Meta-Heuristic Optimizer Inspired by the Philosophy of Yi Jing

Ho-Kin Tang

Harbin Institute of Technology Shenzhen

Qing Cai

Nanyang Technological University

Sim Kuan Goh (✉ simkuan.goh@xmu.edu.my)

Xiamen University - Malaysia

Research Article

Keywords: Yi optimization, Yin-Yang pair optimization, Cauchy flight, CEC2017, numerical optimization.

Posted Date: January 20th, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1259241/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Meta-heuristic Optimizer Inspired by the Philosophy of Yi Jing

Ho-Kin Tang

School of Science

Harbin Institute of Technology (Shenzhen)

P. R. China

denghaojian@hit.edu.cn

Qing Cai

School of Mechanical

and Aerospace Engineering

Nanyang Technological University

Singapore

qcai@ntu.edu.sg

Sim Kuan Goh

School of Electrical and Computer Engineering

Xiamen University

Malaysia

simkuan.goh@xmu.edu.my

Abstract—Drawing inspiration from the philosophy of Yi Jing, the Yin-Yang pair optimization (YYPO) algorithm has been shown to achieve competitive performance in single objective optimizations, in addition to the advantage of low time complexity when compared to other population-based meta-heuristics. Building upon Yi Jing, we propose the novel Yi optimization (YI) algorithm. Specifically, we enhance the Yin-Yang pair in YYPO with a proposed Yi-point, in which we use Cauchy flight to update the solution, by implementing both the harmony and reversal concept of Yi Jing. The proposed Yi-point balances both the effort of exploration and exploitation in the optimization process. To examine YI, we use the IEEE CEC 2017 benchmarks and compare YI against the dynamical YYPO, CV1.0 optimizer, and four classical optimizers, i.e., the differential evolution, the genetic algorithm, the particle swarm optimization, and the simulated annealing. According to the experimental results, YI shows highly competitive performance while keeping the low time complexity. The results of this work have implications for enhancing a meta-heuristic optimizer using the philosophy of Yi Jing.

Index Terms—Yi optimization, Yin-Yang pair optimization, Cauchy flight, CEC2017, numerical optimization.

I. INTRODUCTION

In the long contest of evolution pressure in nature, living beings have developed different survival strategies. The retaining strategies being adapted are the ones that pass on generations through natural selection. They are often highly optimized as a result of million years of evolution. These amazing and brilliant strategies have been a great source of inspiration for a variety of heuristic optimization techniques [1], for examples, the cuckoo breeding behavior inspired the cuckoo search [2], the hunting strategy of a grey wolf pack inspired the Grey Wolf search [3], the evolution principle of DNA inspired the Genetic Adaptive approaches [4]. Among living species, human is unique, reflected by the unprecedented accumulation in cultural and philosophical contexts. These immortal gems represent the trials of humans to interpret nature. The retaining philosophies, similar to survival strategies, also passed through the contest of time. Some of the ideas are sophisticated and inspired the development of meta-heuristic optimization techniques and related applications [5]–[7].

In ancient China, one of the famous philosophy Yi Jing (I Ching), also known as The Book of Changes, carries the profound philosophy of harmony and unity. The Yi Jing script

reads “Taiji generates two complementary forces, two complementary forces generate four aggregates, four aggregates generate eight trigrams.”. In ancient time, the ancestors try to use the binary hierarchy in describing the phenomenon in the universe. It is interesting that the binary structure shares similarity with the Boolean algebra that modern computing relies on. The ideas in Yi Jing have good potential in inspiring new conceptual developments in algorithms.

In 2016, the Yin-Yang idea had inspired Punathanam et al. to develop the Yin-Yang pair optimization (YYPO) [8], [9]. The algorithm uses two points to form the Yin-Yang pair to perform the optimization task. One point focuses on doing exploitation, while the other point focuses on doing exploration. YYPO is shown to possess significantly lower time complexity and better optimization performance compared to a number of algorithms including Artificial Bee Colony (ABC) [10], Ant Lion Optimizer (ALO) [11], Differential Evolution (DE) [12], Grey Wolf Optimizer (GWO) [3], Multi-directional Search (MDS) [13], Pattern Search (PS) [14] and Particle Swarm Optimization (PSO) [15]. Later in 2017, the dynamical archive was incorporated in YYPO to formulate dynamical YYPO (dYYPO), where its performance has a great improvement [16] compared with YYPO. Also, Heidari et al. combined the idea of PSO and YYPO to develop PSYYPO optimizer, which gives outstanding performance in solving the uncapacitated warehouse location problem [17]. YYPO has shown a wide application capability, e.g., to play an important role in a proposed control scheme in solar energy harvesting [2]. In 2020, 3D-YYPO is proposed to work on wind turbines design and optimization, in which a new splitting method for the three-dimensional problem is used, giving better convergence iteration number and convergence time compared to YYPO and PSO [18].

Another line of developing YYPO is to generalize the pair solution solver to the population-based solver, which can extend the scope of application to the multi-objective optimization. Punathanam et al. developed the Front-based Yin-Yang-Pair Optimization (F-YYPO) that aims for solving multi-objective problems [19]. It works with multiple Yin-Yang pair of points to handle conflicting objectives and utilizes the non-dominance-based approach to sort points into fronts.

Fig. 1. Illustration of the conceptual difference between the Yin-Yang pair optimization (YYPO) and the Yi algorithm (YI). (a) In the Yin-Yang pair of YYPO, two points P_1 and P_2 are used to facilitate the exploration and exploitation. P_i splits with the corresponding search radius δ_i , where it is modified by the expansion/contraction factor α in the archive stage. (b) The proposed Yi-point of YI uses Cauchy flight to strike a balance between exploration and exploitation. The flight scope ϵ is divided by the decay rate σ in the archive stage.

The algorithm has been benchmarked on the set of 10 multi-objective problems CEC2009 [20]. Compared with popular algorithms such as MODE-RMO [21], MOGWO [22], and NSGA-II [23], F-YYPO performs better than its competitors and requires much lesser time to solve the CEC2009 problems. F-YYPO was also used to optimize the Stirling machine system [19] and solve the multiobjective dynamical optimization problems [24]. In addition, a recent population-Based Yin-Yang-pair Optimization (PYYPO) is proposed and used to solve the multi-layer perceptron neural network with application to the medical data [25].

In this paper, we are inspired by another important aspect of the philosophy of Yi Jing – the reversal property: Taiji can generate Yin and Yang, and Yin and Yang can also combine back to form Taiji. We unify the Yin-Yang pair into a single Yi-point to achieve an enhanced balance between the exploration process and the exploitation process. We proposed a highly competitive Yi algorithm (YI), in which we replace the Yin-Yang pair with the Yi-point empowered by the heavy-tailed Cauchy flight.

To scrutinize the proposed YI, the benchmark from CEC2017 is used, which contains 29 challenging functions for optimization [26]. The proposed YI is compared against dYYPO [16], CV1.0 [27], and a few classical algorithms, i.e., DE [12], PSO [15], Genetic Algorithm (GA) [4], Simulated Annealing (SA) [28]. These classical algorithms have been the fundamental building blocks of many recent algorithms. The comparison allows us to assess YI, which builds upon the philosophy of Yi Jing, by showing Yi's relative performance to these classical algorithms that have drawn inspirations from other sources in nature originally. From the empirical results, we find that YI outperforms the classical algorithms, the dYYPO, and CV1.0 in most of the functions.

