Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

# Society Deciling Process: A Socio-Inspired Meta-Heuristic Algorithm

***E. Pira*** [*]***, A. Rouhi***

*Faculty of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran.*

**Article Info**

**Article History:**
Received 14 April 2024
Reviewed 28 May 2024
Revised 18 July 2024
Accepted 28 July 2024

**Keywords:**
Society deciles
Meta-heuristic
Test function
Cec 2019
Cec 2022
Convergence speed

[*]Corresponding Author's Email Address: *pira@azaruniv.ac.ir*

**Abstract**

**Background and Objectives:** The development of effective meta-heuristic algorithms is crucial to solve complex optimization problems. This paper introduces the Society Deciling Process (SDP), a novel socio-inspired meta-heuristic algorithm that simulates the social categorization into deciles based on metrics such as income, occupation, and education. The objective of this research is to introduce the SDP algorithm and evaluate its performance in terms of convergence speed and hit rate, comparing it with seven well-established meta-heuristic algorithms to highlight its potential in optimization tasks.

**Methods:** The SDP algorithm's efficacy was evaluated using a comprehensive set of 14 general test functions, including benchmarks from the CEC 2019 and CEC 2022 competitions. The performance of SDP was compared against seven established meta-heuristic algorithms: Artificial Hummingbird Algorithm (AHA), Dwarf Mongoose Optimization algorithm (DMO), Reptile Search Algorithm (RSA), Snake Optimizer (SO), Fick's Law Optimization (FLA), Prairie Dog Optimization (PDO), and Gazelle Optimization Algorithm (GOA). Statistical analysis was conducted using Friedman's rank and Wilcoxon signed-rank tests to evaluate the relative performance in terms of exploration, exploitation capabilities, and proximity to the optimum solution.

**Results:** The results demonstrated that the SDP outperforms its counterparts in terms of convergence speed and hit rate across the selected test functions. In statistical tests, SDP showed significantly better performance in exploration and exploitation, leading to a higher proximity to optimum compared to other algorithms. Furthermore, when applied to five complex engineering design problems, the SDP algorithm exhibited superior performance, outmatching the state-of-the-art algorithms in terms of effectiveness and efficiency.

**Conclusion:** The Society Deciling Process (SDP) algorithm introduces a novel and effective approach to optimization, inspired by societal structure dynamics. Its superior performance in convergence speed, exploration and exploitation capabilities, and application to complex engineering problems establishes SDP as a promising meta-heuristic algorithm. This research not only demonstrates the potential of socio-inspired algorithms in optimization tasks but also opens avenues for further enhancements in meta-heuristic algorithm designs.

This work is distributed under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

## Introduction

In the ever-evolving landscape of technology, optimization problems have become pervasive across diverse domains such as science, engineering, management, and economics. The complexity of these challenges, coupled with resource constraints, necessitates optimal solutions. The primary objective in solving optimization problems lies in determining the most suitable values for

variables to either maximize or minimize a given objective function [1]. However, conventional mathematical techniques and heuristic algorithms encounter a significant hurdle—local optima entrapment—when dealing with intricate problems featuring numerous candidate values and multiple local optima.

To address this challenge, meta-heuristic algorithms have emerged as a powerful class of stochastic optimization methods. Distinguished by their gradient-free, problem-independent, and local-optima-free nature, these algorithms offer a departure from traditional approaches [2]. By applying various operators iteratively, they navigate the search space based on an objective function [3], making them well-suited for a wide array of scientific and industrial applications [3].

Meta-heuristic algorithms can be broadly categorized into two groups: individual-solution-based and population-based [4]. The latter, starting with a randomly generated population of solutions, proves more popular due to its enhanced ability to explore and exploit the search space, targeting global optima [5]. Inspired by evolutionary processes, natural phenomena, and social behaviors, population-based meta-heuristic algorithms can be further classified into Evolutionary Algorithms (EAs), Natural Phenomenon (NP) algorithms, and Social Behaviors (SBs) algorithms:

1. Evolutionary Algorithms (EAs) that mimic the process of natural evolution. Genetic Algorithm (GA) [6], and Genetic Programming (GP) [7], Single Candidate Optimizer (SCO) [8], and Attack-Leave Optimizer (ALO) [9] are the well-known samples of this class.

2. *Natural Phenomenon (NP) algorithms* that exploit physical and chemistry principles. Simulated Annealing (SA) [10], Energy Valley Optimizer (EVO) [11], Water Cycle Algorithm (WCA) [12], Nutcracker optimizer (NOA) [13], Orchard Algorithm (OA) [14], Swarm Magnetic Optimizer (SMO) [15], Fusion–fission optimization (FuFiO) [16], and Fick's Law Optimization (FLA) [17], Geometric Mean Optimizer (GMO) [18] and Physics-Inspired Discriminative Classifier (PIDC) [19] can be considered as popular samples of this class.

3. *Social Behaviors (SBs) algorithms* which are divided into two subgroups: Swarm Intelligence (SI) algorithms and Human Behaviors (HB) algorithms. SI algorithms simulate the self-organized and collective behaviors in nature. Actually, these algorithms originate social behaviors of species, such as ants or bees. Ant Colony Optimization (ACO) inspired by foraging behaviors of ants is a popular algorithm in this group. Particle Swarm Optimization (PSO) [20], Snake Optimizer (SO) [21], Prairie Dog Optimization (PDO) [22], Aquila Optimizer (AO) [23], Red Fox Optimization (RFO) algorithm [24], Honey Badger Algorithm (HBA) [1], Reptile Search Algorithm (RSA) [25], Gazelle Optimization Algorithm (GOA) [26], Artificial Hummingbird Algorithm (AHA) [3], Dung Beetle Optimizer [27], Fox optimizer (FOX) [28], Giant Trevally Optimizer (GTO) [29], Mountain Gazelle Optimizer (MGO) [30], and Dwarf Mongoose Optimization (DMO) algorithm [31], Hippopotamus Optimization (HO) [32], Blood-Sucking Leech Optimizer (BSLO) [33], Greylag Goose Optimization (GGO) [34], Genghis Khan Shark Optimizer (GKSO) [35], Ladybug Beetle Optimization (LBO) [36], and Crayfish Optimization Algorithm (COA) [37] can be mentioned as other popular algorithms in this class.

HB algorithms imitate human behaviors in society. Social Group Optimization (SGO) [38] is a popular HB algorithm inspired by the social interaction of members. This algorithm has two phases: *improving* and *acquiring*. In the first phase, each group member increases his/her knowledge by interacting with the best member of the group. In the second phase, each member acquires knowledge from the best member of the group as well as other randomly selected members. Inspired from the council evolution, City Councils Evolution (CCE) [39] is another well-known HB algorithm. Here, councils evolve from the smallest neighbors to the largest ones, regions, and ultimately the whole city is considered. Due to the hierarchical manner of councils' evolution, CCE uses a hierarchical structure like a tree to model council members and bosses. Teaching–Learning-Based Optimization (TLBO) [40] is another popular HB algorithm inspired by the teaching and learning phenomenon in a classroom. TLBO has two phases named *teacher* and *student*. In the teacher phase, the fittest individual is selected as a teacher and the learning process is done by the teacher. While, in the student phase, all students play the teacher role and learning is carried out by the student interactions. Parliamentary Optimization Algorithm (POA) [41], Ideology Algorithm (IA) [42], Intelligent Clonal Optimizer (ICO) [43], Expectation Algorithm (ExA) [44], and Seasons Optimization (SO) [45] are the well-known HB algorithms.

The question that arises is why additional meta-heuristic optimization algorithms are necessary despite the existence of various ones. In response, it should be noted that technological advancements have unveiled new optimization problems. Based on the No Free Lunch (NFL) theorem [46], existing algorithms may not universally solve these problems. The NFL theorem asserts the absence of a single, all-encompassing meta-heuristic algorithm applicable to every optimization problem. Consequently, this paper introduces a novel socio-inspired meta-heuristic optimization algorithm known as the society deciling process (SDP). SDP emulates the societal deciling process based on factors

such as monthly income, occupation, and education.

To assess and compare the efficacy of SDP against AHA [3], DMO [31], RSA [25], SO [21], PDO [22], FLA [17], and GOA [26], experiments are conducted using 14 general test functions, 9 from CEC 2019, and 12 from CEC 2022. The evaluation criteria include the effectiveness in finding solutions closest to the optimum, early convergence, and hit rate (accuracy).

The key contributions of this paper include the introduction of the novel SDP algorithm, which demonstrates superior convergence speed and hit rate compared to existing algorithms. Our analysis, validated through Friedman's test, establishes the statistical superiority of SDP in terms of finding solutions closest to optima. Additionally, the paper extends the evaluation to three constrained engineering design problems, affirming the algorithm's effectiveness across diverse applications.

SDP proves particularly suitable for addressing optimization challenges in various fields, including but not limited to science, engineering, management, and economics.

Its robust performance, especially in scenarios with intricate and dynamic parameters, positions SDP as a versatile and effective tool for solving real-world optimization problems.

The remaining sections of the paper are outlined as follows. Initially, the inspiration and intricacies of the SDP algorithm will be presented. Following that, the experimental results of SDP across general, CEC 2019, and CEC 2022 test functions will be provided in the Experimental Results section. Subsequently, a statistical analysis of the results is conducted using the Friedman test. Moreover, the proposed algorithm (SDP) on five constrained engineering design problems is evaluated in this section. Finally, the conclusion section concludes the paper, summarizing the findings and offering directions for future research endeavors.

## Society Deciling Process (SDP): The Presented Optimization Algorithm

This section delineates the Society Declining Process (SDP) algorithm, focusing on its inspiration and intricacies.

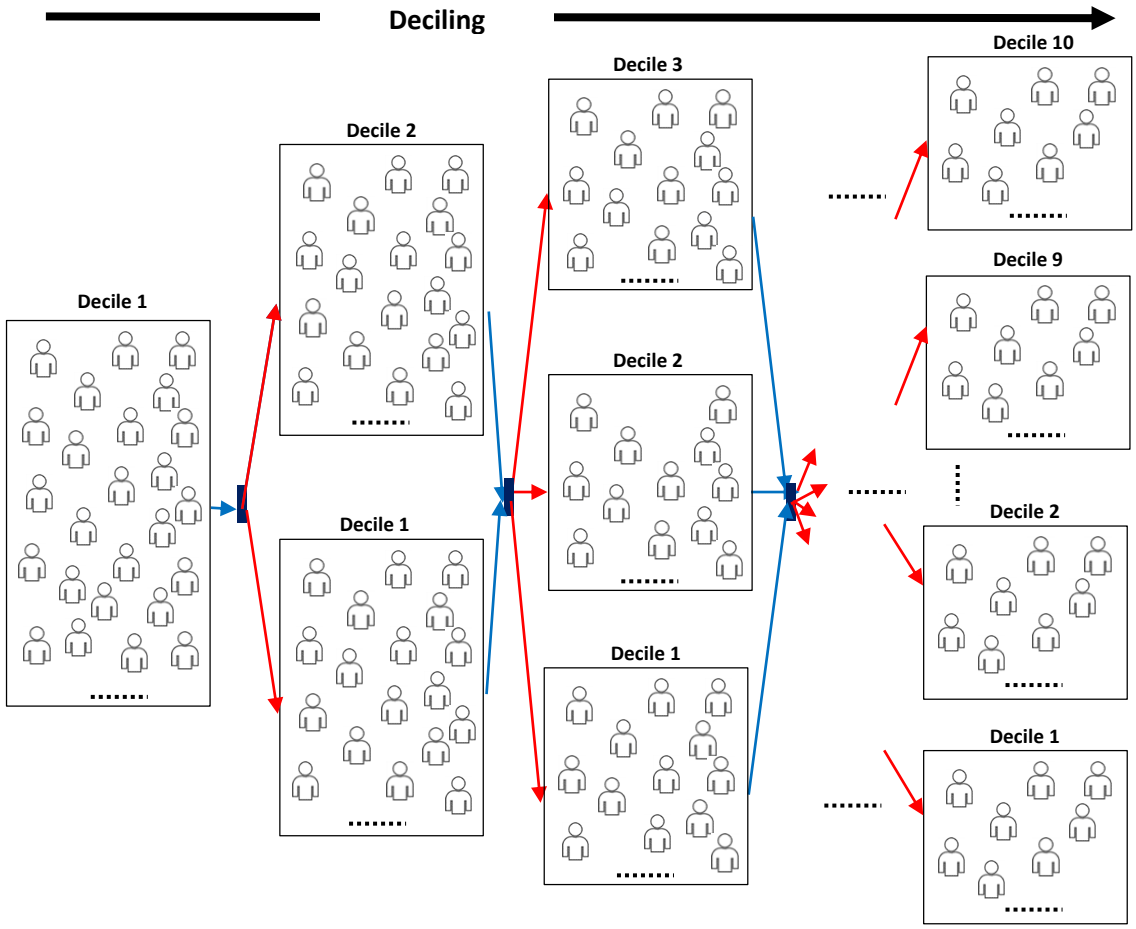


Fig. 1: The deciling process of all people in a society.

