Studies in Systems, Decision and Control 212

Ali Mohamed
Diego Oliva
Ponnuthurai Nagaratnam Suganthan *Editors*

Handbook of Nature-Inspired Optimization Algorithms: The State of the Art

Volume I: Solving Single Objective Bound-Constrained Real-Parameter Numerical Optimization Problems

Springer

Studies in Systems, Decision and Control

Volume 212

The series "Studies in Systems, Decision and Control" (SSDC) covers both new developments and advances, as well as the state of the art, in the various areas of broadly perceived systems, decision making and control–quickly, up to date and with a high quality. The intent is to cover the theory, applications, and perspectives on the state of the art and future developments relevant to systems, decision making, control, complex processes and related areas, as embedded in the fields of engineering, computer science, physics, economics, social and life sciences, as well as the paradigms and methodologies behind them. The series contains monographs, textbooks, lecture notes and edited volumes in systems, decision making and control spanning the areas of Cyber-Physical Systems, Autonomous Systems, Sensor Networks, Control Systems, Energy Systems, Automotive Systems, Biological Systems, Vehicular Networking and Connected Vehicles, Aerospace Systems, Automation, Manufacturing, Smart Grids, Nonlinear Systems, Power Systems, Robotics, Social Systems, Economic Systems and other. Of particular value to both the contributors and the readership are the short publication timeframe and the worldwide distribution and exposure which enable both a wide and rapid dissemination of research output.

Indexed by SCOPUS, DBLP, WTI Frankfurt eG, zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

Ali Mohamed · Diego Oliva ·
Ponnuthurai Nagaratnam Suganthan
Editors

Handbook of Nature-Inspired Optimization Algorithms: The State of the Art

Volume I: Solving Single Objective Bound-Constrained Real-Parameter Numerical Optimization Problems

Springer

Editors
Ali Mohamed
Operations Research Department, Faculty
of Graduate Studies for Statistical Research
Cairo University
Giza, Egypt

Diego Oliva
Department of Computer Sciences
University of Guadalajara
Guadalajara, Jalisco, Mexico

Ponnuthurai Nagaratnam Suganthan
School of EEE
Nanyang Technological University
Singapore, Singapore

ISSN 2198-4182 ISSN 2198-4190 (electronic)
Studies in Systems, Decision and Control
ISBN 978-3-031-07511-7 ISBN 978-3-031-07512-4 (eBook)
https://doi.org/10.1007/978-3-031-07512-4

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Optimization is defined as the selection of the best elements or actions from a set of feasible alternatives. More precisely, optimization consists of finding the set of variables that produces the best values of objective functions in which the feasible domain of the variables is restricted by constraints.

Meta-heuristic and evolutionary algorithms, many of which are inspired by natural systems, are optimization methods commonly employed to calculate good approximate solutions to optimization problems that are difficult or impossible to solve with other optimization techniques such as linear programming, nonlinear programming, integer programming, and dynamic programming. Meta-heuristic and evolutionary algorithms are problem-independent methods of wide applicability that have been proven effective in solving a wide range of real-world and complex engineering problems. Meta-heuristic and evolutionary algorithms have become popular methods for solving real-world and complex engineering optimization problems.

This book presents various metaheuristic algorithms on single-objective bound-constrained real-parameter numerical optimization problems. We strongly encouraged authors to test the performance of their proposed state-of-the art algorithms on **either** any real-world engineering application such as Electrical and power systems, machine learning, Robotics and Expert Systems, Pattern recognition, Image processing, Bioinformatics and bio-medical engineering, Electronics and communication engineering, and Manufacturing Science **or** one of CEC 2013, CEC2014, or CEC 2017 for bound-constrained benchmarks.

Chapter "Chaotic-SCA Salp Swarm Algorithm Enhanced with Opposition Based Learning: Application to Decrease Carbon Footprint in Patient Flow" develops the Chaotic-SCA Salp Swarm Algorithm which integrates the sine cosine algorithm for updating the position of the leader; the chaotic maps are used for updating the position of followers, and opposition-based learning is used for a better exploration of the search space into the SSA. To show the applicability of the algorithm in real-world problems, then CSOSSA is applied to the green patient flow problem to minimize the total carbon emitted during the care process.

Image segmentation is the most significant pre-processing phase of computer vision. Thresholding is one of the most suitable approaches used for the segmentation

of the image. Searching for an optimal value of the threshold for image segmentation has got more attention in recent years due to its accuracy and robustness. In chapter "Design and Performance Evaluation of Objective Functions Based on Various Measures of Fuzzy Entropies for Image Segmentation Using Grey Wolf Optimization", Grey Wolf Optimization (GWO) algorithm is used along with Shannon and non-Shannon measures of fuzzy entropy for selecting optimal threshold value to segment image in a reasonable amount of time. The objective of the proposed research work is to design objective functions based on Shannon and non-Shannon measures of fuzzy entropy for GWO and use them for optimal threshold value selection which will further be used to segment images.