This paper is organized as the following. In section II, we describe the proposed Yi algorithm. In section III, we outline the settings of the CEC2017 experiment and its result on 10D, 30D, and 50D. In section IV, we discuss and compare the algorithm performance between the results from Yi, dYYPO, CV1.0, PSO, DE, GA and SA, and in section V we give conclusions.

II. ALGORITHM DESCRIPTIONS

In this section, we discuss the algorithm of the proposed YI. We first review the algorithm detail of YYPO that includes two stages, the splitting phase and the archiving phase. Then we discuss the newly proposed Yi-point, which performs the function of both exploration and exploitation, and outline the algorithm detail of YI, and give the pseudocode.

A. Review of YYPO

The Yin-Yang pair optimization is inspired by the duality of opposite forces conflicting in nature depicted in the Yi Jing. If one could foster the balance between two forces, it results in harmony. In the field of evolutionary computing, two conflicting behaviors are exploitation and exploration. We illustrate the core idea of the algorithm in Fig. 1(a).

YYPO algorithm starts with two randomly initialized solutions (P_1 and P_2). The fitter one of the two points is nominated as P_1 and the other as P_2 . P_1 is designed to focus on exploitation, and P_2 is designed to focus on exploration. The three required parameters in terms of the minimum and maximum number of archive updates (I_{min} and I_{max}) and the expansion/contraction factor α need to be specified. The number of archive updates is randomly generated between I_{min} and I_{max} . When the iteration loop is initiated, the fitness of the two points are calculated and compared. During the updates, if P_2 becomes fitter than P_1 , the points and their

corresponding δ values are interchanged. This ensures that the loop starts with the fitter point as P1. We summarize the two main stages of YYPO as the following. The interested reader can refer to Ref. [8], [9] for details of YYPO.

1) *The splitting stage*: The inputs to the splitting stage are one of the points (P_1 or P_2) and its corresponding search radii (δ_1 and δ_2). In the splitting stage, they use two strategies that being selected randomly, namely one-way splitting and D-way splitting. Two methods are selected with equal probability in the original proposal. Later, the probabilities of choosing splitting strategy has been optimized $p_{D-way} = (\frac{D}{D+5})^2$, where the choice depends on the dimension D of the problem [9].

In the one-way splitting strategy, two solutions are generated from P_i using one random number. In each new solution, only one element is changed.

$$P_{i,new}^j = P_i + r\delta_i\hat{e}_j \quad (1)$$

$$P_{i,new}^{D+j} = P_i - r\delta_i\hat{e}_j, \quad (2)$$

where r is a random number between 0 and 1, the superscript denotes the index of a new solution, $\hat{e}_j$ is the unit vector along j direction and $j = 1, 2, \dots, D$. To generate $2D$ new solutions for both P_1 and P_2 after splitting, we need $2D$ random numbers.

In the D-way splitting strategy, we add a random vector $\vec{r}$ to generate new solution, in which all the elements changed.

$$P_{i,new} = P_i + \vec{r}\delta_i/\sqrt{2}, \quad (3)$$

where $\vec{r}$ is randomly distributed between -1 and 1. To generate $2D$ new solutions for both P_1 and P_2 after splitting, we need $2D \times D$ random numbers. The D-way splitting process is demonstrated in Fig. 2.

2) *The archive stage*: The archive stage is initiated after the required number of archive updates I have been reached. The archive contains $2I$ points at this stage, corresponding to the two points P_1 and P_2 being added at each update before the splitting stage. The search radius δ_i is updated as follows,

$$\delta_1 = \delta_1 - \delta_1/\alpha \quad (4)$$

$$\delta_2 = \delta_2 + \delta_2/\alpha \quad (5)$$

Initially, both δ_1 and δ_2 are set to be 0.5. During the updating process, the searching radii δ_1 of P1 decreases to facilitate the exploitation task, while δ_2 of P2 increases to enhance the exploration ability.

At the end of the archive stage, the archive matrix is set to null, and a new value for the number of archive updates I is randomly generated within the given bounds, e.g., I_{min} and I_{max} . In the latter development, dYYPO used dynamical archiving to achieve better control over the exploration and the exploitation during different periods in the search, in which I increases from I_{min} in the beginning phase, to I_{max} at the ending phase [16].

B. The proposed YI

The Yi Jing promotes the concept of the infinite cycles of changes in all observing phenomena. This requires the

Fig. 2. Schematic illustration of the splitting process. We demonstrate the D-way splitting strategy in Eq. 3, where the search range is $\delta_i/\sqrt{2}$ associated with P_i . The best among the generated solution would be picked for the seed of next splitting. The splitting process continues until reaching the archive stage.

reversal property of the Yin-Yang.: Taiji can generate Yin and Yang, and Yin and Yang can also combine back to form Taiji. Inspired by the reversal property between Yin-Yang and Taiji, we combine the two points into a unified point, where we call it Yi-point. The main concept is illustrated in Fig. 1(b). We have three major changes in YI compared to YYPO: (1) replace the Yin-Yang pair with the Yi-point, (2) use the Cauchy flight instead of one-way splitting and D-way splitting, (3) decrease I from I_{max} in the beginning phase to I_{min} at the ending phase. We summarize two main stages and give the outline of YI in the following.

1) *The splitting stage*: We replace the Yin-Yang pair with the Yi-point, in which the update needs to strike a balance between the exploration and the exploitation. In our scheme, we use the profound Cauchy flight to perform the task. It has been found that heavy tailed functions like Levy flight or Cauchy flight successfully capture the characteristics of the flight trajectories of many animals and insects [29]–[31], which maximize the efficiency of searching foods. The underlying reason of efficiency is the balance between exploration and exploitation. The splitting equation is given as follows,

$$P_{0,new} = P_0 + \epsilon \otimes \delta_{Cauchy}(\lambda), \quad (6)$$

where δ_{Cauchy} is the random number vector generated from Cauchy distribution with a stability parameter λ , p_0 and p_{new} are the solutions before and after the splitting. The Cauchy distribution is a fat-tailed distribution. We choose $\lambda = 1.5$ for the distribution that provides the chance of far-field exploration without losing intensified local exploitation. Also, we control the scope of the flight by ϵ , initialized as the dimension D . It should be noted that the proposed Cauchy flight is just one of the strategies to update the Yi-point; other schemes are not yet explored.

2) *The archive stage*: YI adapts the framework of dynamical archiving in dYYPO. The archive stage is initiated after the required number of archive updates I have been reached. We first set the P_0 as the best solution found P_{gbest} . Then we update the flight scope by $\epsilon = \epsilon/\sigma$.

The difference between YI and dYYPO is that in YI we decrease I from I_{max} in the beginning phase to I_{min} at the ending phase. We divide the whole optimization process into

$I_{max} - I_{min} + 1$ time intervals according to the number of function evaluations done. Starting from I_{max} , I descends by one every time when reaching the archive stage, until approaching I_{min} at the end of the search.

At the start of the search, the search process favors the strategy of exploration, so a large I is preferred, allowing the Yi-point to explore the function space without the hindrance of starting from the best obtained solution. In the ending phase of searching, the exploitation becomes more important, so a smaller I allows the Yi-point to focus on searching for the best solution locally.