### A. Inspiration

Adverse economic conditions, characterized by economic instability, high inflation, reduced purchasing power, and economic discrimination, significantly impact people's living standards, encompassing nutrition, health, and education. Recognizing vulnerable individuals and empowering them is pivotal. To achieve this, the population is categorized into deciles based on criteria like monthly income, occupation, and education. Regular evaluations, considering changes in these criteria, lead to modifications in the society's decile distribution.

Initially, all individuals are placed in the first decile. As some individuals become relatively wealthier, the society divides into two deciles (one and two), with this process continuing until all individuals are distributed across ten deciles. Fig. 1 illustrates the progressive deciling of society.

### B. Details of SDP

In the SDP algorithm, akin to other meta-heuristic methods, the process begins with a randomly generated population X, illustrated in Fig. 2. This population consists of N vectors, each of length m, representing the individuals and their respective variables related to deciling criteria. The variables ($x_{i,j}$) are determined using the (1):

$$x_{i,j} = rand * (u_j - l_j) + l_j \tag{1}$$

Here, *rand* is a random value in the interval [0, 1] where $l_j$ and $u_j$ denote the lower and upper bounds of j-th variable. The fitness of each solution $X_i$ for the test function *f* is computed using the (2):

$$\text{fitness} = f(X_i) = f(x_{i,1}, x_{i,2}, \dots, x_{i,m}) \tag{2}$$

$$X = \begin{pmatrix} X_1 \\ \vdots \\ X_N \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ \vdots & \vdots & & \vdots \\ x_{N,1} & x_{N,2} & \cdots & \end{pmatrix}$$

Fig. 2: Population X with N candidate solutions.

In alignment with the societal deciling process (Fig. 1), the initial placement involves assigning all individuals to the first decile. After altering the average values of deciling criteria for some individuals, the society is iteratively divided into *two* deciles. Consequently, the entire initial population is initially placed in the *first* decile. Following fitness computation, solutions are sorted and distributed into *two* deciles (1 and 2), with the *N/2* solutions possessing the highest fitness placed in decile 2, and the rest in decile 1, establishing a 2-decile state.

In a general *d*-decile state ($2 \le d \le 10$), the fitness-based sorted population *X* is partitioned into *d* subsets of size *decN* (= *N/d*). These subsets are then allocated to the respective deciles (*d, d-1, ..., 1*). Fig. 3 visually represents the population *X* in a *d*-decile state. Denoting the maximum iteration number of the SDP algorithm as *MaxIter*, the number of iterations for each d-decile state ($2 \le d \le 10$) is *dIter* (= *MaxIter/9*). During each *d*-decile state iteration, the SDP algorithm follows these steps *dIter* times:

**Step 1**: Evaluate the population *X* using the test function *f* and sort *X* based on fitness values.

**Step 2**: Divide *X* into *d* partitions, each with the size of *decN*.

**Step 3**: For each deciling criterion, reposition all solutions in decile *k* (*k = d, d-1, ..., 1*) based on higher deciles *j* ($k \le j \le d$) using the *Repose* function. Algorithm 1 details the pseudo-code of the *Repose* function, which adjusts the position of a solution *P* (in decile *k*) based on the fittest solution in the same decile ($B_k$) and those in higher deciles ($B_j$). The weighted average (*wAvg*) of $B_j$ in higher deciles is calculated using (3), emphasizing the importance of higher deciles. The final position is determined by combining the average of *wAvg* and $B_k$ using a modified arithmetic crossover.

$$\boldsymbol{wAvg} = \frac{\sum_{j=k+1}^{d} (\boldsymbol{j} - \boldsymbol{k}) * \boldsymbol{Bj}}{\sum_{j=k+1}^{d} (\boldsymbol{j} - \boldsymbol{k})} \tag{3}$$

Algorithm 2 outlines the steps of the Society Declining Process (SDP) algorithm. In Line 2, a population *X* comprising candidate solutions is generated randomly. The fitness values of these candidates are then computed using the *calcFitness* function, and the entire population sort is accomplished according to these fitness values.

In Line 9, the algorithm proceeds to compute the first (*sIndex*) and last (*eIndex*) indices of the current decile. Subsequently, in Lines 10-13, the positions of all solutions within the specified range ($sIndex \le j \le eIndex$) are adjusted using the *Repose* function. Importantly, if the result of the *Repose* operation yields a solution with a higher fitness value, it replaces the original solution in the population *X*.

To better understand this algorithm, nested iteration loops are explained in more detail: 1) The first *for* loop in line 4: in this loop, the variable *d* controls the index of the *d*-decile state, which starts from 2 and ends in 10. 2) The second *for* loop in line 6: in this loop, the variable *iter* controls the number of algorithm repetition for the *d*-decile state, which starts from 1 and continues to *diter*

(=*MaxIter*/9). 3) The third *for* loop in line 7: in this loop, the variable *cr* controls the number of deciling criteria (the variable number of test functions), which starts from 1 and continues to *m*. 4) The fourth *for* loop in line 8: in this loop, the variable *k* controls the index of higher deciles, which starts from *d* and continues to 1. 5) The fifth *for* loop in line 10: in this loop, the variable *j* controls the indices of the current decile, which starts from *sIndex* = (*d*-*k*)***decN*+1 and continues to *eIndex* = (*d*-*k*+1)***decN*.

Additionally, Fig. 4 provides a visual representation of the algorithm's flowchart, offering a clear illustration of the sequential steps involved in the SDP algorithm. This visual aid enhances the understanding of the algorithm's execution and aids in visualizing the interplay of operations during each iteration.


Fig. 3: Population X in the d-decile state. Bk (1 ≤ k ≤ d) indicates the fittest solution in decile k.

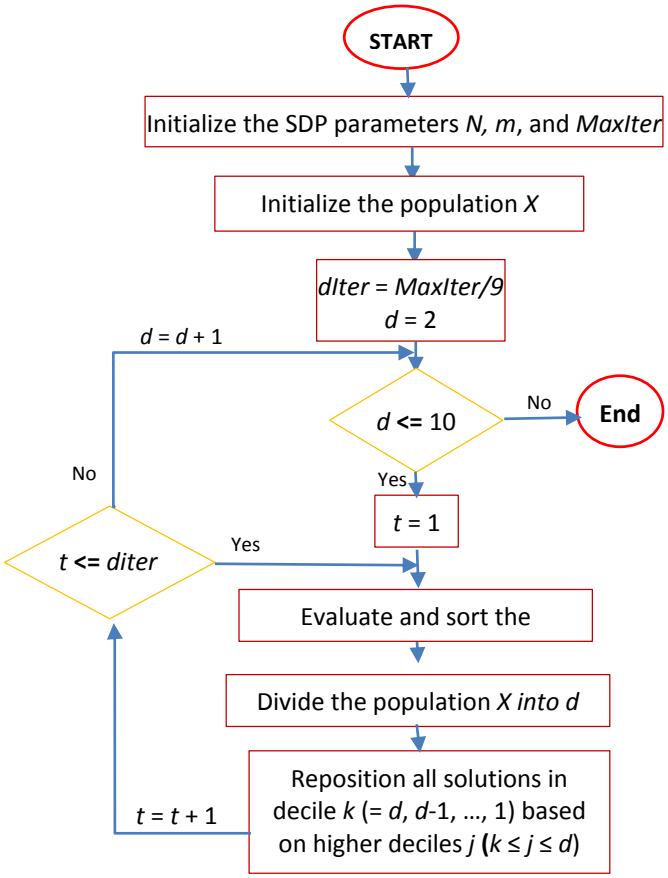


Fig. 4: The flowchart of the SDP algorithm.

**Algorithm 1:** The ***repose*** function

**Input**: *P*: a given solution, *k*,*d*;
**Output:** an individual with the highest fitness.
1: $B_j$ = the fittest solution in decile *j*, ($k \le j \le d$);
2: Compute *wAvg* using Eq. (3);
3: *z* = a value is randomly selected as either 1 or 2;
4: *rnd* = a random value in the interval [0, 1];
5: $\alpha = (-1)^{k+1} * rnd$;
6: Y1 = $\alpha$**P* +(1-$\alpha$)*($B_k$ + *wAvg*)/2; Y2 = (1-$\alpha$)**P* +$\alpha$*($B_k$ + *wAvg*)/2;
7: **return** *max* (Y1, Y2);

**Algorithm 2:** The **SDP** algorithm

**Input**: *N*: the population size, *MaxIter*, *m*: the number of deciling criteria,
*f* : a test function.
**Output**: a solution with the best fitness.
1: Array [][] *X* = new Array [*N*][*m*]; Array [] *fit* = new Array [*N*];
2: *X* = createInitial (*N*, *m*); *fit* = calcFitness (*X, f*); Sort (*X* , *fit*); *best* = *X*[1]; *bestFit* = *fit*[1];
3: *diter* = *MaxIter/9*;
4: **for** *d* =2 to 10 **do**
5: *decN* = *N/d;*
6: **for** *iter* = 1 to *dIter* **do**
7: **for** *cr* = 1 to *m* **do**
8: **for** *k* = *d* downto 1 **do**
9: *sIndex*= (*d*-*k*)**decN*+1; *eIndex*= (*d*-*k*+1)**decN*;
10: **for** *j* = *sIndex* to *eIndex* **do**
11: *P* = *repose* (*X*[*j*], *k*, *d*);
12: **if** *f* (*P*) > *fit* [*j*] **then** M[*j*] =*P*; *fit*[*j*] = *f* (*P*);
13: **end for** *j*
14: **end for** *k*
15: *fit* = calcFitness (*X, f*); Sort (*X* , *fit*);
16: **end for** *cr*
17: **if** *fit* [1]>*bestFit* **then** *best* = *X*[1]; *bestFit* = *fit*[1];
18: **end for** *iter*
19: **end for** *d*
20: **return** best;

## Experimental Results

To evaluate the proposed algorithm, i.e., SDP, four distinct experiment set were conducted. The obtained results were then compared with outcomes generated by seven established meta-heuristic algorithms, namely AHA [3], DMO [31], RSA [25], SO [21], PDO [22], FLA [17], and GOA [26]. The following experiments were carried out for comparison purposes:

**1)** The first experiment applies 14 general test functions to evaluate the exploration and exploitation capabilities of the underlying algorithms. Here, exploitation refers to the algorithm's capacity to enhance the quality of promising solutions through local search, while exploration ability pertains to the algorithm's capability to freely explore different areas of the landscape, thereby avoiding local optima.

**2)** The second experiment utilizes 9 test functions from CEC 2019 to evaluate the efficiency of algorithms in both exploration and exploitation aspects.

**3)** The third experiment encompasses 12 test functions from CEC 2022, serving as a basis for comparing algorithm efficiency in relation to exploration and exploitation.

**4)** The fourth experiment employs five challenging engineering design problems to validate the performance of SDP. The results are then compared with those of various algorithms from the existing literature.

Meta-heuristic algorithms involve various parameters that can be adjusted to suit different optimization problems. These parameters play a crucial role in striking a balance between exploration and exploitation capabilities. This equilibrium allows the algorithm to

navigate diverse regions within the state space. Additionally, when necessary, the algorithm can perform a local search around promising solutions to reach the optimal solution. Hence, determining appropriate values for these parameters is of utmost importance. Notably, algorithms with fewer parameters tend to perform better.

Fortunately, the SDP algorithm does not require dedicated parameter tuning. The parameter settings for SDP and other algorithms are presented in Table 1. To ensure a fair performance comparison among different algorithms, it is essential to set a consistent maximum number of fitness function evaluations (*MNFFE*), considering that the number of evaluations may vary between algorithms. Consequently, the maximum iteration number (*MaxIter*) can be derived from *MNFFE*. Assuming *NFFE* represents the fitness function calling number in each iteration of the algorithm, *MaxIter* can be calculated as *MNFFE*/*NFFE*.

For instance, in the SDP algorithm, *NFFE* is expressed as *2×m×N*, leading to *MaxIter* being *MNFFE*/(*2×m×N*). It's worth noting that the parameters *N* and *MNFFE* are universally set at 30 and 10E+5, respectively, for all algorithms.

The outcomes were derived from the execution of algorithms in the Matlab 2017a environment, performed 30 times using an Intel Core i5 CPU and 6GB RAM. The result tables present the best (*Best*), average (*Ave*), and standard deviation (*Std*) of the solutions identified as the best so far across all runs.

Table 1: Assumed parameter values of all algorithms (*N* = 30 and *MNFFE* = 10E+5)

| Algorithm | Parameters and their appropriate values |
|---|---|
| AHA | *Migration coefficient* = 2×*N* |
| DMO | *Number of babysitters* = 3, *Alpha female vocalization* = 2 |
| RSA | *Alpha* = 0.1, *Beta* = 0.005 |
| SO | *Threshold* = 0.25, *Threshold2* = 0.6, *C1*= 0.5, *C2* = 0.05, *C3* = 2 |
| PDO | *Rho* = 0.005, *epsPD* = 0.1 |
| FLA | *C1* = 0.5, *C2* = 2, *C3* = 0.1, *C4* = 0.2, C5 = 2, *D* = 0.01 |
| GOA | *PSRs*=0.34, *S*=0.88 |

### C. Parameter Analysis

The effectiveness of the SDP algorithm is significantly influenced by certain parameters, such as *N* and *m* (representing the dimension of test functions). To assess this impact, the SDP is executed for various *N* values, considering F1 and F2 from the general test functions, F3 and F4 from the CEC 2019 functions, and F2 and F3 from the CEC 2022 functions. It is worth noting that the maximum number of fitness function evaluations (MNFFE) is set at 30000. As indicated in Table 2, the best effectiveness for SDP is observed when N is equal to 30.