Optimization is a dynamic process and the parameter setup at the initial stage of the search may not perform well in the later stages. Artificial Bee Colony (ABC) is one of the important algorithms in metaheuristic optimization. In order to contribute to the literature on parameter control in ABC, chapter "Improved Artificial Bee Colony Algorithm with Adaptive Pursuit Based Strategy Selection" presents an Adaptive Pursuit (AP) selection rule-based improved ABC algorithm (IABC-AP) that automatically controls among 9 search strategies while the optimization process is running.

In chapter "Beetle Antennae Search Algorithm for the Motion Planning of Industrial Manipulator", BAS is applied to the redundancy resolution of an industrial manipulator with dynamic joint velocity constraints by searching in high-dimensional space. The addressed application does not need to construct inverse kinematics equations in joint velocity level but directly uses forward kinematics to construct antennae fitness function.

The main objective of an optimal power flow technique is to obtain a steady-state operating point that minimizes the total cost of production and losses and maintains the power system in an acceptable performance in terms of physical limits of electrical network equipment, such as generators, transmission lines, transformers, and compensator shunt. The contribution of chapter "Solving Optimal Power Flow with Considering Placement of TCSC and FACTS Cost Using Cuckoo Search Algorithm" is to present an efficient and reliable nature-inspired Cuckoo Search (CS) algorithm for nonlinear constrained optimization to solve the optimal power flow (OPF) problems with the installation of the FACTS devices, considering TCSC cost.

Increasing energy production from renewable sources is essential to meet climate goals without slowing down economic growth and reducing well-being. Chapter "Parameter Estimation of Per-Unit Photovoltaic Models Using Optimization Algorithms: Comparative Study" evaluates the performance of classical and per-unit mathematical models to estimate the current-voltage characteristic curve describing the behaviour of photovoltaic cells or modules. Concretely, we compare photovoltaic parameter estimation using the classic models commonly used in the literature (single-, double-, and three-diode models) with their per-unit variants.

Chapter "Space-Time Concept in Social Network Search Algorithm" embeds the concept of space-time in the implementation process of the SNS algorithm to develop the Space-Time Social Network Search (STSNS) algorithm. Single-objective, bound-constraint benchmark problems of IEEE congress on evolutionary computation 2014

(CEC 2014) are utilized to study the efficiency of the STSNS algorithm in solving challenging optimization problems.

The golden ratio denoted by the Greek letter *Phi (φ)* represents the irrational number 1.6180339887 approximately. Nowadays, every big organization or company converts its design or logo to a golden ratio design and hires the biggest designers with the highest cost to make it as only the highest professional designers can make it and look good. Chapter "Genetic Algorithm Golden Ratio Design Model for Auto Arts" proposes a framework for generating a golden ratio design using golden circles through a sketched design from the user so that the output design is near to the input design.

The topic of designing linear arrays with design limitations such as the minimum and maximum distance between two adjacent elements is addressed in this book chapter. Chapter "Antenna Array Design Using Differential Evolution with Ranking-Based Mutation Operators" applies a design framework based on Differential Evolution (DE) with ranking-based mutation operators.

Battle Royale Optimizer (BRO) is a Gamed-based metaheuristic optimization algorithm that has been recently proposed for the task of continuous problems. Chapter "Battle Royale Optimizer with a New Movement Strategy" proposes a Modified BRO (M-BRO) in order to improve the balance between exploration and exploitation.

It is worth pointing out that the developments in nature-inspired computing are so rapid that it is estimated that there are more than 150 algorithms and variants in the current literature. Thus, it is not possible and not our intention to review all of them. Instead, we have focused on the diversity and different characteristics of algorithmic structures and their capabilities in solving a wider range of problems in various disciplines.

Giza, Egypt — Ali Mohamed
Guadalajara, Mexico — Diego Oliva
Singapore, Singapore — Ponnuthurai Nagaratnam Suganthan
May 2022

Contents

Chaotic-SCA Salp Swarm Algorithm Enhanced with Opposition Based Learning: Application to Decrease Carbon Footprint in Patient Flow

Masoumeh Vali, Khodakaram Salimifard, Amir H. Gandomi, and Thierry Chaussalet

Abstract Salp swarm algorithm is prone to problems such as a slow convergence rate and local optimal solution. To solve these problems, this chapter proposes the CSOSSA algorithm which integrates the sine cosine algorithm for updating the position of the leader, the chaotic maps are used for updating the position of followers, and opposition-based learning is used for a better exploration of the search space into the SSA. The CSOSSA is compared with the original SSA and other meta-heuristic algorithms on 13 benchmark functions of CEC2005 with unimodal or multimodal characteristics, and five De Jong's functions. The experimental results show that the performance of CSOSSA is better than or comparable with the SSA and other meta-heuristic algorithms. To show the applicability of the algorithm in real-world problems, then CSOSSA is applied to the green patient flow problem to minimize the total carbon emitted during the care process.