3) *Outline of YI*: YI has three user-defined parameters, I_{min} , I_{max} and σ . I_{min} and I_{max} are the minimum and the maximum value of I that defines the number of splitting performed before setting P_0 as the best recorded P_{gbest} , and σ is the decay rate of the Cauchy flight scope.

We outline the pseudocode in Algorithm 1. We first initialize the Yi-point randomly in the bounded space and calculate its fitness. We initialize the archive duration $I = I_{max}$, and the search scope $\epsilon = D$. We also define the next interval T_t that gives the time for the next archive stage.

Before reaching the archive stage, we carry out I time of Cauchy splitting, given in Algorithm 2 where we use Eq. 2. If the updated element of the vector is out of the boundary, we choose a random position for this element. We set P_0 as P_{lbest} , the solution with the best fitness among the splitted $P_{0,new}$. We keep using P_{lbest} as P_0 even the fitness of P_{lbest} is worse than the fitness of P_0 . Moreover, we record P_{lbest} as P_{gbest} if it gives a better fitness.

We reach the archive stage once the counter i exceeds T_t . We rescale the scope by dividing it by σ , and decrease I by one. This allows us to perform more local searches and with a shorter archive duration. To maintain the elitism of the Yi-point, we also set P_0 as P_{gbest} . The splitting stage and archive stage repeat until the stopping criteria are met. At the end of the search, we give out the best solution P_{gbest} we found.

We provide the pseudocode of the Yi algorithm in the text, including the main algorithm and the splitting function.

III. EXPERIMENTS AND RESULTS

This section first describes the benchmark data-set from CEC 2017 that is used to assess our proposed method compared against dYYPO and CV1.0. The main properties of these algorithms are shown in Table I. Next, we discuss the parameters of the proposed method. Subsequently, we describe the experimental results and show the statistical tests. We also provide the time complexity analysis.

A. Benchmarks

To validate our algorithm, we use the CEC 2017 benchmark: Single Objective Bound Constrained Real-Parameter Numerical Optimization [26], which comprises four main groups of functions, namely, unimodal functions (F1-F3), simple multimodal functions (F4-F10), hybrid functions (F11-F20) and composition functions (F21-F30). In total, there are 29 functions with diverse search spaces.

Algorithm 1: Yi algorithm

```

Initialize the Yi-point  $P_0$  randomly;
Calculate the fitness of  $P_0$ ,  $f(P_0)$ ;
Set flight scope  $\epsilon = D$ ;
Set archive duration  $I = I_{max}$ ;
Set counter  $i = 0$ ;
Assign the next-interval time
 $T_t = T_{max}/(I_{max} - I_{min} + 1)$ ;
while stopping criteria do
    if  $i > T_t$  then
         $\epsilon = \epsilon/\sigma$ ;
         $I = I - 1$ ;
         $T_t = T_{max}(I_{max} - I + 1)/(I_{max} - I_{min} + 1)$ ;
    end
    Cauchy splitting of  $P_0$  give  $P_{lbest}$ ;
    Record  $P_{gbest} = P_{lbest}$  if  $f(P_{lbest}) < f(P_{gbest})$ ;
    Set  $P_0 = P_{lbest}$ ;
     $i = i + 1$ ;
    if  $i == I$  then
        Set  $P_0 = P_{gbest}$ ;
        Reset counter  $i = 0$ ;
    end
end

```

Algorithm 2: Cauchy splitting function.

```

Input  $P_0$ ;
for  $k = 1$  to  $2 \times D$  do
    Cauchy flight process  $P_{0,new}^k = P_0 + \epsilon \otimes \delta_{Cauchy}$ ;
end
Choose the best in  $P_{0,new}^k$  to output  $P_{lbest}$ 

```

B. Experimental Settings

In all the searches performed in this paper, the parameters are chosen in consideration of convergence in limited times of function evaluation. We use $I_{min} = 6$, $I_{max} = 15$ to divide the whole process into 11 intervals. The number of intervals cannot be too small, as this ensures the desired scope of the Lévy flight at the end of the search to perform good exploitation tasks. For the initial scope and the decay rate, we choose $\epsilon_0 = D$ and $\sigma = 3$. The search scope increases with the dimensions of the problem to secure the exploration coverage in the function space.

To assess YI, we compare it with the classic version of DE, PSO, GA and SA, as well as a Lévy flight-based optimizer CV1.0, and the state-of-the-art dYYPO in YYPO family. We followed their experimental settings. Both papers used the stopping criteria of $10000 \times D$ function evaluations and each benchmark problem was run 51 times. We used the same numbers for our experiments to facilitate fair comparisons.

C. Results of YI on CEC 2017

We also presented our experimental results in terms of best, worst, mean and standard deviation the error values. The error values are computed by calculating the difference expected and

the desired solution. Table III, IV and V showed the statistics of our experimental results for the test problems, for dimension $D = 10, 30, 50$, respectively.

D. Algorithms

To quantitatively assess our methods, we compare our method against DE, PSO, GA, SA, CV1.0 and dYYPO algorithms, to examine the competitiveness of our proposed method. We compare the main features of these algorithms in Table I, and briefly discuss the algorithms in the following texts, for details please refer to the reference.

DE is a simple and one of the most powerful population-based heuristic for global optimization [12]. It is a popular variant of evolutionary algorithms (EA) for multi-dimensional real-valued search spaces. Algorithmically, DE modified the mutation strategies of EA. DE generates new vectors (i.e., solutions) by summing the weighted difference between two population vectors to a third vector. Despite its simplicity, it demonstrates robustness to multi-modal problems and complex constrained optimization problems [32].

PSO is a population based metaheuristic algorithm inspired by the behavior of how a flock of birds move together [15], [33]. The movement of each bird, like a particle, is affected by their own experience and other birds experience. For each bird, they have their own experience best fitness locally (lbest). And the global best fitness (gbest) is defined as the best point found among all birds. The position of gbest and lbest affect the movement of the birds in the update process, in term of the social adjustment by gbest, the self adjustment by lbest and the inertia.

GA is a population based metaheuristic algorithm in which the candidate solutions to an optimization problem are evolved toward better solutions [4]. Each candidate solution, called a chromosome, has a set of properties that can be mutated and altered. The process mimics how genetic evolution occurs. GA first initializes a population of solutions and then improves it through repetitive application of the mutation, crossover, inversion, and selection operators, which are biologically inspired. For each new solution to be produced, a pair of parent solutions are selected for breeding using operators, like the crossover for exploration and the mutation for exploitation. Subsequently, better solutions are selected to form the next generation that replaces the old generation and maintains elitism.

SA is a stochastic global search optimization algorithm inspired by the thermodynamic property in materials [28]. When the temperature is high, the system can be excited to different states without the constraints of energy barriers. While at low temperature, the system could be trapped in the state with a local minimum in energy. SA proposes the new solution and accept it according to the Boltzmann weight of fitness by using the metropolis algorithm. So in the beginning of the search at a high temperature, we can explore variable phase space freely. The temperature is then slowly decreased to force the search to converge to a minimum.