To analyze the effect of *m* on SDP, the algorithm is executed for different values of this parameter across test functions F4, F5, F6, and F7 belonging to CEC 2022. It is important to note that general and CEC 2019 test functions are defined solely with dimension, necessitating the selection of CEC 2022 functions, which are specified for three different dimensions. Table 3 reveals that SDP exhibits optimal effectiveness when m is set to 2 in the majority of the considered test functions.

Furthermore, the table illustrates the impact of increasing the dimension of test functions on the algorithm's execution time. According to the table, as the dimension of the test functions increases, the algorithm's execution time rises due to the escalated number of fitness function evaluations, resulting in a reduction in the number of algorithm repetitions.

### D. Experiment 1: 14 General Test Functions

In this experiment, 14 general test functions are used to analyze and compare SDP and others with regard to exploitation and exploration abilities. Table 4 [47] provides a more detailed descriptions of these functions. Functions F1-F5 are unimodal with single global optima, while functions F6-F9 are multimodal, featuring a global optimum and several local optima. Consequently, finding an optimal solution in test functions such as F6-F9 serves as a robust benchmark for evaluating and comparing the exploration and local optima avoidance capabilities of SDP and other algorithms. Additionally, functions F10-F14 exhibit shifted and rotation attributes, presenting increased complexity for a more precise evaluation of SDP and others. Table 5 presents the obtained results of SDP and other algorithms. Furthermore, the Friedman test is employed to rank the performance of SDP and others—a non-parametric statistical hypothesis test suitable for multiple comparisons of related data sets [48]. Here, first the rank of algorithms in each test

function is determined and then the mean ranks across all test functions are calculated. An algorithm achieving rank 1 will be the best indicating the least mean rank, while the highest rank signifies the worst performance. The results of this test, along with the hit rate based on the reported results in Table 5, are depicted in Fig. 5. As demonstrated in this figure, the first and second ranks are obtained by SO and SDP, respectively. Moreover, the highest hit rate (0.50) after RSA, PDO, and SO is achieved by SDP. In other words, in 7 test functions (i.e., 0.50*14), SDP generates the exact solution.

In addition to utilizing the Friedman test, researchers can employ the Wilcoxon signed-rank test to analyze the obtained results. This non-parametric statistical method, designed for comparing two samples [49], facilitates the determination of cases where Algorithm X outperforms, underperforms, or demonstrates similar performance to Algorithm Y. To fulfill this purpose, three key test statistics, namely $R^-$, $R^+$, and $R^=$, are utilized. Fig. 6 illustrates the Wilcoxon signed-rank test outcomes for the pairwise comparison of SDP versus AHA, DMO, RSA, SO, PDO, FLA, and GOA.

The analysis depicted in Fig. 6 reveals that, in the majority of pairwise comparisons, the values of $R^-$ surpass those of $R^+$. This observation suggests that SDP exhibits superior effectiveness compared to most other algorithms under consideration.

Another criterion that can be employed for comparing the efficiency of our considered algorithms is convergence speed. The optimal solution is reached sooner with a higher convergence speed of an algorithm. To achieve this, test functions F11, F12, F13, and F14, characterized by being shifted and rotated, are considered, and all algorithms are executed up to the number of fitness function evaluations equal to 10E+5. The convergence curve of SDP and others in these test functions is depicted in Fig. 7. In most of these functions, the fastest convergence is observed with SDP.

Table 2: The results of executing SDP for different *N*

| Type | Function | Metric | *N* = 10 | *N* = 20 | *N* = 30 | *N* = 40 | *N* = 50 |
|---|---|---|---|---|---|---|---|
| General | F1 | Best | 3.1824e-43 | 7.0545e-44 | **1.7861e-65** | 1.0526e-32 | 7.8267e-26 |
| | | Ave | 5.522e-43 | 5.7028e-43 | 6.2165e-65 | 2.4259e-32 | 1.719e-25 |
| | | Std | 2.5488e-43 | 8.0896e-43 | 6.5017e-65 | 1.1896e-32 | 8.9457e-26 |
| | F2 | Best | 1.3689e-16 | **6.3971e-33** | 5.6104e-22 | 1.0191e-16 | 1.7863e-13 |
| | | Ave | 1.0695e-15 | 1.0718e-32 | 7.2167e-22 | 1.6129e-16 | 2.1227e-13 |
| | | Std | 1.5509e-15 | 4.3078e-33 | 2.5795e-22 | 5.9981e-17 | 5.3591e-14 |
| CEC 2019 | F3 | Best | **1.27E+01** | **1.27E+01** | **1.27E+01** | **1.27E+01** | **1.27E+01** |
| | | Ave | 1.27E+01 | 1.27E+01 | 1.27E+01 | 1.27E+01 | 1.27E+01 |
| | | Std | 5.7793e-11 | 1.4992e-09 | 2.3484e-09 | 1.4422e-09 | 1.5322e-09 |
| | F4 | Best | 12.775 | 12.585 | **8.792** | 13.325 | 12.313 |
| | | Ave | 16.369 | 14.728 | 13.841 | 14.703 | 13.57 |
| | | Std | 5.7466 | 14.728 | 6.6563 | 1.299 | 1.0942 |
| CEC 2022 | F2 | Best | 400.01 | **400** | **400** | 400.01 | 400.01 |
| | | Ave | 400.05 | 400.07 | 400.04 | 400.10 | 400.12 |
| | | Std | 0.063384 | 0.090989 | 0.06685 | 0.15129 | 0.15441 |
| | F3 | Best | **600** | **600** | **600** | **600** | **600** |
| | | Ave | 600 | 600 | 600 | 600 | 600 |
| | | Std | 8.0389e-14 | 6.00435e-14 | 0.0 | 0.0 | 0.0 |

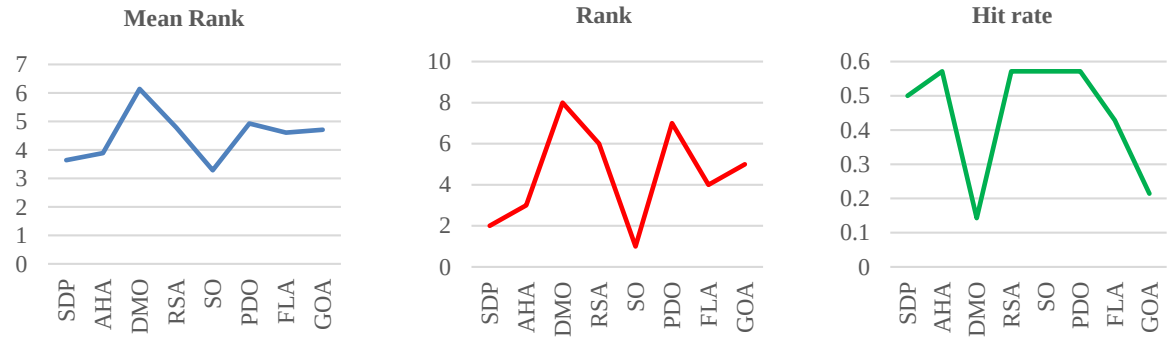


Fig. 5: Results of the Friedman's rank test along with hit rate for general test functions.

Table 3: The results of executing SDP for different *m* (the dimension of test functions)

| Type | Function | Metric | *m* = 2 | *m* = 10 | *m* = 20 |
|---|---|---|---|---|---|
| CEC 2022 | F4 | Best | **800** | 811.96 | 898.77 |
| | | Ave | 800 | 814.3 | 910.4 |
| | | Std | 5.5482e-06 | 4.0129 | 10.674 |
| | | Time | 2.0811 | 1.8952 | 1.7664 |
| | F5 | Best | **900** | 910.74 | 2942.9 |
| | | Ave | 900 | 994.73 | 3458.2 |
| | | Std | 0.0 | 73.803 | 634.74 |
| | | Time | 2.2686 | 1.9223 | 1.9048 |
| | F6 | Best | **1800** | 1851.3 | 4520.3 |
| | | Ave | 1800 | 1994.1 | 6590.2 |
| | | Std | 0.0 | 244.61 | 1927.8 |
| | | Time | 2.1512 | 2.0323 | 2.0964 |
| | F7 | Best | 2463.5 | **2000.5** | 2024.3 |
| | | Ave | 2463.5 | 2001 | 2031.5 |
| | | Std | 0.0 | 0.42429 | 7.7289 |
| | | Time | 3.4253 | 3.0333 | 3.9496 |

Table 4: The general test function details

| Name | Dimension | Range | Optimum value | Type |
|---|---|---|---|---|
| F1 (Sphere) | 30 | [-100, 100] | 0 | Unimodal |
| F2 (Schwefel 2.22) | 30 | [-10, 10] | 0 | |
| F3 (Schwefel 1.2) | 30 | [-100, 100] | 0 | |
| F4 (Rosenbrock) | 30 | [-10, 10] | 0 | |
| F5 (Step) | 30 | [-100, 100] | 0 | |
| F6 (Schwefel's) | 30 | [-500, 500] | 0 | |
| F7 (Rastrigin) | 30 | [-.5.12, 5.12] | 0 | Multimodal |
| F8 (Ackley) | 30 | [-32, 32] | 0 | |
| F9 (Griewank) | 30 | [-600, 600] | 0 | |
| F10 (Shifted Schwefel's P1.2) | 30 | [-100, 100] | -450 | Shifted and Rotated |
| F11 (Shifted Rotated High Conditioned Elliptic) | 30 | [-100, 100] | -450 | |
| F12 (Shifted Rosenbrock's) | 30 | [-100, 100] | 390 | |
| F13 (Shifted Rotated Rastrigin's) | 30 | [-5, 5] | -330 | |
| F14 (Shifted Rotated Weierstrass) | 30 | [-0.5, 0.5] | 90 | |

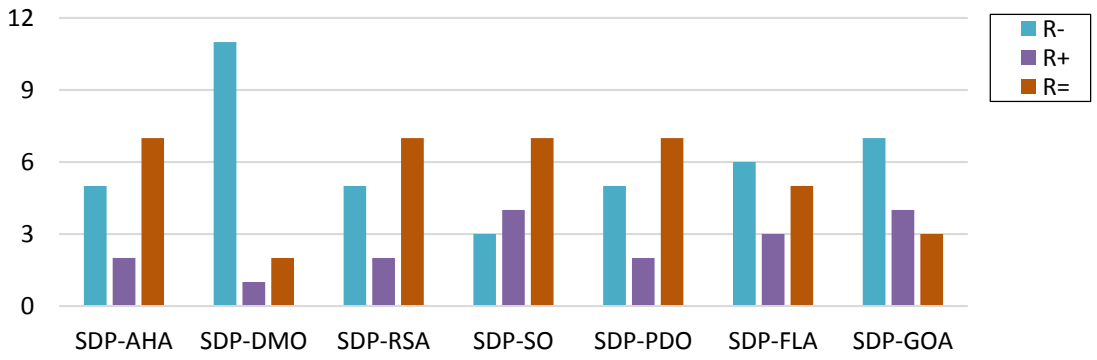


Fig. 6: Results of the Wilcoxon Signed-Rank Test on General Test Functions.