Keywords Salp swarm algorithm · Quantum · Chaotic sequence · Patient flow · Opposition-based learning · Sine cosine algorithm · Carbon footprint

M. Vali · K. Salimifard
Department of Industrial Management, Persian Gulf University, 75169 Bushehr, Iran
e-mail: m.vali@mehr.pgu.ac.ir

K. Salimifard
e-mail: salimifard@pgu.ac.ir

A. H. Gandomi (✉)
Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, Australia
e-mail: Amirhossein.Gandomi@uts.edu.au

T. Chaussalet
Computer Science and Engineering Department, University of Westminster, London, UK
e-mail: chausst@westminster.ac.uk

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_1

1 Introduction

Metaheuristic algorithms produce great quality solutions for complex optimization problems in relatively much less time than conventional optimization procedures [1–3]. Many kinds of research use chaotic variables to the meta-heuristic algorithm. Lin et al. [4] proposed a Chaotic Levy flight bat algorithm and described the application of the model estimating the parameter vector of the dynamics of endocytosis. Gholizadeh and Baghchevan [5] presented a chaotic multi-objective firefly algorithm (CMOFA) to implement the performance-based design of steel moment-resisting frame (SMRF) structures. They showed that CMOFA had a better computational performance of the other metaheuristic algorithms.

Yuan et al. [6] proposed a chaotic hybrid differential evolution (DE) algorithm in order to solve short-term hydrothermal scheduling problems. The efficiency of the proposed method is shown by comparing this method with those obtained by augmented Lagrange and two-phase neural network methods based on solution quality. Ahmadi and Mojallali [7] introduced a novel hybrid approach by integrating chaotic with the invasive weed optimization (IWO) method. They examined the execution of the proposed method through a few benchmark multi-dimensional functions and results illustrated superior performance with respect to other conventional strategies.

Arora and Anand [8] introduced a new method with the combination of Grasshopper optimization algorithm (GOA) with chaos theory to balance the reduction in repulsion/attraction forces between grasshoppers and the exploration. The results of chaotic GOA algorithms on thirteen test functions showed that the chaotic maps have a significant role to boost the performance of GOA. In another research, Gao et al. [9] proposed a hybrid method called Chaotic Gravitational search algorithms (CGSAs) to overcome the disadvantage of the GSA like other meta-heuristic algorithms such as slow convergence speed and local optima trapping problems. Experimental results showed that CGSAs have better performances than GSA and other five chaotic particle swarm algorithms.

Gandomi et al. [10] introduced a chaotic accelerated particle swarm optimization (CAPSO) algorithm with integration chaos with accelerated particle swarm optimization (APSO) algorithm to enhance its ability to a global search. They evaluated CAPSO on twelve benchmark problems and three engineering problems to show the efficiency of the CAPSO on complex problems. Gandomi and Yang [11] introduced chaos into the Bat algorithm (BA) to increase its global search mobility for robust global optimization. They utilized thirteen different chaotic maps to validate four variants of them. They showed that some variants of chaotic BAs could outperform the standard BA for proposed benchmarks.

Gandomi et al. [12] introduced chaos into firefly algorithm (FA) called chaotic FA to extend its global search mobility for robust global optimization. They utilized chaotic FA to 12 distinctive chaotic maps to tune the movement of the fireflies in the chaotic FA. The results showed that a few chaotic FAs could outperform the standard FA. Mirjalili and Gandomi [13] introduced chaos-based Gravitational

Search algorithms in terms of exploration and exploitation. The results demonstrated that the sinusoidal map is the best map for improving the performance of GSA. Wang et al. [14] introduced chaos theory into the Krill Herd optimization process to quicken its worldwide convergence speed. The results showed that the performance of Chaotic Krill Herd with an appropriate chaotic map is way better than the KH and other robust optimization methods.

Salp Swarm Algorithm (SSA) is one of the meta-heuristic algorithms as of late proposed by Mirjalili [15]. Zhao et al. [16] proposed a chaotic salp swarm algorithm based on opposition-based learning (OCSSA). They utilized opposition-based learning (OBL) to ensure a better convergence speed and better develops the search space, and also they used the chaotic local search (CLS) method which can progress the performance of the algorithm to obtain the global optimal solution. They showed that execution of the OCSSA, with a suitable chaotic map, was better than or comparable with the SSA and other meta-heuristic algorithms. Table 1 shows a summary of applications of SSA and its hybridization with other algorithms.

There are a number of meta-heuristic algorithms integrated with chaotic but few of them are integrated with chaotic and opposition-based learning. In this chapter, a mathematical model is proposed for global optimization problems. The proposed algorithm is the integration of SSA with chaotic number, sine cosine algorithm and opposition-based learning to enrich the searching behavior of SSA and to avoid being trapped into the local optimum.

The balance between exploitation and exploration is a decisive factor in the performance and efficiency of a meta-heuristic algorithm. Besides opposition-based learning that increases the diversity of the solutions, randomization is a good way to move away from local search and avoids the solutions being trapped at local optima.

The remaining contents of this research are organized as follows. In Sect. 2, the concepts of Salp swarm algorithm, opposition-based learning, chaotic maps, and Sine Cosine Algorithm are described. In Sect. 3, the proposed algorithm is introduced. In Sect. 4, the results of the proposed algorithm on CEC2005 with other algorithms are examined. In Sect. 5, the performance of proposed algorithm is compared with other well-known algorithms on De Jong's Functions. In the Sect. 6, the proposed algorithm is used on a practical problem to minimize carbon footprint in hospital. Finally, the chapter is concluded in the last section.