A number of studies have used Lévy flight process to enhance the performance of the optimizer [34]–[37]. CV1.0 [27] was one of them as a variant of the cuckoo search [31]. The Cauchy-based global search was used in the initial phase for exploration, then the Grey wolf optimization algorithm (GWO) [3] was utilized in the remaining phase for exploitation.

We have chosen the classical implementation of DE, PSO, GA, and SA that are provided in the python package scikit-opt [38]. Population size of 50 is used for DE, PSO and GA. The inertia weight w , cognitive parameter $c1$, social parameter $c2$ are set to be 0.9, 2.0, 2.0. The F and crossover rate of DE are chosen to be 0.5 and 0.001. In GA, the probability of mutation and crossover are chosen as 0.001 and 0.9. In SA, min temperature, max temperature, long of chain, cooldown time are set to be $1e-7$, 100, 300, 150. These hyper-parameters are chosen to be the values recommended by scikit-opt, except for the $c1$ and $c2$ of PSO. Here, we use the values recommended by [15], which produces better optimization results for PSO. The experimental results of CV1.0 and dYYPO are extracted from their respective papers. One-tail two-sample t-test is applied to assess the relative performance of our proposed methods against CV1.0 and dYYPO. The outcome of the comparison is provided in Table VI.

We use + to signify that a baseline algorithm under consideration is better than our proposed algorithm, – to signify the opposite, and = to signify either they are performing similarly without statistical significance or the comparison is irrelevant. From the last row of Table VI, the total count of the win, tie, loss (w/t/l) of CV1.0 and dYYPO against Yi are shown. It is observed that the proposed algorithm is better than both CV1.0 and dYYPO the majority of the time. Yi won CV1.0 for 23 out of 29 functions while it won dYYPO for 21 out of 29 functions. The only irrelevant comparison was found when comparing dYYPO with Yi for F3. Yi was found to be much better even by comparing their mean and standard deviation; however, the standard deviation of dYYPO was too large to make statistical relevance. If dYYPO had obtained a more stable and reliable result, Yi would win dYYPO for 22 out of 29 functions.

In addition, Yi is generally found to obtain a relatively smaller standard deviation across runs (Table VI), which demonstrates the stability of Yi in convergence. Despite the computational simplicity of Yi, it achieved a good balance between exploration and exploitation in the benchmark.

E. Time Complexity of Yi

The time complexity is calculated using the testing scheme provided in the CEC2017 [39]. T_0 is counted by 1,000,000 times of basic operations, including addition, multiplication, exponentiation. T_1 is counted by running 200,000 times the optimization function F18. T_2 is counted by first running five trials of the Yi algorithm with the maximum number of function evaluations as 200,000, then averaging between the trials. The time complexity of the algorithm is calculated by $(T_2 - T_1)/T_0$. We summarize the time complexity in Table II. The time complexity of Yi just increases slightly

TABLE I
BRIEF COMPARISON BETWEEN ALGORITHMS

Properties	PSO	DE / GA	SA	CV1.0	dYYPO	Yi
Inspiration	Swarm of birds	Genetic evolution	Metallurgy in physics	Breeding of Cuckoo	Yin-yang concept	Yi Jing concept
Population based	Yes	Yes	Yes	Yes	No	No
No of initial points	population	population	population	population	Two	One
User defined parameter	Four	Three	Three	Three	Three	Three
function eval. in step	population	population	population	population	4D	2D

with the dimensionality. This is similar to what happens in dYYPO. The low time complexity is beneficial when we perform searches in high dimensions. Especially the real-time problem today is complicated and often requires to search in high dimensional function space.

TABLE II
TIME COMPLEXITY ($T_0 = 11.61$)

D	T_1	T_2	$(T_2 - T_1)/T_0$
10	4.50	24.44	1.72
30	6.53	27.32	1.79
50	8.95	30.47	1.85

IV. EXPERIMENTAL RESULTS ANALYSIS

In this section, we analyze the main experimental results of the proposed method, in terms of the convergence to a good solution and time complexity. Based on the results, we also discuss the concept of YI in view of its algorithm complexity and competitive performance against DE, PSO, GA, SA, dYYPO and CV1.0.

A. Performance in CEC 2017

Generally, YI performs consistently well in different types of functions. DE performs comparably well with YI in uni-modal and multi-modal functions F1-F10, while SA performs comparably well with YI in composition function F21-F30.

- Convergence and handling local minima: In the uni-modal functions, YI performed very well in F3 Zakharov function, converging to high precision near the global minimum in all the dimensions. Its error in means is three orders of magnitude smaller than dYYPO in 50D. The result indicates the exploitation task is done much better in YI than in dYYPO. However, YI does not outperform dYYPO in F1 Bent Cigar function, where the narrow ridge of local minimum exists; this might be due to the multi-strategy of splitting in dYYPO, enabling it performs better search along the ridge. In the multi-modal functions F4 to F10, YI outperformed both dYYPO and CV1.0 in most cases. It demonstrated the competitiveness of YI in escaping different types of local minimums. Especially in F9 Lévy Function, where a huge number of local minimum presents, Yi can achieve two orders of magnitude better in error. YI is highly competitive with DE, except for the function where the ridge or the large area of plateau exist, like F1 Bent Cigar function, F4 Rosenbrock function, F6 Schaffer's Function and F9

TABLE III
STATISTICAL RESULTS FOR 10D

	Best	Worst	Mean	Std
F1	7.65E-01	1.07E+04	2.62E+03	2.60E+03
F3	1.14E-06	4.97E-06	2.81E-06	8.87E-07
F4	7.90E-03	3.01E-02	1.83E-02	5.05E-03
F5	3.98E+00	3.08E+01	1.15E+01	5.36E+00
F6	5.20E-04	2.46E-03	8.21E-04	2.60E-04
F7	3.82E+00	3.20E+01	2.13E+01	5.51E+00
F8	1.99E+00	2.29E+01	9.91E+00	4.66E+00
F9	2.81E-07	8.82E-07	5.84E-07	1.15E-07
F10	3.60E+00	6.96E+02	2.89E+02	1.88E+02
F11	1.17E+00	2.03E+01	7.88E+00	4.82E+00
F12	9.39E+02	4.98E+04	1.43E+04	1.31E+04
F13	4.58E+02	1.73E+04	5.52E+03	4.88E+03
F14	2.20E+01	9.46E+01	4.40E+01	1.64E+01
F15	1.17E+01	1.28E+02	5.22E+01	2.62E+01
F16	1.06E+00	1.20E+02	5.20E+00	1.63E+01
F17	2.03E+00	3.88E+01	2.39E+01	9.32E+00
F18	1.26E+02	2.15E+04	5.39E+04	5.75E+03
F19	3.25E+00	5.64E+01	2.13E+01	1.35E+01
F20	1.00E+00	4.60E+01	2.04E+01	9.75E+00
F21	1.00E+02	2.27E+02	1.57E+02	5.58E+01
F22	1.00E+02	1.03E+02	1.02E+02	6.78E-01
F23	3.03E+02	3.24E+02	3.12E+02	4.49E+00
F24	1.22E-02	3.53E+02	2.90E+02	9.01E+01
F25	3.98E+02	4.46E+02	4.16E+02	2.24E+01
F26	1.46E-02	3.00E+02	2.65E+02	5.88E+01
F27	3.89E+02	3.98E+02	3.93E+02	2.25E+00
F28	3.00E+02	6.12E+02	3.06E+02	4.32E+01
F29	2.33E+02	2.78E+02	2.47E+02	1.12E+01
F30	1.03E+03	1.24E+06	1.05E+05	3.11E+05

Levy function. Also, YI outperforms both PSO, GA and SA in most of the functions.