Table 5: Comparative results for general test functions

| Function | Metric | SDP | AHA | DMO | RSA | SO | PDO | FLA | GOA |
|---|---|---|---|---|---|---|---|---|---|
| | Best | **0.00E+00** | **0.00E+00** | 1.89E-24 | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.81E-203 |
| F1 | Ave | 0.00E+00 | 0.00E+00 | 4.20E-24 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 5.89E-101 |
| | Std | 0.00E+00 | 0.00E+00 | 2.15E-24 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 1.32E-100 |
| | Best | **0.00E+00** | **0.00E+00** | 1.64E-18 | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 6.11E-122 |
| F2 | Ave | 0.00E+00 | 0.00E+00 | 1.51E-17 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 2.02E-43 |
| | Std | 0.00E+00 | 0.00E+00 | 1.29E-17 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 4.52E-43 |
| | Best | **0.00E+00** | **0.00E+00** | 2.07E-22 | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.33E-206 |
| F3 | Ave | 0.00E+00 | 0.00E+00 | 6.09E-22 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 3.95E-119 |
| | Std | 0.00E+00 | 0.00E+00 | 3.66E-22 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 8.83E-119 |
| | Best | 1.15E+01 | 2.35E+01 | 2.14E+01 | **1.06E-28** | 2.29E-01 | 8.14E-02 | 6.82E-01 | 2.23E+01 |
| F4 | Ave | 1.51E+01 | 2.35E+01 | 2.33E+01 | **1.45E+01** | 1.46E+00 | 3.29E+00 | 1.30E+00 | 2.32E+01 |
| | Std | 3.44E+00 | 4.83E-03 | 1.73E+00 | **2.05E+01** | 1.74E+00 | 4.54E+00 | 8.47E-01 | 5.64E-01 |
| | Best | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** |
| F5 | Ave | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| | Best | 8.66E+04 | **0.00E+00** | 3.37E+05 | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | 1.75E-35 |
| F6 | Ave | 1.28E+05 | 0.00E+00 | 5.33E+05 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 9.80E-11 |
| | Std | 3.75E+04 | 0.00E+00 | 1.70E+05 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 2.19E-10 |
| | Best | **0.00E+00** | **0.00E+00** | 6.71E+01 | **0.00E+00** | **0.00E+00** | **0.00E+00** | 9.95E-01 | **0.00E+00** |
| F7 | Ave | 0.00E+00 | 0.00E+00 | 8.97E+01 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 1.35E+00 | 0.00E+00 |
| | Std | 0.00E+00 | 0.00E+00 | 3.23E+01 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 6.15E-01 | 0.00E+00 |
| | Best | **0.00E+00** | **0.00E+00** | 1.77E-12 | **0.00E+00** | **0.00E+00** | **0.00E+00** | 3.55E-15 | 3.55E-15 |
| F8 | Ave | 0.00E+00 | 0.00E+00 | 2.03E-12 | 0.00E+00 | 1.78E-15 | 0.00E+00 | 3.55E-15 | 3.55E-15 |
| | Std | 0.00E+00 | 0.00E+00 | 2.27E-13 | 0.00E+00 | 2.51E-15 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| | Best | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** | **0.00E+00** |
| F9 | Ave | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| | Std | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| | Best | 5.04E+03 | 6.24E+01 | 2.71E+04 | 3.37E+04 | **1.66E+01** | 4.35E+04 | 1.57E+02 | 1.37E+02 |
| F10 | Ave | 7.37E+03 | 1.65E+02 | 3.27E+04 | 3.76E+04 | 1.89E+01 | 6.08E+04 | 1.95E+02 | 4.66E+02 |
| | Std | 3.05E+03 | 1.45E+02 | 5.04E+03 | 5.46E+03 | 3.27E+00 | 2.44E+04 | 3.28E+01 | 2.39E+02 |
| | Best | **1.84E+06** | 2.49E+06 | 1.36E+08 | 2.86E+08 | 4.33E+06 | 4.01E+08 | 9.82E+06 | 8.12E+05 |
| F11 | Ave | 1.07E+07 | 4.11E+06 | 2.25E+08 | 6.51E+08 | 6.42E+06 | 6.27E+08 | 1.36E+07 | 3.58E+06 |
| | Std | 1.61E+06 | 2.29E+06 | 7.85E+07 | 5.17E+08 | 2.95E+06 | 3.19E+08 | 3.44E+06 | 2.19E+06 |
| | Best | **1.16E+01** | 1.98E+02 | 2.49E+01 | 1.37E+10 | 2.03E+02 | 1.01E+10 | 2.40E+02 | 4.26E+02 |
| F12 | Ave | 4.16E+01 | 2.01E+02 | 8.86E+01 | 2.29E+10 | 2.16E+02 | 1.06E+10 | 3.87E+02 | 1.92E+03 |
| | Std | 2.18E+01 | 4.62E+00 | 1.02E+02 | 1.31E+10 | 1.87E+01 | 6.88E+08 | 1.74E+02 | 2.39E+03 |
| | Best | 2.30E+02 | 2.83E+02 | 2.04E+02 | 6.07E+02 | **1.16E+02** | 5.01E+02 | 3.28E+02 | 9.44E+01 |
| F13 | Ave | 2.67E+02 | 4.15E+02 | 2.14E+02 | 6.15E+02 | 1.51E+02 | 6.25E+02 | 4.05E+02 | 1.15E+02 |
| | Std | 3.64E+01 | 1.87E+02 | 1.08E+01 | 1.19E+01 | 5.06E+01 | 1.76E+02 | 9.61E+01 | 1.31E+01 |
| | Best | **1.88E+01** | 3.05E+01 | 3.82E+01 | 4.03E+01 | 2.08E+01 | 4.07E+01 | 3.05E+01 | 1.96E+01 |
| F14 | Ave | 2.70E+01 | 3.13E+01 | 3.98E+01 | 4.12E+01 | 2.67E+01 | 4.07E+01 | 3.23E+01 | 2.37E+01 |
| | Std | 5.67E+00 | 1.18E+00 | 1.46E+00 | 1.32E+00 | 8.29E+00 | 1.09E-01 | 1.53E+00 | 3.72E+00 |

### *E. Experiment 2: CEC 2019 Test Functions*

In this experiment, the performance of SDP is compared with that of the considered algorithms in terms of both exploration and exploitation for solving the test functions designed for CEC 2019 [50]. These functions are of a multimodal type with one global minimum. Furthermore, functions F4 to F10 possess few movement and rotation attributes, whereas functions F1 to F3 have default attributes. The details of these functions, such as name, dimension, range, and optimum value, are presented in Table 5. According to this table, functions F1 to F3 have different dimensions and ranges, whereas F4 to F10 are 10-dimensional and fall within the interval [-100, 100]. Additionally, the optimum value for all functions is set to 1. Some algorithms, such as SDP and AHA, exhibit peculiar behavior when solving the F7 test function, leading to the exclusion of this test function from the test suite. The results of all algorithms on solving CEC 2019 test functions are reported in Table 7.

The statistical comparison of SDP and other algorithms in solving the test functions of CEC 2019 involves the use of the Friedman test. The results of this test, presented in Fig. 8 along with the hit rate in Table 5, confirm that SDP exhibits the least mean rank, signifying superior performance in identifying solutions closest to an optimum.

This observation implies that powerful balancing of exploitation and exploration abilities is achieved by SDP compared to other algorithms. Furthermore, the highest hit rate (0.11) among the considered algorithms is attained by SDP, DMO, and RSA, as depicted in the third diagram of Fig. 8.

Furthermore, alongside the Friedman test, the Wilcoxon signed-rank test highlights that across all paired comparisons, the values of $R^-$ consistently surpass $R^+$, suggesting that SDP exhibits superior effectiveness compared to the majority of other algorithms, as depicted in Fig. 9.

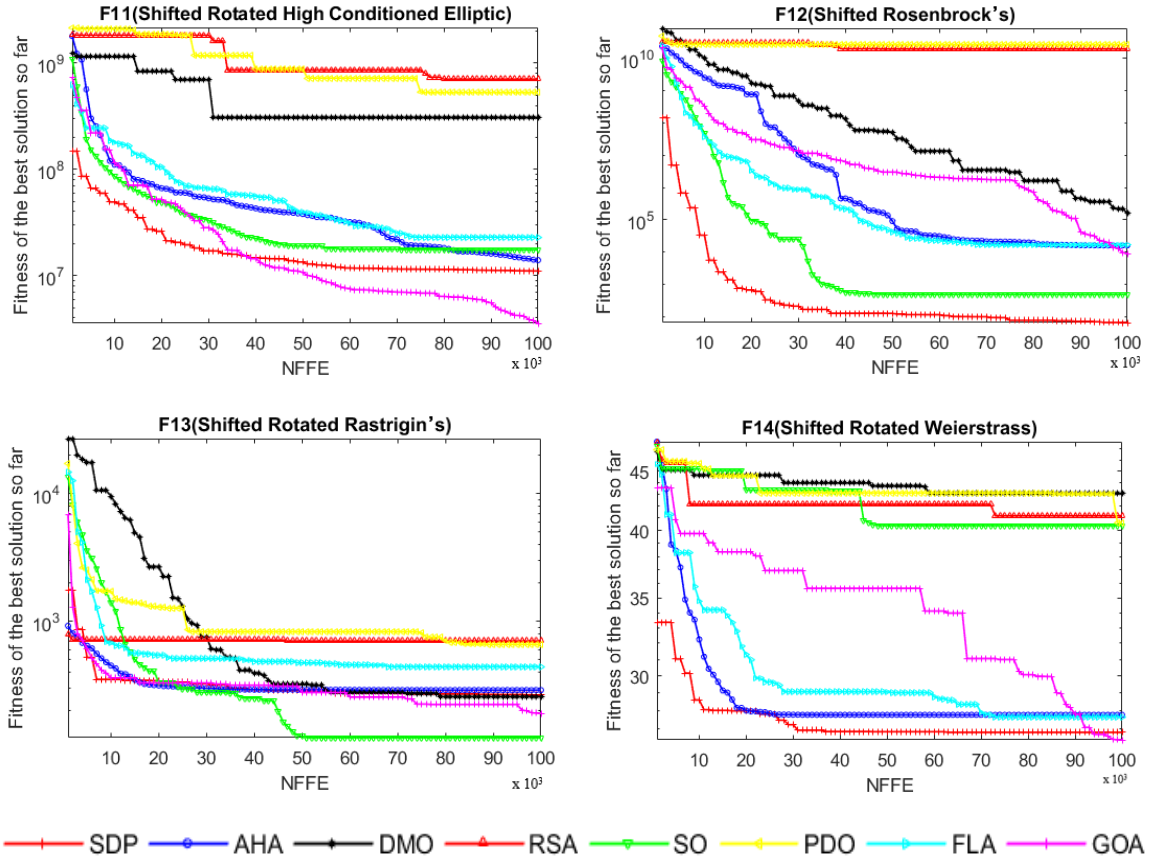


Fig. 7: Convergence curve of all algorithms for general test functions.

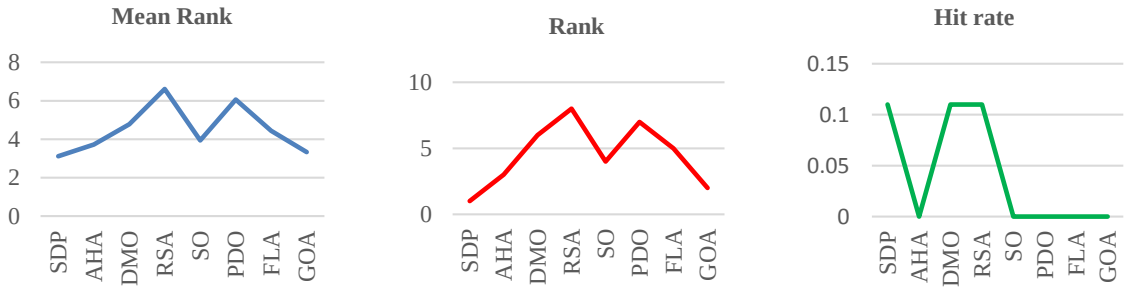


Fig. 8: Results of the Friedman's rank test along with hit rate for CEC 2019 test functions.


Fig. 9: Results of the Wilcoxon signed-rank test to the CEC 2019 test functions.

Table 6: The details of CEC 2019 test functions

| Function | Name | D | Range | Optimum value |
|---|---|---|---|---|
| F1 | Storn's Chebyshev Polynomial Fitting Problem | 9 | [-8192,8192] | 1 |
| F2 | Inverse Hilbert Matrix Problem | 16 | [-16384,16384] | 1 |
| F3 | Lennard-Jones Minimum Energy Cluster | 18 | [-4,4] | 1 |
| F4 | Rastrigin's Function | 10 | [-100,100] | 1 |
| F5 | Griewank's Function | 10 | [-100,100] | 1 |
| F6 | Weierstrass Function | 10 | [-100,100] | 1 |
| F8 | Expanded Schaffer's F6 Function | 10 | [-100,100] | 1 |
| F9 | Happy Cat Function | 10 | [-100,100] | 1 |
| F10 | Ackley Function | 10 | [-100,100] | 1 |