2 Preliminaries

2.1 Salp Swarm Algorithm

The Salp Swarm Algorithm (SSA) is presented by Mirjalili et al. [15] which is inspired by a kind of a barrel-shaped, planktonic tunicate known as Salp. The SSA mimics the Salp chain behavior to reach food source. The movement and changing the

Table 1 Summary of SSA and its applications

Method	Application
SSA	Engineering design problems [15, 17], blocks on critical path for reentrant job shop scheduling problems [18], optimization of renewable power systems [19], fish image segmentation [20], expansion planning with security constraints [21], multiobjective electric power load dispatch problem [22], extraction of uncertain parameters of single and double diode model of a photovoltaic panel [23], fiber-optic perimeter intrusion detection system [24], redundant container deployment problem [25], facility layout problem [26], a hybrid optimal PID-LQR control of structural system [27], wireless sensor network node location method [28], short-term hydro-thermal-wind scheduling [29], load frequency control of multi-area power system [30], energy management of fuel cell/supercapacitor/batteries in highly fluctuated load condition [31], feature selection problems [32–35], PID controller tuning for Doha reverse osmosis desalination plant [36], three dimensional route planning for multiple unmanned aerial vehicles [37], speed control [38], training neural network [39], variable speed wind generators [40], extracting optimal parameters of PEM fuel cells [41], training feed-forward neural networks [42], load frequency control of multi-area islanded ac micro grid [43], electrical distribution system reconfiguration for power loss reduction [44], extracting optimal parameters of fuel cells [41], to solve optimal power flow comprising voltage stability analysis [45], allocation and capacity of renewable distributed generators on distribution grids [46], parameter estimation for soil water retention curve [47], predicting chemical compound activities [48], optimal allocation of distributed generations and shunt capacitors [49], target localization [50], radial distribution system reconfiguration for real power losses reduction [51], regulation of solar integrated power system [52], structural reliability assessment [53], node localization in wireless sensor networks salp swarm algorithm for node localization in wireless sensor networks predicting pan evaporation in the arid and semi-arid regions of china [54], an image impulsive denoising method [55], economic and emission dispatch problem [56], selective harmonics elimination technique in cascaded h-bridge multi-level inverters [57], optimization of support vector machine parameters [57], multilevel thresholding for color image segmentation [58], numerical and engineering optimization [59], optimal allocation of renewable distributed generation and capacitor banks in distribution systems [60], parameter estimation in an industry 4.0 control systems IoT framework [61], optimization of voltage source inverter's controllers [62], dynamic response and power quality enhancement of an islanded micro grid [35]

(continued)

Table 1 (continued)

Method	Application
SSA with chaos	Feature selection [32, 63], function optimization [64], feature selection for data classification [65, 66]
Opposition based SSA	Numerical optimization [67], feature selection [68], graph coloring problem [69]
Opposition-based chaotic	Global optimization [16]
SSA with metaheuristic algorithms	Feature selection [70], network reconfiguration and capacitor placement for power loss reduction [71], multiobjective big data optimization [72]
Multi-objective SSA(MSSA)	Short-term load forecasting [73], optimum allocation of active and reactive power sources in radial distribution systems [74], feature selection [75]
Hybrid SSA and Fuzzy	A multilevel image thresholding [76]
SSA with sine cosine algorithm	Optimization of nonlinear functions [77]
Quantum SSA	Contrast enhancement of natural images [78], mechanical design [79]
Learning machine model with SSA	Predictive model for hydrological application [80]
Data mining with SSA	Prediction of long-term temperature [81]

position pattern of Salp chain makes it empower to search food using fast harmonious changes [82, 83].

Based on salps' behavior, the population of salps is divided to two groups: a leader and followers. The leader moves at the front of the chain, whereas the rest of salps are considered followers. The position of salps is characterized in an n-dimensional search space where n is the number of variables of a given problem. Consequently, the position of all salps is stored in a two-dimensional matrix called x. Also, there is a food source called F in the search space as the swarm's target. Based on the taking after equation the position of the leader is updated:

$$x_j^1 = \begin{cases} F_j + c_1\big(\big(ub_j - lb_j\big)c_2 + lb_j\big)c_3 \geq 0 \\ F_j - c_1\big(\big(ub_j - lb_j\big)c_2 + lb_j\big)c_3 < 0 \end{cases} \tag{1}$$

where x_j^1 shows the position of the first salp (called leader) in the jth dimension, F_j is the position of the food source in the jth dimension, ub_j determined the upper bound of the jth dimension, lb_j determined the lower bound of the jth dimension, c_1, c_2, and c_3 are random numbers. The coefficient c_1 is the foremost vital parameter in SSA because it balances exploration and exploitation defined as follows:

$$c_1 = 2e^{-\left(\frac{4l}{L}\right)^2} \tag{2}$$

where l is the current iteration and L is the maximum number of iterations. Newton's law of motion is utilized to upgrade the position of the follower as follows:

$$x_j^i = \frac{1}{2}at^2 + v_0 t \tag{3}$$

where $i \geq 2$, x_j^i is the position of the ith follower salp in the jth dimension, t is time, v_0 is the initial speed, and $a = \frac{v_{final}}{v_0}$ where $v = \frac{x - x_0}{t}$. Due to the truth that the time in optimization is iteration, the discrepancy between iterations is equal to 1 and considering $v_0 = 0$, the position of the ith follower salp in the jth dimension is expressed as follows:

$$x_j^i = \frac{1}{2}\left(x_j^i + x_j^{i-1}\right) i \geq 2 \tag{4}$$

The pseudo-code of the SSA is outlined in Algorithm 1 according to the research of Mirjalili et al. [15]. The SSA begins approximating the global optimum by initiating multiple salps with random positions. It calculates the fitness of each salp, finds the salp with the best fitness, and assigns the position of the best salp to the variable F as the source food to be chased by the salp chain. Meanwhile the coefficient c_1 is updated using Eq. (2). For each dimension, the position of leading salp is updated using Eq. (1) and the position of follower salps are updated utilizing Eq. (4). If any of the salps goes outside the search space, the salp will be brought back on the boundaries. All the above steps except initialization are iteratively executed until the satisfaction of an end criterion.

Algorithm 1: Pseudocode of the SSA [34].

Initialize the salp population x_i *(i = 1, 2, ..., n) considering ub and lb*
While*(end condition is not satisfied)*
Calculate the fitness of each search agent(salp)
F=the best search agent
Update c_1 *by Eq. (2)*
  For *each salp* (x_i)
    If *(i==1)*
      Update the position of the leading salp by Eq. (1)
    else
      Update the position of the follower salp by Eq.(4)
    End
  End
  Amend the salps based on the upper and lower bounds of variables
End
Return *F*

2.2 Sine Cosine Algorithm for Updating the Leader's Position

The SCA could be a population-based optimization strategy and a modern meta-heuristic algorithm [84], the solutions are updated based on the sine or cosine function as in Eq. (5).

$$X_i = \begin{cases} X_i + r_1 * \sin(r_2) * |r_3 P_i - X_i| r_4 < 0.5 \\ X_i + r_1 * \cos(r_2) * |r_3 P_i - X_i| r_4 \geq 0.5 \end{cases} \tag{5}$$

where P_i is the destination solution, X_i is the current solution, $|.|$ indicates the absolute value. r_1, r_2, r_3 and r_4 are random variables. The parameter r_1 update using Eq. (6) to balance exploration and exploitation [84].

$$r_1 = a - t\frac{a}{T} \tag{6}$$

where T is the maximum number of iterations, a is a constant and t is the current iteration. The r_2 is a random variable which utilized to discover the direction of the movement of the next solution (i.e. if it towards or outwards P_i). Also, the r_3 is a random variable which gives random weights for P_i to stochastically emphasize ($r_3 > 1$) or deemphasize ($r_3 < 1$) the effect of desalination in defining the distance. The r_4 is utilized to switch between the sine and cosine functions as in Eq. (5).

Algorithm 2. Sine Cosine Algorithm [92].

Initialize *a set of search agents (solutions) (X)*
Repeat
Evaluate each of the search agents by the objective function
Update the best solution obtained so far ($P = X^*$)
Update r_1, r_2, r_3, *and* r_4
Update the position of search agents using Eq. (5)
Until *(* $t <$ *maximum number of iterations)*
Return *the best solution obtained so far as the global optimum*

2.3 Chaotic Maps for Updating the Followers' Position

In this study, chaotic numbers rather than random number generators (RNGs) are utilized to improve the performance of SSA employing a novel approach to upgrading the position of each individual. Within the unique form of SSA, the Newton law has been utilized for upgrading the position of each salp and most SSAs utilize uniform probability distribution to generate random numbers. In many research,

chaotic sequences have been adopted rather than random sequences to obtain good results in many applications.

Utilizing of chaotic sequences as a substitute for random sequences within the proposed algorithm (CSOSSA) may be an effective strategy to diversify the CSOSSA population and improve its performance in preventing premature convergence to local minima. Different chaotic time series sequences such as Lorenz's systems, logistic map, Ikeda map, tent map, Henon map, Chua's system, and Lozi's map are available.

In a few papers, rather than random number generators (RNGs) were utilized chaos methods based on the logistic map to overcome the drawbacks of SSA technique of premature convergence [85–88]. In most of the stochastic search algorithms, RNGs have been broadly utilized. RNGs are known for moderate convergence and have an inherent characteristic of sticking to a solution [89]. To overcome this difficulty, chaotic generators have recently been utilized rather than RNGs. In this study, chaotic maps are utilized to upgrade the followers' position and updating the position of salp to select the best search agent. The chaotic maps (Table 2) are utilized to replace RNGs [90].

Based on Eq. (7), the chaotic maps are used to replace random number generators for updating the followers' position.

$$x_j^i = N(i) * \left(\frac{x_j^i + x_j^{i-1}}{2} \right) i \geq 2 \tag{7}$$

where $N(i+1)$ is obtained using chaotic maps from Table 2.