- Hybrid function: The hybrid functions are complicated functions involving different basic functions for different subcomponents of the variable. The efficacy of search in these functions highly depends on the searching strategy. YI and dYYPO share many similarities in the strategy design, while CV1.0 is in a completely different realm. We expect the performance of YI and dYYPO are mostly consistent. Although YI still performs better in many cases comparing to dYYPO and CV1.0, it does not significantly outperform dYYPO in any of the hybrid functions. In F13, F15, F19, dYYPO, GA and DE is better than YI significantly. In these three hybrid functions, the F1 Bent Cigar function is involved; this could result in the weak performance of YI.
- Composition functions: YI performs consistently better in the composition function than DE, PSO, GA, dYYPO and CV1.0, in which sub-functions properties are merged. SA

TABLE IV
STATISTICAL RESULTS FOR 30D

	Best	Worst	Mean	Std
F1	1.91E+02	2.11E+04	6.63E+03	6.38E+03
F3	6.01E-04	1.50E-03	9.73E-04	1.78E-04
F4	5.21E+00	1.19E+02	8.18E+01	2.17E+01
F5	3.48E+01	1.03E+02	6.22E+01	1.54E+01
F6	6.20E-03	3.04E-01	3.78E-02	6.15E-02
F7	6.82E+01	1.58E+02	9.85E+01	1.81E+01
F8	3.98E+01	1.08E+02	7.46E+01	1.60E+01
F9	9.75E-05	1.75E+01	1.81E+00	3.05E+00
F10	1.35E+03	3.13E+03	2.34E+03	4.41E+02
F11	3.10E+01	1.64E+02	8.35E+01	3.20E+01
F12	2.82E+04	1.26E+06	3.45E+05	2.89E+05
F13	8.81E+03	1.02E+05	4.34E+04	2.22E+04
F14	1.75E+02	1.15E+04	1.88E+03	2.20E+03
F15	7.99E+03	5.66E+04	2.46E+04	1.29E+04
F16	5.65E+01	1.01E+03	4.83E+02	2.11E+02
F17	3.28E+01	2.41E+02	1.13E+02	5.92E+01
F18	1.15E+04	2.18E+05	7.80E+04	4.19E+04
F19	7.22E+02	5.30E+04	1.08E+04	1.25E+04
F20	3.38E+01	3.24E+02	1.38E+02	7.63E+01
F21	2.40E+02	3.10E+02	2.66E+02	1.39E+01
F22	1.00E+02	1.05E+02	1.01E+02	1.39E+00
F23	3.80E-02	4.47E+02	4.12E+02	1.87E+01
F24	4.58E+02	5.59E+02	4.95E+02	1.93E+01
F25	3.83E+02	3.87E+02	3.87E+02	9.24E-01
F26	2.00E+02	2.18E+03	1.56E+03	4.34E+02
F27	4.74E+02	5.35E+02	5.08E+02	1.32E+01
F28	3.00E+02	4.89E+02	3.36E+02	5.73E+01
F29	4.53E+02	7.64E+02	5.43E+02	6.56E+01
F30	9.13E+03	4.10E+04	2.39E+04	7.94E+03

TABLE V
STATISTICAL RESULTS FOR 50D

	Best	Worst	Mean	Std
F1	2.17E+03	3.37E+04	8.94E+03	7.07E+03
F3	7.90E-03	1.69E-02	1.23E-02	1.78E-03
F4	2.85E+01	2.24E+02	1.26E+02	4.68E+01
F5	7.66E+01	2.41E+02	1.47E+02	3.03E+01
F6	2.74E-02	1.21E+00	2.06E-01	2.29E-01
F7	1.28E+02	3.04E+02	2.04E+02	3.82E+01
F8	9.55E+01	2.06E+02	1.40E+02	2.36E+01
F9	1.91E+00	6.05E+02	8.46E+01	1.22E+02
F10	3.09E+03	6.44E+03	4.58E+03	7.28E+02
F11	8.22E+01	3.04E+02	1.68E+02	4.69E+01
F12	2.31E+05	8.72E+06	3.14E+06	1.83E+06
F13	2.04E+04	1.38E+05	6.03E+04	2.48E+04
F14	4.80E+02	4.61E+04	1.49E+04	1.32E+04
F15	5.34E+03	6.01E+04	2.68E+04	1.25E+04
F16	5.10E+02	1.70E+03	1.03E+03	2.87E+02
F17	4.29E+02	1.22E+03	7.36E+02	2.04E+02
F18	3.23E+04	3.41E+05	1.32E+05	7.04E+04
F19	1.13E+03	4.18E+04	1.63E+04	1.32E+04
F20	1.58E+02	8.96E+02	4.82E+02	1.93E+02
F21	2.91E+02	4.19E+02	3.42E+02	2.92E+01
F22	1.00E+02	6.64E+03	2.41E+03	2.59E+03
F23	4.82E+02	6.27E+02	5.62E+02	2.86E+01
F24	5.78E+02	7.10E+02	6.50E+02	3.12E+01
F25	4.60E+02	5.78E+02	5.09E+02	3.44E+01
F26	2.08E+03	3.77E+03	2.72E+03	3.46E+02
F27	5.08E+02	7.77E+02	5.65E+02	4.09E+01
F28	4.59E+02	5.10E+02	4.67E+02	1.63E+01
F29	3.96E+02	1.18E+03	7.40E+02	1.97E+02
F30	8.96E+05	1.94E+06	1.41E+06	2.57E+05

is competitive with YI in handling composition function, as SA optimizes to a global minimum by decreasing temperature repeatedly, which is advantageous in handling composite function in which the different functions are at different energy scales. Even without sharing algorithmic features with SA, YI performs decently in handling composition functions. This proves the YI algorithm can handle complicated function landscape very well to search the global minimum.

- Dimensionality: When we compare the performance of YI with dYYPO [16] in different dimensions, we find YI performs better in the higher dimensions. In 10D, YI is only slightly better than dYYPO in terms of smaller mean errors among all 29 functions. The advantage becomes more and more obvious when coming to 50D, showing the capability of Yi algorithm in complicated landscapes with high dimensions.

B. Sensitivity Analysis

We have tested different choices of hyper-parameters σ , I_{min} and I_{max} . As listed in Table VII, the decay rate σ has a great influence on the optimization result. In the searching process, either decaying too fast ($\sigma = 5$) or too slow ($\sigma = 1.5$) deteriorates the performance of YI. We also find that the performance of YI is not very sensitive to the change in I_{max} and I_{min} , which control the number of intervals and the archive time of each interval in the run.

C. Algorithm Comparison

As YI is a newly proposed variant of YYPO, we first compare YI with other variants of YYPO. Subsequently, we compare YI with other types of algorithms.