Table 7: Comparative results for CEC 2019 test functions

| Function | Metric | SDP | AHA | DMO | RSA | SO | PDO | FLA | GOA |
|---|---|---|---|---|---|---|---|---|---|
| | Best | 3.23E+10 | 3.74E+04 | 2.34E+09 | 4.36E+04 | 2.58E+05 | 4.62E+04 | 4.62E+04 | **3.14E+04** |
| F1 | Ave | 4.07E+10 | 3.76E+04 | 5.70E+09 | 1.22E+05 | 2.60E+06 | 5.65E+04 | 5.65E+04 | 3.14E+04 |
| | Std | 1.19E+10 | 1.80E+02 | 3.72E+09 | 1.11E+05 | 3.15E+06 | 1.45E+04 | 1.45E+04 | 3.64E-12 |
| | Best | **1.73E+01** | **1.73E+01** | **1.73E+01** | 1.80E+01 | **1.73E+01** | 1.74E+01 | 1.74E+01 | **1.73E+01** |
| F2 | Ave | 1.73E+01 | 1.73E+01 | 1.73E+01 | 1.80E+01 | 1.73E+01 | 1.74E+01 | 1.74E+01 | 1.73E+01 |
| | Std | 0.00E+00 | 4.35E-15 | 0.00E+00 | 1.70E-05 | 0.00E+00 | 1.25E-02 | 6.15E-03 | 0.00E+00 |
| | Best | **1.27E+01** | **1.27E+01** | **1.27E+01** | **1.27E+01** | **1.27E+01** | **1.27E+01** | **1.27E+01** | **1.27E+01** |
| F3 | Ave | 1.27E+01 | 1.27E+01 | 1.27E+01 | 1.27E+01 | 1.27E+01 | 1.27E+01 | 1.27E+01 | 1.27E+01 |
| | Std | 1.04E-09 | 1.26E-15 | 0.00E+00 | 6.93E-06 | 6.86E-14 | 1.35E-07 | 1.76E-08 | 0.00E+00 |
| | Best | **1.02E+01** | 3.68E+01 | 8.45E+00 | 7.86E+03 | 4.98E+00 | 4.34E+03 | 2.40E+01 | 3.62E+01 |
| F4 | Ave | 1.62E+01 | 5.88E+01 | 1.55E+01 | 8.82E+03 | 1.02E+01 | 1.16E+04 | 4.34E+01 | 4.80E+01 |
| | Std | 5.29E+00 | 2.41E+01 | 6.60E+00 | 1.35E+03 | 5.38E+00 | 1.03E+04 | 2.64E+01 | 1.66E+01 |
| | Best | **1.00E+00** | 1.13E+00 | **1.00E+00** | 3.73E+00 | 1.06E+00 | 3.31E+00 | 1.08E+00 | 1.02E+00 |
| F5 | Ave | 1.02E+00 | 1.13E+00 | 1.06E+00 | 3.83E+00 | 1.08E+00 | 3.37E+00 | 1.13E+00 | 1.02E+00 |
| | Std | 9.71E-03 | 6.25E-03 | 5.56E-02 | 1.41E-01 | 1.87E-02 | 8.57E-02 | 4.13E-02 | 3.45E-03 |
| | Best | 3.94E+00 | 3.75E+00 | 9.06E+00 | 9.34E+00 | 8.99E+00 | 8.74E+00 | **3.63E+00** | 5.07E+00 |
| F6 | Ave | 4.42E+00 | 4.63E+00 | 9.58E+00 | 9.82E+00 | 1.00E+01 | 8.88E+00 | 4.27E+00 | 5.60E+00 |
| | Std | 3.80E-01 | 9.61E-01 | 5.75E-01 | 6.81E-01 | 6.77E-01 | 1.87E-01 | 6.28E-01 | 7.50E-01 |
| | Best | 3.01E+00 | 3.44E+00 | 6.44E+00 | 5.51E+00 | **2.89E+00** | 5.21E+00 | 5.04E+00 | 3.05E+00 |
| F8 | Ave | 3.90E+00 | 4.35E+00 | 6.48E+00 | 6.05E+00 | 3.54E+00 | 5.52E+00 | 5.52E+00 | 3.15E+00 |
| | Std | 5.27E-01 | 9.44E-01 | 3.58E-02 | 7.61E-01 | 4.22E-01 | 4.41E-01 | 4.83E-01 | 9.32E-01 |
| | Best | **2.31E+00** | 2.35E+00 | 2.34E+00 | 1.48E+03 | 2.34E+00 | 9.17E+02 | 2.47E+00 | 2.34E+00 |
| F9 | Ave | 2.42E+00 | 2.36E+00 | 2.34E+00 | 1.94E+03 | 2.34E+00 | 9.97E+02 | 2.63E+00 | 2.36E+00 |
| | Std | 9.39E-02 | 1.25E-02 | 2.08E-03 | 6.55E+02 | 2.84E-03 | 1.14E+02 | 1.38E-01 | 1.96E-02 |
| | Best | 1.86E+00 | **2.78E-13** | 2.03E+01 | 2.02E+01 | 2.03E+01 | 2.03E+01 | 2.00E+01 | 2.00E+01 |
| F10 | Ave | 1.64E+01 | 7.05E+00 | 2.03E+01 | 2.04E+01 | 2.04E+01 | 2.03E+01 | 2.00E+01 | 2.00E+01 |
| | Std | 8.12E+00 | 1.12E+01 | 2.97E-02 | 1.81E-01 | 5.38E-02 | 8.96E-02 | 1.76E-03 | 8.07E-03 |

To assess the convergence speed of SDP and other algorithms, they were executed on the test functions of F2 and F3 (including default attributes), as well as F4, F5, F8, and F9 (including a few movement and rotation attributes), up to the number of fitness function evaluations equal to 10E+5. The convergence speed of all algorithms in these test functions is depicted in Fig. 10. In all of these functions, the fastest convergence speed is exhibited by SDP.

*F. Experiment 3: CEC 2022 Test Functions*

In this experiment, the performance of SDP was compared with that of AHA, DMO, RSA, SO, PDO, FLA, and GOA in terms of both exploration and exploitation capabilities on the designed test functions of CEC 2022. The suite used includes 12 functions categorized into different types: *one unimodal* (F1), *four basic* (F2-F5), *three hybrids* (F6-F8), and *four composition functions* (F9-F12). As detailed in Table 8, each function is characterized by similar dimensions, uniformly set at 10, and a consistent range of [-100, 100]. The comparative results for all algorithms on the CEC 2022 test functions are presented in Table 9.

The evaluation of the performance of SDP and others in addressing these test functions involves the utilization of the Friedman test. The results of this test, along with the hit rate from Table 7, are presented in Fig. 11. In the first and second diagrams of this figure, confirmation is provided that the first rank in discovering the closest solutions to optima is achieved by SDP. It is thereby inferred that a powerful balance between exploitation

and exploration abilities is maintained by SDP when compared to other algorithms. In essence, SDP is capable of freely exploring different parts of the search space, if necessary, and enhancing the quality of current promising solutions through local searching around them. Consequently, SDP can effectively avoid local optima.

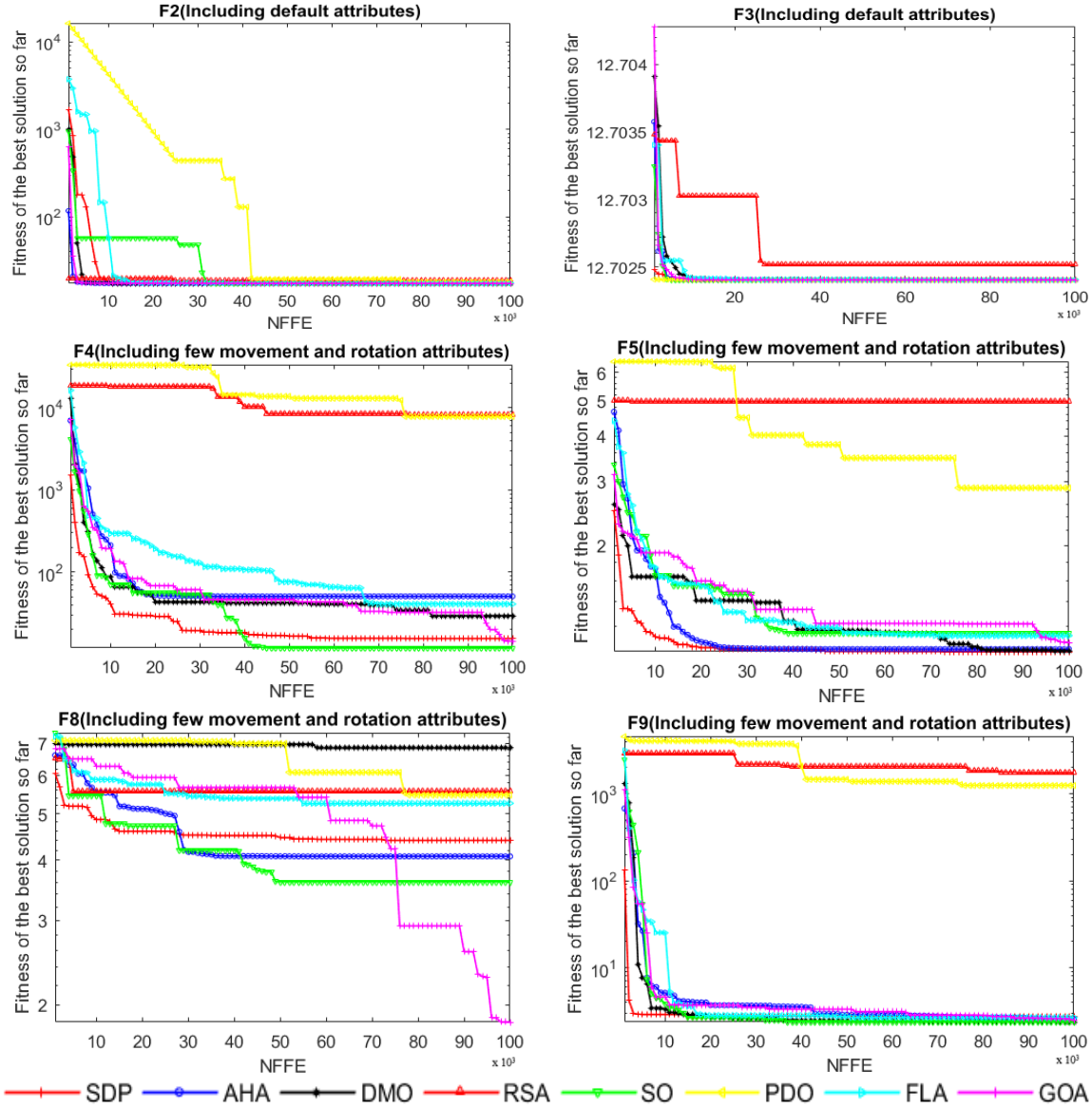


Fig. 10: Convergence curve of all algorithms on solving the CEC 2019 test functions.

Moreover, the third diagram in Fig. 11 illustrates that, following GOA, which boasts the highest hit rate (0.5), SDP, together with AHA and SO, attains the second-highest hit rate (0.41) among the algorithms under consideration. Additionally, beyond the Friedman test, the Wilcoxon signed-rank test results in Fig. 12 indicate that the values of R- surpass R+ in all pair comparisons. It is thereby concluded that SDP exhibits superior effectiveness compared to the majority of other algorithms. Given that the test suite of CEC 2022 comprises four distinct types of test functions, the evaluation and comparison of algorithm convergence speed involve the selection of specific functions of varying types.

The chosen functions include unimodal (F1), basic (F2 and F3), hybrid (F6, F7, and F8), and composition (F9 and F10). The convergence speed of all algorithms in these selected test functions is depicted in Fig. 13. In the majority of these functions, the fastest convergence speed is observed with SDP.

### *G. Experiment 4: Five Constrained Engineering Design Problems*

In this experiment, five constrained engineering design problems are utilized to evaluate the performance of SDP, and the results are compared with various algorithms presented in the literature. Further details regarding these problems, including

mathematical formulas, can be found in [3]. To assess the effectiveness of SDP in optimizing the specified constrained problems, the constraint-handling mechanism introduced in [51] is applied to SDP. In this mechanism, also known as the penalty method, a penalty function is defined to transform the constrained problem into an unconstrained problem. The outcomes from each problem are presented in Tables 10-14, reflecting 30 independent runs for each.

**Speed reducer design**: The objective of this problem is to minimize the weight of a speed reducer. Seven decision variables need to be defined to meet eleven constraints. Table 10 presents the outcomes from SDP and various notable algorithms from the literature, including SC [52], PSO-DE [53], DELC [54], DEDS [55], HEAA [56], MDE [57], and ABC [58]. It is confirmed by this table that the closest solution to the optimal is jointly achieved by SDP, DELC, and DEDS, each attaining the highest rank.

**Pressure vessel design*:*** The goal here is to minimize the fabrication cost of a pressure vessel, involving four design variables and four constraints. The results from SDP and several reputable algorithms like GA2 [59], GA3 [60], QPSO [61], and PSO [53] are shown in Table 11. According to the table, the first rank in achieving the closest solution to the optimal is maintained by SDP.

**Tension/compression spring design*:*** This problem focuses on designing a tension/compression spring with the aim of minimizing its weight, involving four design variables and constraints concerning minimum deflection, shear stress, and surge frequency. Table 12 showcases results from SDP and other recognized algorithms in the literature such as GA2, GA3, CAEP [62], CSPSO [63], HPSO [64], DE [65], SC, and ABC. Based on this table, the closest solution to the optimum is found by SDP, which is ranked first.

**Multiple disc clutch brake (MDCB):** The optimization of a multiple disc clutch brake, aimed at minimizing its overall mass, is the goal of this problem. Five key design variables are manipulated in this optimization process: the inner radius, outer radius, disc thickness, actuating force, and the number of friction surfaces. The results are displayed in Table 13, where the efficacy of SDP in comparison with other notable algorithms such as JAYA [66], TLBO [40], ABC [40], MVO [67], and CMVO [68] is demonstrated. It is confirmed by this table that the closest solution to the optimum is obtained by SDP, which ranks first.