Table 2 Details of chaotic maps applied on CSOSSA

Map name	Map equation
Logistic map	$N(i+1) = 4N(i)(1 - N(i))$
Cubic map	$Ni + 1 = 2.59\, Ni(1 - Ni2)$
Sine map	$Ni + 1 = \sin(\pi N(i))$
Sinusoidal map	$Ni + 1 = 2.3\ N(i)^2 \sin(\pi N(i))$
Singer map	$N(i+1) = 1.073(7.86\ N(i) - 23.31N(i)^2 + 28.75N(i)^3 - 13.302875N(i)^4)$
Tent map	$N(i+1) = \begin{cases} \frac{N(i)}{0.4}, 0 < N(i) \leq 0.4 \\ \frac{1-N(i)}{0.6}, 0.4 < N(i) \leq 1 \end{cases}$
Gaussian map	$N(i+1) = \begin{cases} 0, N(i) = 0 \\ \left(\frac{1}{N(i)}\right) mod(1), N(i) \neq 0 \end{cases}$
Chebyshev map	$N(i+1) = \cos(0.5\ cos^{-1} N(i))$
Bernoulli map	$N(i+1) = \begin{cases} \frac{N(i)}{0.6}, 0 < N(i) \leq 0.6 \\ \frac{(N(i)-0.6)}{0.4}, 0.6 < N(i) \leq\leq 1 \end{cases}$
Circle map	$N(i+1) = N(i) + 0.5 - \frac{1.1}{\pi} \sin(2\pi N(i))\, mod(1)$

2.4 Opposition Based Learning

The OBL method is implemented to depict the opposite solution to the current solution, and after that evaluate the fitness function to accept or reject the new solution. This method is displayed in Eq. (8).

$$\overline{x_j^i} = U_j + L_j - x_j^i, j = 1, \ldots, dimention \tag{8}$$

In which the $\overline{x_j^i}$ is the opposite vector from the real vector x_j^i. Moreover, within the process of approaching the optimum solution, the two solutions $(\overline{x_j^i}, x_j^i)$ will be compared and the ones which improved the solution will be accepted and the other one will be rejected. The OBL provides a strategy to search for a solution in the opposite direction to the current solution, in this manner, the solution gets to be closer to the optimal solution and the convergence gets to be faster.

3 Proposed Algorithm

In this section, a framework of our proposed algorithm (CSOSSA) is presented. Due to the three previously mentioned parts, the SSA is modified utilizing the sine cosine algorithm for updating the position of salp leader. Moreover, the chaotic maps are used for updating the position of followers, and OBL to reach more exploitation of the search space and guarantees a better convergence speed and better develops the search space. In other words, for escaping the drawbacks of the traditional SSA such as slow convergence, getting stuck in the local optimal solution, and time-consuming, the proposed algorithm is empowered using opposite direction (OBL), chaotic maps, and the Sine Cosine algorithm. These modifications make the SSA faster in comparison with the original and also other versions.

Within the proposed strategy, the salps are initialized using uniform randomization compel by the lower and upper bounds of each dimension. At that point the objective value of each salp is computed to discover the best salp and the best solution P is decided. This best salp's location represents the food location. The position of salps is defined in an n-dimensional search space where n is the number of variables of a given problem. Therefore, the position of all salps is stored in a two-dimensional matrix called x. It is additionally accepted that there is a food source called F in the search space as the swarm's target. To update the position of the leader the Eq. (9) is utilized.

$$X_j^1 = \begin{cases} X_j + r_1 * \sin(r_2) * \left| r_3 F_j - X_j \right| r_4 < 0.5 \\ X_j + r_1 * \cos(r_2) * \left| r_3 F_j - X_j \right| r_4 \geq 0.5 \end{cases} \tag{9}$$

where X_j^1 shows the position of the first salp (leader) in the jth dimension,F_j is the position of the food source in the jth dimension, X_j is the current solution, |.| indicates the absolute value.r_1,r_2, r_3 and r_4 are random variables which r_1 is obtained using Eq. (6) and r_2, r_3 and r_4 are described previously. To update the position of the followers, chaotic maps are utilized. After updating the position of leader and followers of salp swarm utilizing Eqs. (9) and (7), their fitness functions are computed. Also, the chaotic maps and the OBL are utilized to update the position of salps. The proposed OBSCA is outlined in Algorithm 3.

Algorithm 3: Pseudocode of the OBSCA.

Initialize the salp population x_i *(i = 1, 2, ..., n) considering ub and lb*
while*(end condition is not satisfied)*
Calculate the fitness of each search agent(salp)
F=the best search agent
Update r_1 *by Eq. (6)*
For *each salp* (x_i)
If *(i==1)*
Update the position of the leading salp by Eq. (9)
else
Update the position of the follower salp by Eq.(7)
end
end
reposition the salp which go out search space based on lower and upper bounds of problem variables
F=best salp
Using chaotic maps and opposition-based learning if there is a better solution found then update F; otherwise, F left unchanged
End
return

4 Experimental Results of CSOSSA and Discussion

In this section, a set of 13 test functions from the CEC2005 benchmark functions are selected to compare the performance of the CSOSSA with the original SSA and other well-known algorithms. These benchmark functions can be partitioned into unimodal functions (F1–F7) and multimode functions (F8–F13). The details of these test functions are available in Tables 7 and 8 in the Appendix. Two performance indicators are utilized to quantify the performance of the CSOSSA and compared it with other similar algorithms in the literature. The performance metrics utilized are standard deviation and average of the best solutions obtained found in 30 independent runs.