YI inherits YYPO's point-based feature and the dynamical archiving of dYYPO. The main difference is that we further reduce the number of points used in the searching from two to one, replacing the Yin-Yang pair with the Yi-point, this further enhances the low time complexity as well as improving the control over the exploration and exploitation. Second, instead of using both one-way splitting and D-way splitting of updating in YYPO, YI use the unified Lévy flight updating. Third, contrary to dYYPO that uses ascending I , we use descending I every time when reaching the archive stage. All these improved features contributed to the better control of the exploration and the exploitation compared to dYYPO, which is believed to performs best among the YYPO family in the general single objective optimization. We have not adapted the above changes to the population-based YYPOs, like F-YYPO and PYYPO, that are tailored to the multi-objective optimizations. This research direction is left for future works.

When compared to the PSO family, similar to PSO memory of its global best point (gbest), YI has temporary memory of the Yi-point before reaching the archive stage. This is also common for other YYPO variants that are using the Ying-Yang pair. However, one main difference with PSO is that YI's memory does not affect the decision to search using

TABLE VI
STATISTICAL RESULTS OF THE PROPOSED ALGORITHM FOR 50D IN COMPARISON WITH OTHERS

			CVI.0		dYYPO		GA		DE		PSO		SA		YI
F1	mean	-	1.00E+10	+	6.60E+03	=	7.40E+03	+	1.92E+03	-	1.21E+11	-	2.79E+08		8.94E+03
	std		0.00E+00		7.10E+03		7.76E+03		1.74E+03		3.74E+10		5.17E+07		7.07E+03
F3	mean	-	1.95E+04	=	4.70E+01	-	2.24E+05	-	8.60E+04	-	2.63E+05	-	6.70E+04		1.23E+02
	std		6.27E+03		2.30E+02		5.83E+04		9.77E+03		8.54E+04		1.60E+04		1.78E+03
F4	mean	=	1.16E+02	=	1.40E+02	=	1.33E+02	+	5.19E+01	-	2.50E+04	+	1.08E+02		1.26E+02
	std		6.27E+03		5.00E+01		4.16E+01		3.70E+01		1.24E+04		5.25E+01		4.68E+01
F5	mean	-	3.41E+02	-	1.90E+02	-	1.94E+02	-	3.16E+02	-	7.20E+02	-	3.91E+02		1.47E+02
	std		8.02E+01		3.90E+01		3.50E+01		1.38E+01		1.03E+02		4.33E+01		3.03E+01
F6	mean	-	4.85E+01	-	3.80E+00	-	2.43E+00	+	1.14E-13	-	8.28E+01	-	5.57E+01		2.06E-01
	std		4.85E+01		2.00E+00		1.03E+00		0.00E+00		1.09E+01		1.22E+01		2.29E-01
F7	mean	-	2.74E+02	-	2.60E+02	-	4.31E+02	-	3.70E+02	-	3.70E+03	-	5.40E+02		2.04E+02
	std		7.29E+01		4.30E+01		7.50E+01		1.52E+01		5.00E+02		6.64E+01		3.82E+01
F8	mean	-	3.29E+02	-	1.90E+02	-	1.86E+02	-	3.16E+02	-	7.42E+02	-	3.95E+02		1.40E+02
	std		7.29E+01		4.70E+01		3.76E+01		1.54E+01		1.13E+02		6.16E+01		2.36E+01
F9	mean	-	1.00E+04	-	3.50E+03	-	4.59E+03	+	6.02E-14	-	2.26E+04	-	2.62E+04		8.46E+01
	std		2.90E+03		1.90E+03		1.77E+03		5.67E-14		4.90E+03		5.84E+03		1.22E+02
F10	mean	-	7.10E+03	=	4.80E+03	-	5.36E+03	-	1.16E+04	-	8.86E+03	-	8.21E+03		4.58E+03
	std		5.34E+02		6.40E+02		7.65E+02		3.76E+02		1.28E+03		8.15E+02		7.28E+02
F11	mean	=	1.66E+02	-	1.90E+02	-	6.87E+02	=	1.57E+02	-	7.89E+03	-	3.91E+02		1.68E+02
	std		3.38E+01		5.20E+01		8.50E+02		1.22E+01		7.29E+03		8.36E+01		4.69E+01
F12	mean	-	1.00E+10	-	7.80E+06	-	3.91E+06	-	8.50E+06	-	3.30E+10	-	5.72E+07		3.14E+06
	std		0.00E+00		5.10E+06		2.03E+06		3.50E+06		1.58E+10		3.43E+07		1.83E+06
F13	mean	-	1.00E+10	+	7.60E+03	+	5.01E+03	+	7.97E+03	-	1.36E+10	-	4.23E+05		6.03E+04
	std		0.00E+00		7.40E+03		4.79E+03		6.22E+03		8.60E+09		2.71E+05		2.48E+04
F14	mean	+	2.05E+02	-	2.90E+04	-	7.25E+05	-	2.37E+05	-	6.01E+06	-	4.89E+05		1.49E+04
	std		2.13E+01		2.80E+04		6.75E+05		1.38E+05		6.70E+06		3.74E+05		1.32E+04
F15	mean	-	1.37E+09	+	8.20E+03	+	8.50E+03	+	3.08E+03	-	2.20E+09	-	3.88E+04		2.68E+04
	std		3.47E+09		7.00E+03		6.09E+03		1.35E+03		3.77E+09		2.52E+04		1.25E+04
F16	mean	-	1.53E+03	-	1.30E+03	-	2.03E+03	-	1.80E+03	-	3.66E+03	-	1.67E+03		1.03E+03
	std		2.74E+02		4.10E+02		4.16E+02		1.78E+02		8.51E+02		4.52E+02		2.87E+02
F17	mean	-	1.25E+03	-	8.90E+02	-	1.51E+03	-	8.47E+02	-	1.11E+04	-	1.43E+03		7.36E+02
	std		1.85E+02		2.70E+02		2.95E+02		1.24E+02		2.33E+04		3.56E+02		2.04E+02
F18	mean	+	5.21E+02	-	1.80E+05	-	1.88E+06	-	2.00E+06	-	5.15E+07	-	3.11E+06		1.32E+05
	std		1.19E+02		8.10E+04		1.45E+06		8.27E+05		8.79E+07		2.47E+06		7.04E+04
F19	mean	+	1.73E+02	+	9.60E+03	=	1.60E+04	+	7.55E+03	-	1.14E+09	-	2.08E+04		1.63E+04
	std		4.17E+02		8.80E+03		1.08E+04		3.57E+03		1.54E+09		1.44E+04		1.32E+04
F20	mean	-	1.05E+03	-	6.50E+02	-	1.24E+03	-	6.49E+02	-	1.69E+03	-	1.06E+03		4.82E+02
	std		2.14E+02		2.80E+02		3.47E+02		1.31E+02		3.16E+02		2.72E+02		1.93E+02
F21	mean	-	5.41E+02	-	4.00E+02	-	3.91E+02	-	5.21E+02	-	9.04E+02	-	6.09E+02		3.42E+02
	std		6.27E+01		4.00E+01		3.87E+01		1.21E+01		1.11E+02		6.97E+01		2.92E+01
F22	mean	-	7.33E+03	-	4.80E+03	-	6.16E+03	-	1.05E+04	-	9.31E+03	-	8.01E+03		2.41E+03
	std		1.99E+03		2.00E+03		9.36E+02		2.99E+03		1.25E+03		3.22E+03		2.59E+03
F23	mean	-	7.74E+02	-	6.50E+02	-	6.80E+02	-	7.38E+02	-	1.36E+03	-	9.59E+02		5.62E+02
	std		8.06E+01		5.70E+01		4.29E+01		1.42E+01		1.77E+02		9.08E+01		2.86E+01
F24	mean	-	8.32E+02	-	7.30E+02	-	7.65E+02	-	8.45E+02	-	1.37E+03	-	1.02E+03		6.50E+02
	std		1.21E+01		7.40E+01		5.00E+01		1.06E+01		1.68E+02		9.47E+01		3.12E+01
F25	mean	-	5.43E+02	-	5.20E+02	-	5.49E+02	=	5.17E+02	-	1.57E+04	=	5.03E+02		5.09E+02
	std		1.51E+01		3.00E+01		3.73E+01		3.77E+01		6.70E+03		3.44E+01		3.44E+01
F26	mean	=	2.48E+03	-	3.40E+03	-	4.03E+03	-	4.08E+03	-	1.19E+04	+	9.57E+02		2.72E+03
	std		1.88E+03		7.80E+02		5.64E+02		1.06E+02		2.25E+03		1.24E+03		3.46E+02
F27	mean	-	7.38E+02	-	6.70E+02	-	8.53E+02	+	5.50E+02	-	1.59E+03	+	4.91E+02		5.65E+02
	std		8.21E+01		7.30E+01		9.70E+01		1.27E+01		3.48E+02		1.35E+01		4.09E+01
F28	mean	-	4.94E+02	-	4.80E+02	-	5.07E+02	-	4.76E+02	-	9.39E+03	-	4.97E+02		4.67E+02
	std		1.93E+01		2.50E+01		1.48E+01		2.23E+01		1.80E+03		3.03E+01		1.63E+01
F29	mean	-	1.69E+03	-	9.80E+02	-	1.47E+03	-	1.09E+03	-	5.82E+03	-	1.69E+03		7.40E+02
	std		2.29E+02		3.10E+02		3.49E+02		1.37E+02		3.01E+03		3.98E+02		1.97E+02
F30	mean	-	4.64E+06	=	1.50E+06	+	1.06E+06	-	2.25E+06	-	1.87E+09	+	8.90E+05		1.41E+06
	std		8.59E+06		3.20E+05		3.09E+05		3.60E+05		1.70E+09		3.98E+05		2.57E+05
(w,t,l)			(3,3,23)		(4,4,21)		(3,3,23)		(8,2,19)		(0,0,29)		(4,1,24)		