**Welded beam design (WBD):** The development of an economical welded beam design through cost minimization is the objective of this problem, which involves four critical variables: weld thickness, length of the welded segment, beam height, and beam width. The outcomes of various algorithms, including SDP, TEO [69], SCA [52], CDE [70], HAS-GA [71], and CAEP [62], are presented in Table 14. It is shown by this table that SDP achieves the top rank, confirming its effectiveness in securing a solution that closely aligns with the optimum.

Table 8: The details of CEC 2022 test functions (For all test functions: D = 10 and Range = [-100, 100])

| Function | Name | Optimum value | Type |
|---|---|---|---|
| F1 | Shifted and full Rotated Zakharov Function | 300 | Unimodal |
| F2 | Shifted and full Rotated Rosenbrock's Function | 400 | |
| F3 | Shifted and full Rotated Expanded Schaffer's *f*6 Function | 600 | Basic |
| F4 | Shifted and full Rotated Non-Continuous Rastrigin's Function | 800 | |
| F5 | Shifted and full Rotated Levy Function | 900 | |
| F6 | HF 1 (*N* = 3) | 1800 | |
| F7 | HF 2 (N = 6) | 2000 | Hybrid |
| F8 | HF 3 (*N* = 5) | 2200 | |
| F9 | CF 1 (N = 5) | 2300 | |
| F10 | CF 2 (*N* = 4) | 2400 | Composition |
| F11 | CF 3 (N = 5) | 2600 | |
| F12 | CF 4 (N = 6) | 2700 | |

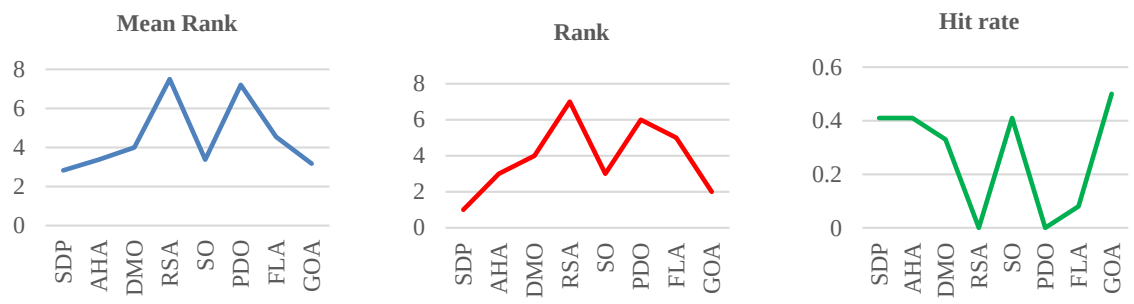


Fig. 11: Results of the Friedman's rank test along with hit rate for CEC 2022 test functions.

Table 9: Comparative results for CEC 2022 test functions

| Function | Metric | SDP | AHA | DMO | RSA | SO | PDO | FLA | GOA |
|---|---|---|---|---|---|---|---|---|---|
| | Best | 2.01E+03 | **3.00E+02** | **3.00E+02** | 5.73E+03 | **3.00E+02** | 1.16E+04 | 3.03E+02 | **3.00E+02** |
| F1 | Ave | 5.58E+03 | 3.00E+02 | 3.00E+02 | 7.13E+03 | 3.00E+02 | 1.53E+04 | 3.09E+02 | 3.00E+02 |
| | Std | 2.91E+03 | 8.04E-14 | 0.00E+00 | 1.97E+03 | 2.84E-14 | 5.12E+03 | 7.73E+00 | 0.00E+00 |
| | Best | **4.00E+02** | **4.00E+02** | 4.01E+02 | 5.32E+02 | **4.00E+02** | 7.75E+02 | 4.07E+02 | **4.00E+02** |
| F2 | Ave | 4.00E+02 | 4.02E+02 | 4.02E+02 | 5.36E+02 | 4.02E+02 | 8.31E+02 | 4.08E+02 | 4.00E+02 |
| | Std | 2.70E-01 | 2.53E+00 | 1.72E+00 | 4.70E+00 | 2.55E+00 | 7.99E+01 | 9.79E-01 | 0.00E+00 |
| | Best | **6.00E+02** | **6.00E+02** | **6.00E+02** | 6.35E+02 | **6.00E+02** | 6.40E+02 | **6.00E+02** | **6.00E+02** |
| F3 | Ave | 6.00E+02 | 6.00E+02 | 6.00E+02 | 6.38E+02 | 6.00E+02 | 6.42E+02 | 6.00E+02 | 6.00E+02 |
| | Std | 8.04E-14 | 5.81E-02 | 0.00E+00 | 4.26E+00 | 6.00E-01 | 3.86E+00 | 3.65E-02 | 1.51E-08 |
| | Best | 8.13E+02 | 8.18E+02 | 8.21E+02 | 8.39E+02 | 8.13E+02 | 8.34E+02 | 8.19E+02 | **8.02E+02** |
| F4 | Ave | 8.18E+02 | 8.23E+02 | 8.25E+02 | 8.40E+02 | 8.14E+02 | 8.36E+02 | 8.33E+02 | 8.05E+02 |
| | Std | 6.62E+00 | 5.01E+00 | 3.20E+00 | 7.46E-01 | 1.30E+00 | 1.91E+00 | 1.28E+01 | 4.22E+00 |
| | Best | 9.37E+02 | 9.01E+02 | **9.00E+02** | 1.22E+03 | **9.00E+02** | 1.23E+03 | 9.01E+02 | **9.00E+02** |
| F5 | Ave | 1.03E+03 | 9.23E+02 | 9.00E+02 | 1.26E+03 | 9.00E+02 | 1.38E+03 | 9.54E+02 | 9.00E+02 |
| | Std | 1.10E+02 | 2.69E+01 | 0.00E+00 | 5.42E+01 | 8.23E-09 | 2.13E+02 | 8.56E+01 | 0.00E+00 |
| | Best | 1.83E+03 | 1.84E+03 | 6.31E+03 | 4.43E+07 | 1.93E+03 | 1.20E+07 | 1.98E+03 | **1.80E+03** |
| F6 | Ave | 1.87E+03 | 1.87E+03 | 8.70E+03 | 4.78E+07 | 2.90E+03 | 5.18E+07 | 3.46E+03 | 1.80E+03 |
| | Std | 3.59E+01 | 2.77E+01 | 2.76E+03 | 4.97E+06 | 1.05E+03 | 5.63E+07 | 1.77E+03 | 7.43E-02 |
| | Best | **2.00E+03** | **2.00E+03** | 2.02E+03 | 2.10E+03 | 2.02E+03 | 2.08E+03 | 2.02E+03 | **2.00E+03** |
| F7 | Ave | 2.00E+03 | 2.01E+03 | 2.02E+03 | 2.10E+03 | 2.03E+03 | 2.08E+03 | 2.02E+03 | 2.00E+03 |
| | Std | 3.11E-01 | 1.36E+01 | 4.36E-01 | 7.51E-01 | 1.45E+01 | 2.47E+00 | 8.47E-02 | 9.44E-03 |
| | Best | **2.21E+03** | 2.22E+03 | **2.21E+03** | 2.24E+03 | 2.22E+03 | 2.23E+03 | 2.22E+03 | 2.22E+03 |
| F8 | Ave | 2.22E+03 | 2.22E+03 | 2.22E+03 | 2.25E+03 | 2.22E+03 | 2.24E+03 | 2.22E+03 | 2.22E+03 |
| | Std | 2.67E+00 | 4.55E-01 | 6.80E+00 | 1.13E+01 | 4.05E-01 | 4.55E+00 | 3.19E-01 | 3.81E-01 |
| | Best | 2.53E+03 | 2.53E+03 | **2.49E+03** | 2.70E+03 | 2.53E+03 | 2.73E+03 | 2.53E+03 | 2.63E+03 |
| F9 | Ave | 2.54E+03 | 2.53E+03 | 2.49E+03 | 2.70E+03 | 2.53E+03 | 2.76E+03 | 2.53E+03 | 2.63E+03 |
| | Std | 4.93E+00 | 5.57E-13 | 0.00E+00 | 2.18E+00 | 2.27E-13 | 3.58E+01 | 1.18E-03 | 0.00E+00 |
| | Best | **2.40E+03** | 2.50E+03 | 2.50E+03 | 2.69E+03 | 2.50E+03 | 2.52E+03 | 2.61E+03 | 2.50E+03 |
| F10 | Ave | 2.42E+03 | 2.54E+03 | 2.51E+03 | 2.70E+03 | 2.61E+03 | 2.52E+03 | 2.62E+03 | 2.50E+03 |
| | Std | 4.39E+01 | 6.17E+01 | 1.75E+01 | 3.86E+00 | 9.97E+01 | 1.59E+00 | 4.05E+00 | 4.29E-02 |
| | Best | **2.60E+03** | **2.60E+03** | **2.60E+03** | 2.90E+03 | **2.60E+03** | 2.84E+03 | 2.60E+03 | **2.60E+03** |
| F11 | Ave | 2.60E+03 | 2.60E+03 | 2.63E+03 | 3.40E+03 | 2.72E+03 | 3.03E+03 | 2.78E+03 | 2.60E+03 |
| | Std | 8.51E-13 | 2.52E-11 | 6.73E+01 | 7.13E+02 | 1.64E+02 | 2.77E+02 | 2.02E+02 | 0.00E+00 |
| | Best | **2.87E+03** | **2.87E+03** | 2.90E+03 | 2.89E+03 | **2.87E+03** | 2.88E+03 | 2.86E+03 | 2.88E+03 |
| F12 | Ave | 2.87E+03 | 2.87E+03 | 2.90E+03 | 2.89E+03 | 2.87E+03 | 2.88E+03 | 2.86E+03 | 2.88E+03 |
| | Std | 1.64E+00 | 5.68E+00 | 1.48E-04 | 5.38E+00 | 2.70E+00 | 1.75E-01 | 2.44E+00 | 2.79E+00 |


Fig. 12: Results of the Wilcoxon signed-rank test for CEC 2022 test functions.

*H. Discussion*

The intense exploitation and exploration capabilities of SDP are evident from the aforementioned four experiments. Its performance in solving unimodal/multimodal functions affirms its exploitation/exploration capability. The competitive edge of SDP over alternative meta-heuristic algorithms is demonstrated by the test outcomes.

Superiority is exhibited by SDP, particularly in effectively addressing high-dimensional functions, surpassing most other algorithms in this context. This result signifies a crucial finding, reinforcing the no-free-lunch theory, as SDP does not excel in solving all 40 functions; in some cases, other algorithms prove more effective than the SDP algorithm.

## Conclusion

This paper introduces the innovative Society Deciling Process (SDP), a meta-heuristic algorithm inspired by the societal practice of categorizing individuals into deciles based on factors like monthly income, occupation, and education. The evaluation of SDP against seven established meta-heuristic algorithms (AHA, DMO, RSA, SO, PDO, FLA, and GOA) using three sets of test functions reveals its superior performance in terms of convergence speed and hit rate. Friedman's rank analysis consistently positions SDP with the best mean rank, indicating its proximity to optimal solutions and heightened exploration and exploitation capabilities compared to other algorithms.

Moreover, the potential expansion of the SDP algorithm to include other quantiles, such as quartiles, is discussed. This expansion introduces flexibility in population division and repositioning, with implications for granularity, sensitivity, computational efficiency, exploration and exploitation balance, adaptability to problem characteristics, convergence speed, and algorithm robustness. The choice of quantiles becomes crucial and should align with the specific properties of the optimization problem at hand.

In summary, the inclusion of other quantiles enhances the adaptability of the SDP approach to different problem characteristics and optimization requirements. The careful consideration of quantile choice is essential for optimizing the algorithm's performance. Additionally, recognizing SDP as a single-objective algorithm prompts future considerations for developing a multi-objective version or exploring its binary iteration. Combining SDP with other optimization techniques also offers promising avenues for future research.

To assess the overall performance of the presented SDP algorithm, a statistical analysis of the experiment results is conducted using the Friedman test. The results of the Friedman test, as depicted in Fig. 14, include the hit rate for all reported outcomes. Due to the test result (i.e., Fig. 14), the least mean rank (i.e., first rank) for finding the closest solutions to an optimum is attributed to SDP. Furthermore, the highest hit rate (0.40) among the considered algorithms is confirmed by the third diagram in this figure, signifying that SDP achieves an optimum solution in 40% of the considered test functions.