The standard deviation demonstrates how stable CSOSSA is during all the runs and the average is average obtained found in 30 independent runs. For providing a reasonable comparison, the main controlling parameters of these algorithms, maximum iteration, and the number of search agents, are equal to 500 and 30, respectively. The results are presented in Table 3. The comes about on the unimodal functions show that the CSOSSA outperforms other algorithms based on average and standard deviation in the majority of the test functions.

The results on multimodal test functions again show the CSOSSA outperforms other algorithms on most of the test functions. Multi-modal test functions in contrast to unimodal ones have many optima, in which one of them is global and the rest are local. The results of CSOSSA on these cases show that CSOSSA can explore the search space efficiently. The high exploration of CSOSSA causes avoiding the many numbers of local optima in a multi-modal search space.

The standard deviation and average of the best solution obtained during 30 runs testify that CSOSSA shows steady and superior performance on average. Table 3 shows the comparison results of the proposed algorithm with chaotic salp swarm algorithm based on opposition-based learning (OCSSA) and SSA with the particle swarm optimization [16] on CEC2005 benchmark functions.

5 Comparison of CSOSSA with Other Algorithms on De Jong's Functions

In the following subsection, a test suite for comparison, the so-called De Jong's functions is introduced. Then, experimental results are discussed.

5.1 De Jong's Functions

Our approach is examined in De Jong's five-function test suite [91]. We chose these functions because they represent the common difficulties in optimization problems in an isolated manner. Moreover, De Jong's functions are quite popular in the function optimization literature [92–95].

Table 4 presents a survey of De Jong's test suite. The first function of De Jong is one of the simplest test benchmarks. The function is continuous, convex, three-dimensional, and unimodal. Unimodal is a function with one solution. The test area is usually restricted to the polytope $-5.12 < x_i < 5.12$, $i = 1, 2, 3$. Moreover, global minimum $f(x) = 0$ is obtainable for $x_i = 0$.

Known as Rosenbrock's valley or banana function, the second function of De Jong is a classic optimization problem. Although the function is unimodal and two-dimensional, the global optimum lays inside a long, narrow, parabolic-shaped flat valley. Thus, convergence to the global optimum is difficult. The test area is usually

Table 3 Results of unimodal and multi-modal test functions

Algorithm	F1			F2		
	Best	Mean	Std	Best	Mean	Std
SSA	2.95E-09	6.63E-09	1.83E-09	2.91E-04	1.87E-01	2.57E-01
CSOSSA1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**2.75E-242**	**4.48E-218**	**0.00E+00**
CSOSSA2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**6.90E-243**	**2.41E-213**	**0.00E+00**
CSOSSA3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**2.91E-243**	**2.96E-215**	**0.00E+00**
CSOSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**1.80E-302**	**1.03E-289**	**0.00E+00**
CSOSSA5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**3.71E-256**	**7.68E-210**	**0.00E+00**
CSOSSA6	2.05E-43	2.86E-43	7.49E-44	7.36E-23	8.63E-23	1.58E-23
CSOSSA7	1.06E-51	2.42E-46	4.70E-46	7.47E-26	2.11E-22	5.06E-22
CSOSSA8	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.05E-237	2.20E-218	0.00E+00
CSOSSA9	**9.90E-47**	**1.33E-46**	**2.20E-47**	1.16E-24	1.47E-24	2.57E-25
CSOSSA10	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
OCSSA1	4.97E-47	6.10E-42	2.80E-41	9.27E-25	3.14E-09	1.75E-08
OCSSA2	1.12E-52	1.20E-38	6.67E-38	2.08E-27	3.71E-20	2.06E-19
OCSSA3	2.56E-49	1.22E-41	3.79E-41	2.16E-25	2.43E-17	1.35E-16
OCSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**2.01E-230**	**1.57E-179**	**0.00E+00**
OCSSA5	3.09E-116	8.12E-28	4.51E-27	9.18E-55	1.64E-05	9.09E-05
OCSSA6	1.42E-110	6.19E-102	2.12E-101	1.45E-57	1.54E-31	8.57E-31
OCSSA7	2.95E-108	4.48E-94	2.40E-93	9.05E-58	1.18E-51	5.29E-51
OCSSA8	0.00E+00	4.29E-34	2.31E-33	3.96E-197	3.18E-06	1.77E-05
OCSSA9	3.57E-110	6.08E-98	1.83E-97	2.80E-55	1.73E-23	9.64E-23
OCSSA10	1.05E-10	1.24E-09	1.39E-09	6.61E-06	1.27E-04	4.28E-04
SSAPSO	3.38E-31	8.17E-10	6.82E-10	5.02E-31	8.21E-23	2.80E-22
	F3			**F4**		
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**
SSA	1.02E+01	1.09E+02	1.06E+02	1.78E-01	2.80E+00	3.51E+00
CSOSSA1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
CSOSSA2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
CSOSSA3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
CSOSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
CSOSSA5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
CSOSSA6	5.35E-44	2.38E-43	1.53E-43	1.32E-22	1.62E-22	1.96E-23
CSOSSA7	**1.55E-96**	**5.92E-46**	**1.36E-45**	**4.80E-49**	**1.23E-26**	**3.88E-26**
CSOSSA8	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
CSOSSA9	1.57E-47	1.17E-46	1.10E-46	1.87E-24	4.01E-24	1.10E-24
CSOSSA10	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**