TABLE VII
SENSITIVITY ANALYSIS
(COMPARE WITH THE CASE $\epsilon = 3, I_{min} = 6$ AND $I_{max} = 15$)

I_{min}	I_{max}	σ	Win	Tie	Loss
6	15	1.5	0	1	28
6	15	5	2	13	14
1	10	3	2	21	6
16	25	3	6	21	2
6	25	3	0	21	8

the Cauchy flight. No velocity is calculated in YI, which is advantageous in keeping the low time complexity.

In the context of evolution strategy (ES), YI shares some similarities the $(1, \lambda)$ -ES, in which the best among λ mutants becomes the parent of the next generation while the current parent is always disregarded. It is surprising that the reduction of the Yin-Yang pair to the Yi-point leads us to another brilliant strategy in the evolution algorithm. However, we use the heavy-tail Cauchy flight in YI to perform the exploration, instead of the crossover step in the genetic algorithms that are closely related to ES.

YI, being a variant of YYPO, has another conceptual difference with ES, where YI emphasizes the control between

exploration and exploitation. In YI, the Cauchy flight is accompanied by the dynamical archive, which allows excellent control between the exploration and the exploitation throughout different phases of the search. Given the similarity between YI and ES, it is highly motivated to adapt some ideas of ES in YI or YYPO, for instance, using the covariance matrix adaptation [40]. This might further improve the performance of YI or YYPOs.

D. Analysis of the Algorithm

Complexity of an algorithm is determined by the amount of time and space required to execute it. Analysis of the algorithm refers to the process of deriving estimates for the time and space complexity. YI inherits and further improves the low time complexity property of YYPOs. For the search with a maximum number of function evaluation N_{eval} , the time complexity is mainly dominated by the Cauchy flight with complexity of $O(D)$, in which the generation of a new vector requires $2D$ random number. Besides, a small amount of time complexity of $O(N_{eval})$ is spent for logical comparison of fitness. Hence, the overall time complexity of YI is $O(DN_{eval})$.

The actual computational time depends on a number of factors such as the computer being used, the way the data are represented, and with which programming language the code is implemented. Despite the fact that there might be a difference between the actual computational time and estimated time complexity, the above analysis of YI clearly demonstrate the low algorithmic complexity that is beneficial when YI performs searches in high dimensional space.

In terms of space complexity, YI has a lower memory requirement due to the simplification of the YYPOs' archive process. The simplification removes the need to record all the lbest point before the archive stage. The required memory to run the algorithm is proportional to the dimension D , i.e. three vector of dimension D , one for gbest, one for lbest, and one for the currently splited vector.

V. CONCLUSIONS

The paper proposed an algorithm YI for heuristics optimization that draws inspiration from Yi Jing, creating a novel variant in YYPO family. Instead of using the Yin-Yang pair in YYPO, the concept of YI is incorporated into Cauchy flight for enhanced exploration and exploitation while maintaining a trade-off between both of them. The proposed method also inherits the concept of simplicity from the Yi Jing, which provides the advantage of low time complexity. This allows YI to be used in the high dimensional optimization problems. Also, we have provided a brief review of YYPOs family and analyze YI with other algorithms. The analysis allows us to place YYPOs better in the context algorithm and inspires new ideas to develop YYPOs, such as adapting ideas from ES.

According to the experimental results, our proposed methods demonstrated superior performance against CV1.0 and dYYPO on CEC 2017 benchmark that contains numerous

types of challenging functions. The competitive results suggested that the proposed method achieved a good balance between exploration and exploitation. Comparing to dYYPO and CV1.0, YI is better at escaping local minima and handling complicated function landscapes in most cases. In the ridge type of function landscape like Bent Cigar function, YI performs relatively weak. This could be improved if we include other types of searching strategies, like the directional Cauchy flight as in dYYPO. The Yi-point does not limit to use Cauchy flight updating strategy. We can also adapt other strategies, like using the multivariate normal distribution using an evolving covariance matrix in ES [41].

While this work starts a new chapter for the Yi Jing inspired optimizer, the work can be extended in numerous ways. The extension of YI from single-objective optimization to multi-objective or many objective optimizations deserves research attention [42]. The concept of improved control over the exploration and exploitation in YI using the dynamical achieving and the YI point can fuse with other algorithms, including the population base algorithms (e.g., PSO, GA), to propose some competitive optimizer.

STATEMENTS AND DECLARATIONS

Funding and Competing interests

This work is supported by the by the Start-Up Research Fund in HITSZ (Grant No. ZX20210478, Grant No. X20220001) and Shenzhen Yuliang Technology Co., Ltd. (Grant No. HX20210370)

Authorship Contributions

Ho-Kin Tang and Sim Kuan Goh performed acquisition, analysis, interpretation of data, and drafted the work. Qing Cai interpreted the data and revised it critically for important intellectual content.