Table 10: Results of different methods on the speed reducer problem

| Method | Best | Worst | Ave | Std | Rank (Ave) |
|---|---|---|---|---|---|
| SDP | 2994.471066 | 2994.471066 | 2994.471066 | 0.000000 | **1** |
| SC | 2994.744241 | 3009.964736 | 3001.758264 | 4.0 | **6** |
| PSO-DE | 2996.348167 | 2996.348204 | 2996.348174 | 6.4E-06 | **3** |
| DELC | 2994.471066 | 2994.471066 | 2994.471066 | 1.9E-12 | **1** |
| DEDS | 2994.471066 | 2994.471066 | 2994.471066 | 3.6E-12 | **1** |
| HEAA | 2994.499107 | 2994.752311 | 2994.613368 | 7.0E-02 | **2** |
| MDE | 2996.356689 | NA | 2996.367220 | 8.2E-03 | **4** |
| ABC | 2997.058000 | NA | 2997.058000 | 0.0 | **5** |

Table 11: Results of different methods on the pressure vessel problem

| Method | Best | Worst | Ave | Std | Rank (Ave) |
|---|---|---|---|---|---|
| SDP | 6047.5862 | 6262.3625 | 6156.4525 | 101.4904 | **1** |
| GA2 | 6288.7445 | 6308.4970 | 6293.8432 | 7.4133 | **3** |
| GA3 | 6059.9463 | 6469.3220 | 6177.2533 | 130.9297 | **2** |
| QPSO | 6059.7209 | 8017.2816 | 6440.3786 | 6059.7209 | **4** |
| PSO | 6693.7212 | 14076.3240 | 8756.6803 | 1492.5670 | **5** |

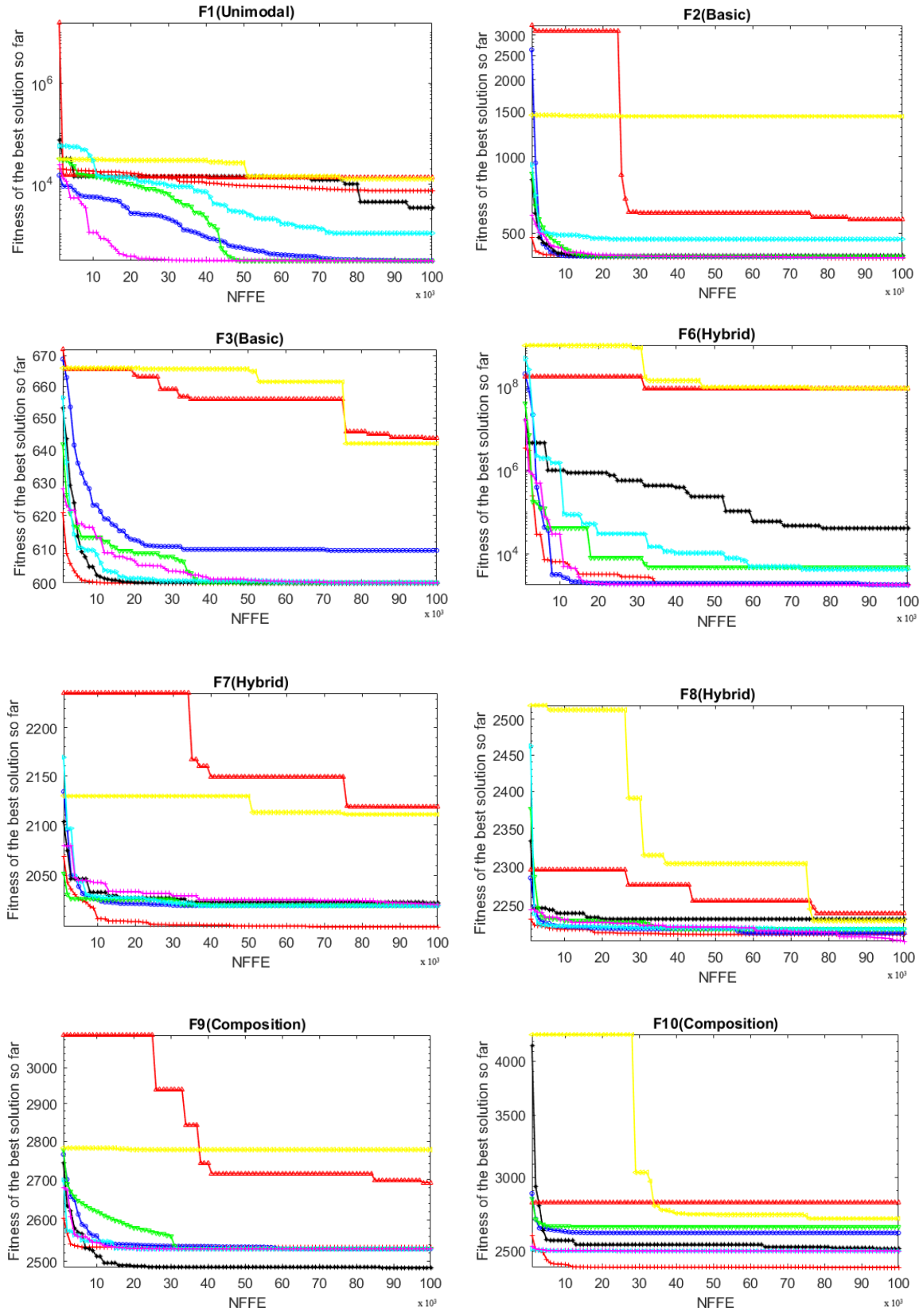


Fig. 13: Convergence curve of all algorithms on solving the CEC 2022 test functions.

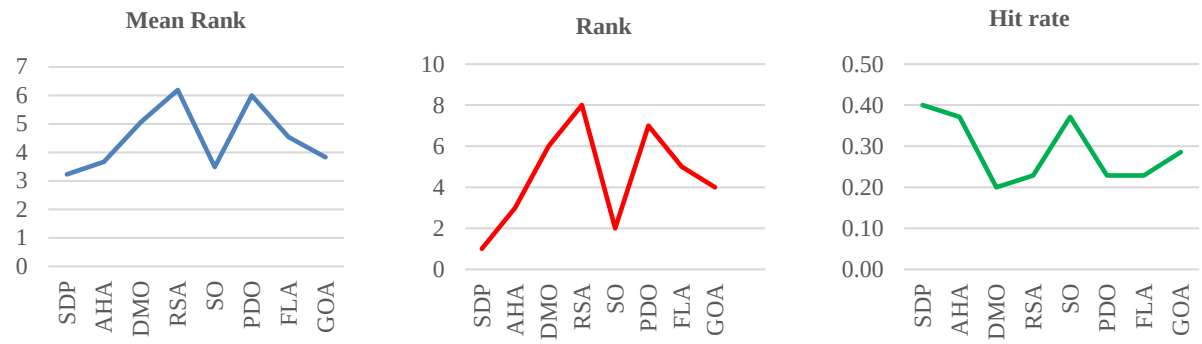


Fig. 14: Results of the Friedman's rank test along with hit rate for all test functions.

Table 12: Comparative analysis of various approaches for tension/compression spring design

| Method | Best | Worst | Ave | Std | Rank (Ave) |
|---|---|---|---|---|---|
| SDP | 0.012668 | 0.012712 | 0.012680 | 0.000018 | **1** |
| GA2 | 0.012704 | 0.012822 | 0.012769 | 3.94E-05 | **7** |
| GA3 | 0.012681 | 0.012973 | 0.012742 | 5.90E-05 | **6** |
| CAEP | 0.012721 | 0.015116 | 0.013568 | 8.42E-04 | **9** |
| CPSO | 0.012674 | 0.012924 | 0.012730 | 5.20E-04 | **5** |
| HPSO | 0.012665 | 0.012719 | 0.012707 | 1.58E-05 | **3** |
| DE | 0.012670 | 0.012790 | 0.012703 | 2.7E-05 | **2** |
| SC | 0.012669 | 0.016717 | 0.012922 | 1.2E-08 | **8** |
| ABC | 0.012665 | NA | 0.012709 | 0.012813 | **4** |

Table 13 Comparative analysis of various approaches to addressing the multiple disc clutch brake issue

| Algorithm | Best | Worst | Ave | Std | Rank(Ave) |
|---|---|---|---|---|---|
| SDP | 0.313657 | 0.42573 | 0.321232 | 0.421 | **1** |
| JAYA | 0.313657 | 0.3867384 | 0.324425 | 0.69 | **3** |
| TLBO | 0.313657 | 0.392071 | 0.3271662 | 0.67 | **5** |
| ABC | 0.313657 | 0.324751 | 0.324751 | 0.54 | **4** |
| MVO | 0.313656 | 0.40120 | 0.351241 | 2.31E-05 | **6** |
| CMVO | 0.313656 | 0.32316 | 0.320104 | 3.22E−07 | **2** |

Table 14 Comparative analysis of various approaches for solving the welded beam design challenge

| Algorithm | Best | Worst | Ave | Std | Rank(Ave) |
|---|---|---|---|---|---|
| SDP | 1.5621 | 1.8394 | 1.7332 | 0.2671 | **1** |
| TEO | 1.7252 | 1.9311 | 1.7680 | 0.0581 | **3** |
| SCA | 2.3854 | 6.3996 | 3.2551 | 0.9590 | **6** |
| CDE | 1.7335 | 1.8241 | 1.7681 | 0.0221 | **4** |
| HAS-GA | 2.2500 | 2.2800 | 2.2600 | 0.0078 | **5** |
| CAEP | 1.7248 | 3.1797 | 1.9718 | 0.4430 | **2** |

## Author Contributions

Both authors of the paper, namely E. Pira and A. Rouhi, developed the metaheuristic algorithm, executed the experiments, analyzed the results, and authored the manuscript.

## Acknowledgment

We would like to express our sincere gratitude to the editor and anonymous reviewers for their valuable time, insightful feedback, and constructive comments, which greatly contributed to the improvement and quality of

this paper.

## Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

## Abbreviations

| | |
|---|---|
| *EA* | Evolutionary Algorithm |
| *GA* | Genetic Algorithm |
| *GP* | Genetic Programming |
| *NP* | Natural Phenomenon |
| *SA* | Simulated Annealing |
| *EVO* | Energy Valley Optimizer |
| *WCA* | Water Cycle Algorithm |
| *NOA* | Nutcracker optimizer |
| *OA* | Orchard Algorithm |
| *SMO* | Swarm Magnetic Optimizer |
| *FuFiO* | Fusion–fission optimization |
| *FLA* | Fick's Law Optimization |
| *SBs* | Social Behaviors |
| *SCO* | Single Candidate Optimizer |
| *GMO* | Geometric Mean Optimizer |
| *HO* | Hippopotamus Optimization |
| *GGO* | Greylag Goose Optimization |
| *LBO* | Ladybug Beetle Optimization |
| *PDO* | Prairie Dog Optimization |
| *AO* | Aquila Optimizer |
| *RFO* | Red Fox Optimization |
| *HBA* | Honey Badger Algorithm |
| *RSA* | Reptile Search Algorithm |
| *IA* | Ideology Algorithm |
| *ICO* | Intelligent Clonal Optimizer |
| *EXA* | Expectation Algorithm |
| *SO* | Seasons Optimization |
| *NFL* | No Free Lunch |
| *SDP* | Society Deciling Process |
| *GOA* | Gazelle Optimization Algorithm |
| *AHA* | Artificial Hummingbird Algorithm |
| *DBO* | Dung Beetle Optimizer |
| *FOX* | Fox optimizer |
| *GTO* | Giant Trevally Optimizer |
| *MGO* | Mountain Gazelle Optimizer |
| *DMO* | Dwarf Mongoose Optimization |
| *ALO* | Attack-Leave Optimizer |
| *PIDC* | Physics-Inspired Discriminative Classifier |
| *BSLO* | Blood-Sucking Leech Optimizer |
| *GKSO* | Genghis Khan shark Optimizar |
| *COA* | Crayfish Optimization Algorithm |
| *MaxIter* | Maximum iteration number |
| *wAvg* | Weighted average |
| *MNFFE* | Maximum number of fitness function evaluations |
| *NFFE* | The number of fitness function calls |
| *Ave* | Average |
| *m* | The dimension of test functions |

## References


[1] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk, W. Al-Atabany, "Honey badger algorithm: New metaheuristic algorithm for solving optimization problems," Math. Comput. Simul., 192: 84-110, 2022.

[2] E. Pira, "City councils evolution: a socio-inspired metaheuristic optimization algorithm," J. Ambient Intell. Hum. Comput., 14(9): 12207-12256, 2023.

[3] W. Zhao, L. Wang, S. Mirjalili, "Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications," Comput. Methods Appl. Mech. Eng., 388: 114194, 2022.

[4] M. Abdel-Basset, L. Abdel-Fatah, A. K. Sangaiah, "Metaheuristic algorithms: A comprehensive review," Comput. Intell. multimedia Big Data Cloud Eng. Appl., 2018: 185-231, 2018.

[5] I. Boussaïd, J. Lepagnot, P. Siarry, "A survey on optimization metaheuristics," Inf. Sci., 237: 82-117, 2013.

[6] J. H. Holland, "Genetic algorithms," Sci. Am., 267(1): 66-73, 1992.

[7] J. R. Koza, "Evolution of subsumption using genetic programming," in Proc. the First European Conference on Artificial Life: 110-119, 1992.

[8] T. M. Shami, D. Grace, A. Burr, P. D. Mitchell, "Single candidate optimizer: a novel optimization algorithm," Evol. Intell., 17(2): 863-887, 2024.

[9] P. D. Kusuma, F. C. Hasibuan, "Attack-Leave optimizer: A new metaheuristic that focuses on the guided search and performs random search as alternative," Int. J. Intell. Eng. Syst., 16(3), 2023.

