
Pressure vessel design	5866.01	5866.01	5866.01	5866.01	5866.01
Telescopic spring design	0.01267	0.01267	0.01267	0.01267	0.01267

The results show that all these models can reach the same optimal value within a finite number of iterations. Therefore, for further comparison, the focus here will be on the convergence speed of each algorithm when optimizing each task as a basis for judging the performance of the different models.

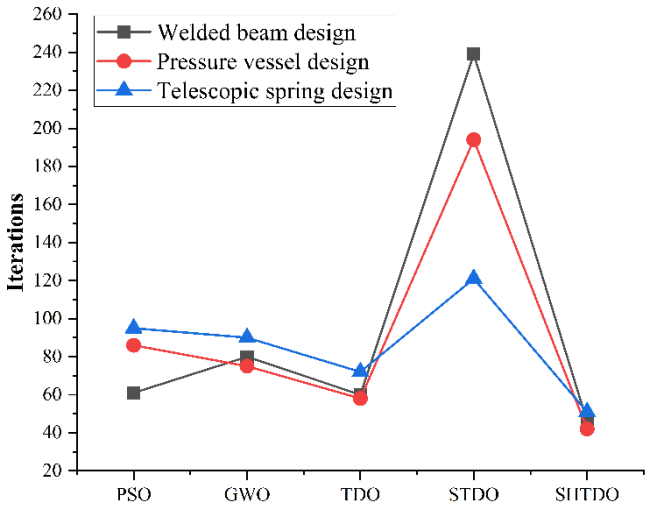


Fig. 7. Comparison of the number of epochs required for different algorithms to reach convergence on industrial design optimization problems

It is specified that the global optimal change of less than 10^{-6} for five consecutive iterations is considered as convergence of the algorithm, and the number of iterations at convergence is recorded, and the results are shown in Fig. 7. It can be seen that our proposed algorithm takes significantly less iterations to reach the optimal value than the comparison algorithm. Therefore, it is concluded that its convergence ability is better.

Performance Comparison of LTR Problem. This section compares the ranking performance of our algorithm with other optimization algorithms for solving the LTR problem, as shown in the Table 3. It can be seen that our algorithm achieves high ranking scores under several metrics. Also, the standard deviations of SHTDO for each ranking evaluation metric are mostly smaller than those of other algorithms under multiple repeated trials, which confirms its better stability. In summary, our optimization algorithm is competitive in handling the LTR problem.

Table 3. Performance comparison of different algorithms on the LTR dataset

Algorithms		NDCG@1	NDCG@3	NDCG@5	NDCG@10	MAP
PSO	Average	0.2073	0.2320	0.2493	0.2785	0.4736
	Std. Dev.	0.0056	0.0025	0.0027	**0.0015**	0.0264

GWO	Average	0.5120	0.4920	0.4875	0.4897	**0.5903**
	Std. Dev.	0.0035	0.0041	0.0045	0.0049	0.0054
TDO	Average	0.5030	0.4867	0.4815	0.4856	0.5670
	Std. Dev.	0.0038	0.0027	0.0034	0.0036	**0.0041**
STDO	Average	0.5269	0.4961	0.4910	0.4939	0.5873
	Std. Dev.	0.0016	0.0009	0.0015	0.0026	0.0163
SHTDO	Average	**0.5342**	**0.4972**	**0.4999**	**0.5030**	0.5899
	Std. Dev.	**0.0011**	**0.0001**	**0.0010**	0.0020	0.0055

6 Conclusion and Outlooks

In this paper, a new SHTDO algorithm is designed by combining the gray wolf operator and the stu mechanism with the TDO algorithm. To validate the performance, training was performed using on several evaluation benchmark functions and data sets. Overall, the experimental results show that the improved algorithm can significantly improve the performance in dealing with different scenarios, especially the LTR problem.

There are still limitations in the current work. For the predictor, only linear regression predictor has been tested in this paper, and no other predictor has been optimized; therefore, it is not known whether there is a better optimization solution. On the other hand, whether the algorithm proposed in this paper can still maintain high performance in other fields, whether there are better directions for improvement, and whether there are better optimization algorithms to face the LTR problem still need to be explored. Therefore, these are the directions of our next work.

7 Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 61872168).

References

[1] G. Salton, Automatic information organization and retrieval, McGraw Hill Text, 1968. https://dl.acm.org/doi/abs/10.5555/1096906
[2] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li, Learning to Rank: From Pairwise Approach to Listwise Approach, in: Proceedings of the 24th International Conference on Machine Learning, 2007.
[3] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, 1995.
[4] D. Simon, Biogeography-Based Optimization, IEEE Transactions on Evolutionary Computation, 12(6)(2008) 702–713.
[5] D. Karaboga, B. Akay, A comparative study of Artificial Bee Colony algorithm, Applied Mathematics and Computation 214(1)(2009) 108–132.
[6] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation. <https://arxiv.org/abs/1003.1409>, 2010 (accessed 10.07.2022).
[7] A.H. Gandomi, A.H. Alavi, Krill herd: A new bio-inspired optimization algorithm, Communications in Nonlinear Science and Numerical Simulation 17(12)(2012) 4831–4845.
[8] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey Wolf Optimizer, Advances in Engineering Software 69(2014) 46–61.
[9] S. Mirjalili, The Ant Lion Optimizer, Advances in Engineering Software 83(2015) 80–98.
[10] S. Arora, S. Singh, Butterfly algorithm with Lèvy Flights for global optimization, in: Proceedings of the 2015 International Conference on Signal Processing, Computing and Control (ISPCC), 2015.
[11] X. Jiang, S. Li, BAS: Beetle Antennae Search Algorithm for Optimization Problems. <https://arxiv.org/abs/1710.10724>, 2017 (accessed 10.07.2022).
[12] M. Dehghani, S. Hubálovský, P. Trojovský, Tasmanian Devil Optimization: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm, IEEE Access 10(2022) 19599–19620.
[13] R. Nallapati, Discriminative Models for Information Retrieval, in: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2004.
[14] J. Xu, H. Li, Adarank: a boosting algorithm for information retrieval, in: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 2007.

[15] J.-Y. Yeh, J.-Y. Lin, H.-R. Ke, W.-P. Yang, Learning to rank for information retrieval using genetic programming, in: Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval, 2007.
[16] W. Khatib, P.J. Fleming, The stud GA: A mini revolution?, in: Proceedings of the Parallel Problem Solving from Nature — PPSN V, 1998.
[17] N.H. Awad, M.Z. Ali, P.N. Suganthan, J.J. Liang, B.Y. Qu, Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization, Technical Report, Nanyang Technological University, Singapore, November, 2016.
[18] S. Mirjalili, A. Lewis, The Whale Optimization Algorithm, Advances in Engineering Software 95(2016) 51–67.
[19] B.K. Kannan, S.N. Kramer, An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design, Journal of Mechanical Design, 116(2)(1994) 405–411.
[20] T. Qin, T.-Y. Liu, Introducing LETOR 4.0 Datasets. <https://arxiv.org/abs/1306.2597>, 2013 (accessed 14.11.2021).