DATA AVAILABILITY

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

STATEMENTS AND DECLARATIONS

REFERENCES

- [1] S. Mirjalili, J. Song Dong, A. S. Sadiq, and H. Faris, "Genetic algorithm: Theory, literature review, and application in image reconstruction," in *Nature-Inspired Optimizers: Theories, Literature Reviews and Applications*. Springer, 2020, pp. 69–85.
- [2] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, 2009, pp. 210–214.
- [3] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
- [4] A. Brindle, "Genetic algorithms for function optimization," 1980.
- [5] H. Al-Sahaf, Y. Bi, Q. Chen, A. Lensen, Y. Mei, Y. Sun, B. Tran, B. Xue, and M. Zhang, "A survey on evolutionary machine learning," *J. R. Soc. N. Z.*, vol. 49, no. 2, pp. 205–228, Apr. 2019.
- [6] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, Apr. 2018.

- [7] A. Fernandez, F. Herrera, O. Cordon, M. Jose del Jesus, and F. Marcelloni, "Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to?" *IEEE Comput. Intell. Mag.*, vol. 14, no. 1, pp. 69–81, Feb. 2019.
- [8] V. Punnathanam and P. Kotecha, "Reduced Yin-Yang-Pair optimization and its performance on the CEC 2016 expensive case," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2016, pp. 2996–3002.
- [9] —, "Yin-Yang-pair optimization: A novel lightweight optimization algorithm," *Eng. Appl. Artif. Intell.*, vol. 54, pp. 62–79, Sep. 2016.
- [10] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *J. Global Optimiz.*, vol. 39, no. 3, pp. 459–471, Nov. 2007.
- [11] S. Mirjalili, "The ant lion optimizer," *Adv. Eng. Softw.*, vol. 83, pp. 80–98, May 2015.
- [12] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimiz.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.
- [13] K. Deb and S. Agrawal, "A Niched-Penalty approach for constraint handling in genetic algorithms," in *Artificial Neural Nets and Genetic Algorithms*, A. Dobnikar, N. C. Steele, D. W. Pearson, and R. F. Albrecht, Eds. Springer Vienna, 1999, pp. 235–243.
- [14] K. Deep, K. P. Singh, M. L. Kansal, and C. Mohan, "A real coded genetic algorithm for solving integer and mixed integer optimization problems," *Appl. Math. Comput.*, vol. 212, no. 2, pp. 505–518, Jun. 2009.
- [15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Nov. 1995, pp. 1942–1948 vol.4.
- [16] D. Maharana, R. Kommadath, and P. Kotecha, "Dynamic Yin-Yang pair optimization and its performance on single objective real parameter problems of CEC 2017," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, Jun. 2017, pp. 2390–2396.
- [17] A. A. Heidari, O. Kazemzade, and F. Hakimpour, "A new hybrid Yin-Yang-pair-particle swarm optimization algorithm for uncapacitated warehouse location problems," *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XLII-4/W4, pp. 373–379, Sep. 2017.
- [18] D. Song, J. Liu, J. Yang, M. Su, Y. Wang, X. Yang, L. Huang, and Y. H. Joo, "Optimal design of wind turbines on high-altitude sites based on improved Yin-Yang pair optimization," *Energy*, vol. 193, p. 116794, 2020.
- [19] V. Punnathanam and P. Kotecha, "Multi-objective optimization of Stirling engine systems using Front-based Yin-Yang-Pair Optimization," *Energy Convers. Manag.*, vol. 133, pp. 332–348, 2017.
- [20] —, "Front-Based Yin-Yang-Pair optimization and its performance on CEC2009 benchmark problems," in *Smart Innovations in Communication and Computational Sciences*. Springer Singapore, 2019, pp. 387–397.
- [21] X. Chen, W. Du, and F. Qian, "Multi-objective differential evolution with ranking-based mutation operator and its application in chemical process optimization," *Chemometrics Intellig. Lab. Syst.*, vol. 136, pp. 85–96, Aug. 2014.
- [22] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. d. S. Coelho, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Syst. Appl.*, vol. 47, pp. 106–119, Apr. 2016.
- [23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
- [24] V. Punnathanam and P. Kotecha, "Optimization of multi-objective dynamic optimization problems with Front-Based Yin-Yang-Pair optimization," in *Smart Innovations in Communication and Computational Sciences*. Springer Singapore, 2019, pp. 377–386.
- [25] M. Shekhar, "Training Multi-Layer perceptron using Population-Based Yin-Yang-Pair optimization," in *Proceedings of International Conference on Artificial Intelligence and Applications*. Springer Singapore, 2021, pp. 417–425.
- [26] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*, 2017.
- [27] R. Salgotra, U. Singh, and S. Saha, "New cuckoo search algorithms with enhanced exploration and exploitation properties," *Expert Syst. Appl.*, vol. 95, pp. 384–420, Apr. 2018.
- [28] S. Kirkpatrick, C. D. Gelatt, Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.
- [29] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
- [30] C. T. Brown, L. S. Liebovitch, and R. Glendon, "Lévy flights in dobe ju/hoansi foraging patterns," *Hum. Ecol. Interdiscip. J.*, vol. 35, no. 1, pp. 129–138, Feb. 2007.
- [31] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *J. Comput. Phys.*, vol. 226, no. 2, pp. 1830–1844, Oct. 2007.
- [32] M. Georgioudakis and V. Plevris, "A comparative study of differential evolution variants in constrained structural optimization," *Frontiers in Built Environment*, vol. 6, p. 102, 2020.
- [33] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE Int. Conf. Evol. Comput. proceedings. IEEE world Congr. Comput. Intell. (Cat. No. 98TH8360)*. IEEE, 1998, pp. 69–73.
- [34] R. Jensi and G. W. Jiji, "An enhanced particle swarm optimization with levy flight for global optimization," *Appl. Soft Comput.*, vol. 43, pp. 248–261, Jun. 2016.
- [35] S. K. Chini, A. Bagheri, and F. Najafi, "PSOSCALF: A new hybrid PSO based on sine cosine algorithm and levy flight for solving optimization problems," *Appl. Soft Comput.*, vol. 73, pp. 697–726, Dec. 2018.
- [36] H. A. Abdulwahab, A. Noraziah, A. A. Alsewari, and S. Q. Salih, "An enhanced version of black hole algorithm via Lévy flight for optimization and data clustering problems," *IEEE Access*, vol. 7, pp. 142 085–142 096, 2019.
- [37] S. Amirsadri, S. J. Mousavirad, and H. Ebrahimpour-Komleh, "A Lévy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training," *Neural Comput. Appl.*, vol. 30, no. 12, pp. 3707–3720, Dec. 2018.
- [38] "scikit-optimize," <https://scikit-optimize.github.io/>, 2021.
- [39] N. H. Awad, M. Z. Ali, P. N. Suganthan, J. J. Liang, and 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*, 2016.
- [40] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 75–102.
- [41] O. M. Shir and A. Yehudayoff, "On the covariance-hessian relation in evolution strategies," *Theor. Comput. Sci.*, vol. 801, pp. 157–174, Jan. 2020.
- [42] R. Tanabe and H. Ishibuchi, "A review of evolutionary multimodal multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 193–200, Feb. 2020.