[10] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, "Optimization by simulated annealing," science, 220(4598): 671-680, 1983.

[11] M. Azizi, U. Aickelin, H. A. Khorshidi, M. Baghalzadeh Shishehgarkhaneh, "Energy valley optimizer: a novel metaheuristic algorithm for global and engineering optimization," Sci. Rep., 13(1): 226, 2023.

[12] H. Eskandar, A. Sadollah, A. Bahreininejad, M. Hamdi, "Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems," Comput. Struct., 110: 151-166, 2012.

[13] M. Abdel-Basset, R. Mohamed, M. Jameel, M. Abouhawwash, "Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems," Knowledge-Based Syst., 262: 110248, 2023.

[14] M. Kaveh, M. S. Mesgari, B. Saeidian, "Orchard Algorithm (OA): A new meta-heuristic algorithm for solving discrete and continuous optimization problems," Math. Comput. Simul., 208: 95-135, 2023.

[15] P. D. Kusuma, F. C. Hasibuan, "Swarm magnetic optimizer: A new optimizer that adopts magnetic behaviour," Int. J. Intell. Eng. Syst., 16(4), 2023.

[16] B. Nouhi, N. Darabi, P. Sareh, H. Bayazidi, F. Darabi, S. Talatahari, "The fusion–fission optimization (FuFiO) algorithm," Sci. Rep., 12(1): 12396, 2022.

[17] F. A. Hashim, R. R. Mostafa, A. G. Hussien, S. Mirjalili, K. M. Sallam, "Fick's law algorithm: A physical law-based algorithm for numerical optimization," Knowledge-Based Syst., 260: 110146, 2023.

[18] F. Rezaei, H. R. Safavi, M. Abd Elaziz, S. Mirjalili, "GMO: geometric mean optimizer for solving engineering problems," Soft Comput., 27(15): 10571-10606, 2023.

[19] M. Monemizadeh, S. R. Samareh Hashemi, M. Sheikh-Hosseini, H. Fehri, "A new physics-inspired discriminative classifier," AUT J. Electr. Eng., 2024.

[20] R. Eberhart, J. Kennedy, "A new optimizer using particle swarm theory," in MHS'95. Proc. the sixth International Symposium on Micro Machine And Human Science: 39-43, 1995.

[21] F. A. Hashim, A. G. Hussien, "Snake optimizer: A novel meta-heuristic optimization algorithm," Knowledge-Based Syst., 242: 108320, 2022.

[22] A. E. Ezugwu, J. O. Agushaka, L. Abualigah, S. Mirjalili, A. H. Gandomi, "Prairie dog optimization algorithm," Neural Comput. Appl., 34(22): 20017-20065, 2022.

[23] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. Al-Qaness, A. H. Gandomi, "Aquila optimizer: a novel meta-heuristic optimization algorithm," Comput. Ind. Eng., 157: 107250, 2021.

[24] D. Połap, M. Woźniak, "Red fox optimization algorithm," Expert Syst. Appl., 166: 114107, 2021.

[25] L. Abualigah, M. Abd Elaziz, P. Sumari, Z. W. Geem, A. H. Gandomi, "Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer," Expert Syst. Appl., 191: 116158, 2022.

[26] J. O. Agushaka, A. E. Ezugwu, L. Abualigah, "Gazelle optimization algorithm: A novel nature-inspired metaheuristic optimizer," Neural Comput. Appl., 35: 4099-4131, 2022.

[27] J. Xue, B. Shen, "Dung beetle optimizer: A new meta-heuristic algorithm for global optimization," J. Supercomput., 79(7): 7305-7336, 2023.

[28] H. Mohammed, T. Rashid, "FOX: a FOX-inspired optimization algorithm," Appl. Intell., 53(1): 1030-1050, 2023.

[29] H. T. Sadeeq, A. M. Abdulazeez, "Giant trevally optimizer (GTO): A novel metaheuristic algorithm for global optimization and challenging engineering problems," IEEE Access, 10: 121615-121640, 2022.

[30] B. Abdollahzadeh, F. S. Gharehchopogh, N. Khodadadi, S. Mirjalili, "Mountain gazelle optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems," Adv. Eng. Software, 174: 103282, 2022.

[31] J. O. Agushaka, A. E. Ezugwu, L. Abualigah, "Dwarf mongoose optimization algorithm," Comput. Methods Appl. Mech. Eng., 391: 114570, 2022.

[32] M. H. Amiri, N. Mehrabi Hashjin, M. Montazeri, S. Mirjalili, N. Khodadadi, "Hippopotamus optimization algorithm: A novel nature-inspired optimization algorithm," Sci. Rep., 14(1): 5032, 2024.

[33] J. Bai, H. Nguyen-Xuan, E. Atroshchenko, G. Kosec, L. Wang, M. A. Wahab, "Blood-sucking leech optimizer," Adv. Eng. Software, 195: 103696, 2024.

[34] E. S. M. El-kenawy, N. Khodadadi, S. Mirjalili, A. A. Abdelhamid, M. M. Eid, A. Ibrahim, "Greylag goose optimization: Nature-inspired optimization algorithm," Expert Syst. Appl., 238: 122147, 2024.

[35] G. Hu, Y. Guo, G. Wei, L. Abualigah, "Genghis Khan shark optimizer: a novel nature-inspired algorithm for engineering optimization," Adv. Eng. Inf., 58: 102210, 2023.

[36] S. Safiri, A. Nikoofard, "Ladybug Beetle optimization algorithm: Application for real-world problems," J. Supercomput., 79(3): 3511-3560, 2023.

[37] H. Jia, H. Rao, C. Wen, S. Mirjalili, "Crayfish optimization algorithm," Artif. Intell. Rev., 56(Suppl 2): 1919-1979, 2023.

[38] S. Satapathy, A. Naik, "Social group optimization (SGO): A new population evolutionary optimization technique," Complex Intell. Syst., 2(3): 173-203, 2016.

[39] E. Pira, "City councils evolution: a socio-inspired metaheuristic optimization algorithm," J. Ambient Intell. Hum. Comput., 14: 12207-12256, 2022.

[40] R. V. Rao, V. J. Savsani, D. Vakharia, "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems," Comput.- Aided Des., 43(3): 303-315, 2011.

[41] A. Borji, "A new global optimization algorithm inspired by parliamentary political competitions," in Proc. Mexican International Conference on Artificial Intelligence: 61-71, 2007.

[42] T. T. Huan, A. J. Kulkarni, J. Kanesan, C. J. Huang, A. Abraham, "Ideology algorithm: A socio-inspired optimization methodology," Neural Comput. Appl., 28(1): 845-876, 2017.

[43] V. Sahargahi, V. Majidnezhad, S. T. Afshord, Y. Jafari, "An intelligent chaotic clonal optimizer," Appl. Soft Comput., 115: 108126, 2022.

[44] A. S. Shastri, A. Jagetia, A. Sehgal, M. Patel, A. J. Kulkarni, "Expectation algorithm (ExA): a socio-inspired optimization methodology," in Socio-cultural Inspired Metaheuristics: Springer, pp. 193-214, 2019.

[45] H. Emami, "Seasons optimization algorithm," Eng. Comput., 38: 1845-1865, 2020.

[46] D. H. Wolpert, W. G. Macready, "No free lunch theorems for optimization," IEEE Trans. Evol. Comput., 1(1): 67-82, 1997.

[47] A. Kaveh, T. Bakhshpoori, "Water evaporation optimization: a novel physically inspired optimization algorithm," Comput. Struct., 167: 69-85, 2016.

[48] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," Ann. Math. Stat., 11(1): 86-92, 1940.

[49] R. F. Woolson, "Wilcoxon signed-rank test," Wiley Encyclopedia of Clinical Trials: 1-3, 2007.

[50] G. Hu, J. Zhong, B. Du, G. Wei, "An enhanced hybrid arithmetic optimization algorithm for engineering applications," Comput. Methods Appl. Mech. Eng., 394: 114901, 2022.

[51] A. H. Gandomi, X. S. Yang, A. H. Alavi, S. Talatahari, "Bat algorithm for constrained optimization tasks," Neural Comput. Appl., 22: 1239-1255, 2013.

[52] T. Ray, K. M. Liew, "Society and civilization: An optimization algorithm based on the simulation of social behavior," IEEE Trans. Evol. Comput., 7(4): 386-396, 2003.

[53] H. Liu, Z. Cai, Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," Appl. Soft Comput., 10(2): 629-640, 2010.

[54] L. Wang, L. p. Li, "An effective differential evolution with level comparison for constrained engineering design," Struct. Multidiscip. Optim., 41(6): 947-963, 2010.

[55] M. Zhang, W. Luo, X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," Inf. Sci., 178(15): 3043-3074, 2008.

[56] Y. Wang, Z. Cai, Y. Zhou, Z. Fan, "Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique," Struct. Multidiscip. Optim., 37(4): 395-413, 2009.

[57] E. Mezura-Montes, C. C. Coello, J. Velázquez-Reyes, "Increasing successful offspring and diversity in differential evolution for engineering design," in Proc. the Seventh international Conference on Adaptive Computing in Design and Manufacture (ACDM 2006): 131-139, 2006.

[58] D. Karaboga, B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," in Proc. International Fuzzy Systems Association World Congress: 789-798, 2007.

[59] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," Comput. Ind., 41(2): 113-127, 2000.

[60] C. A. C. Coello, E. M. Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," Adv. Eng. Inf., 16(3): 193-203, 2002.

[61] L. dos Santos Coelho, "Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems," Expert Syst. Appl., 37(2): 1676-1683, 2010.

[62] C. A. Coello Coello, R. L. Becerra, "Efficient evolutionary optimization through the use of a cultural algorithm," Eng. Optim., 36(2): 219-236, 2004.

[63] Q. He, L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," Eng. Appl. Artif. Intell., 20(1): 89-99, 2007.

[64] Q. He, L. Wang, "A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization," Appl. Math. Comput., 186(2): 1407-1422, 2007.

[65] J. Lampinen, "A constraint handling approach for the differential evolution algorithm," in Proc. the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), 2: 1468-1473, 2002.

[66] R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," Int. J. Ind. Eng. Comput., 7(1): 19-34, 2016.

[67] S. Mirjalili, S. M. Mirjalili, A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," Neural Comput. Appl., 27: 495-513, 2016.

[68] G. I. Sayed, A. Darwish, A. E. Hassanien, "A new chaotic multi-verse optimization algorithm for solving engineering optimization problems," J. Exp. Theor. Artif. Intell., 30(2): 293-317, 2018.

[69] A. Kaveh, A. Dadras, "A novel meta-heuristic optimization algorithm: thermal exchange optimization," Adv. Eng. Software, 110: 69-84, 2017.

[70] F. Z. Huang, L. Wang, Q. He, "An effective co-evolutionary differential evolution for constrained optimization," Appl. Math. Comput., 186(1): 340-356, 2007.

[71] S. Korkmaz, N. B. H. Ali, I. F. Smith, "Configuration of control system for damage tolerance of a tensegrity bridge," Adv. Eng. Inf., 26(1): 145-155, 2012.

## Biographies

**Einollah Pira** received his B.Sc. degree in Computer Engineering (Software) from the University of Kharazmi, Tehran, Iran [1996–2000], the M.Sc. degree in Computer Engineering (Software) from the Sharif University of Technology, Tehran, Iran [2000–2002], and Ph.D. degree in Computer Engineering (Software) from Arak University, Iran [2013-2017]. Currently, he is an Associate Professor with Department of Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran. His research interests include model checking, formal methods, software testing, evolutionary computation, and machine learning.

- Email: pira@azaruniv.ac.ir
- ORCID: 0000-0001-9010-6113
- Web of Science Researcher ID: NA
- Scopus Author ID 55941352000
- Homepage: http://pajouhesh.azaruniv.ac.ir/_Pages/ResearcherEn.aspx?ID=6617

**Alireza Rouhi** received his B.Sc. at Kharazmi University of Tehran in September, 2000; M.Sc. at Sharif University of Technology in June, 2004; and Ph.D. at University of Isfahan in September 2017, all in Software Engineering field. He rewarded as outstanding researcher of Ph.D. students at Faculty of Computer Engineering, University of Isfahan in 2017. Currently, he is an Associate Professor at Azarbaijan Shahid Madani University, Tabriz, Iran. He is interested in Software Engineering in general and Formal Specification, Model Transformation, Metaheuristics, and Social Networks in particular.

- Email: rouhi@azaruniv.ac.ir
- ORCID: 0000-0003-1494-3467
- Web of Science Researcher ID: L-2209-2018
- Scopus Author ID 57189992181
- Homepage: http://pajouhesh.azaruniv.ac.ir/_Pages/ResearcherEn.aspx?ID=5384

**How to cite this paper:**
E. Pira, A. Rouhi, "Society deciling process: A socio-inspired meta-heuristic algorithm," J. Electr. Comput. Eng. Innovations, 12(2): 535-556, 2024.

**DOI:** 10.22061/jecei.2024.10831.740

**URL:** https://jecei.sru.ac.ir/article_2152.html