(continued)

Table 3 (continued)

Algorithm	F1			F2		
	Best	Mean	Std	Best	Mean	Std
OCSSA1	2.74E-11	5.39E-02	1.55E-01	6.69E-13	3.42E-12	2.80E-12
OCSSA2	8.60E-10	1.13E-02	2.48E-02	2.50E-20	1.82E-18	1.97E-18
OCSSA3	2.03E-20	2.59E-02	7.42E-02	1.79E-13	1.42E-12	1.96E-12
OCSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.75E-235**	**4.21E-208**	**0.00E+00**
OCSSA5	1.25E-09	9.04E-01	2.56E+00	1.20E-63	9.58E-47	2.91E-46
OCSSA6	2.92E-52	6.97E-10	1.90E-09	1.29E-47	4.89E-44	1.40E-43
OCSSA7	7.86E-71	2.20E-10	5.68E-10	9.40E-42	1.45E-37	4.58E-37
OCSSA8	4.49E-241	6.23E-03	1.73E-02	1.41E-233	3.19E-198	0.00E+00
OCSSA9	1.77E-145	3.49E-85	1.00E-84	3.31E-39	9.29E-38	1.09E-37
OCSSA10	1.07E-09	6.21E+00	1.70E+01	5.26E-06	8.12E-06	1.75E-06
SSAPSO	4.43E-48	1.09E-34	4.09E-34	4.17E-25	8.77E-21	3.16E-20
	F5			**F6**		
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**
SSA	1.70E+01	5.55E+01	1.70E+03	3.85E-09	8.30E-09	3.02E-09
CSOSSA1	8.80E+00	8.9373	2.06E-02	4.26E-11	1.01E-11	3.83E-11
CSOSSA2	8.89E+00	8.9373	2.04E-02	3.33E-11	1.08E-11	4.24E-11
CSOSSA3	8.82E+00	8.93921	4.90E-02	5.61E-11	1.03E-11	3.53E-11
CSOSSA4	8.89E+00	8.92136	2.42E-02	2.08E-11	1.06E-11	4.63E-11
CSOSSA5	**8.91E-12**	8.943	**1.93E-12**	4.97E-11	1.13E-11	4.62E-11
CSOSSA6	8.96E-08	**8.94E-10**	1.73E-07	**1.29E-15**	**2.16E-11**	**4.40E-11**
CSOSSA7	8.96E-08	8.99E-07	1.29E-08	1.97E-11	2.28E-11	2.39E-11
CSOSSA8	8.89E-08	8.99E-07	2.98E-10	1.92E-11	6.80E-11	3.08E-11
CSOSSA9	8.95E+00	8.94E-07	1.59E-02	1.31E-11	2.22E-11	4.06E-11
CSOSSA10	8.91E+00	8.99E+00	1.93E-02	4.51E-11	9.26E-12	3.27E-12
OCSSA1	1.86E+01	1.86E+01	3.04E-02	6.21E-10	2.97E-09	1.44E-09
OCSSA2	1.85E+01	1.86E+01	3.71E-02	1.32E-09	4.43E-09	1.63E-09
OCSSA3	1.86E+01	1.86E+01	2.45E-02	3.71E-10	2.20E-09	1.06E-09
OCSSA4	9.41E-09	1.52E+01	6.95E+00	3.26E-14	1.04E-10	1.81E-10
OCSSA5	2.28E-07	1.57E+01	6.34E+00	1.35E-09	5.24E-09	1.75E-09
OCSSA6	1.86E+01	1.86E+01	3.11E-02	3.75E-10	2.24E-09	1.24E-09
OCSSA7	1.86E+01	1.86E+01	3.19E-02	1.12E-11	1.76E-09	1.56E-09
OCSSA8	1.85E-07	1.75E+01	3.34E+00	8.43E-11	4.05E-09	1.87E-09
OCSSA9	1.86E+01	1.87E+01	2.64E-02	8.50E-12	1.20E-09	9.70E-10
OCSSA10	1.49E+01	2.95E+01	2.68E+01	3.82E-09	7.92E-09	2.39E-09

(continued)

Table 3 (continued)

Algorithm	F1			F2		
	Best	Mean	Std	Best	Mean	Std
SSAPSO	7.3333	7.8707	2.62E-01	2.55E-11	6.40E-10	3.76E-10
	F7			**F8**		
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**
SSA	2.53E-02	6.45E-02	2.56E-02	− 6360.384	− 5100.721	537.607
CSOSSA1	1.24E-05	3.25E-05	1.82E-05	− 4189.080	− 4.19E+03	324.1259
CSOSSA2	2.58E-04	2.58E-04	1.11E-04	− 4185.286	− 4.19E+03	181.9815
CSOSSA3	3.57E-05	3.57E-05	1.78E-04	− 4188.789	− 4.17E+03	365.2767
CSOSSA4	3.57E-05	3.57E-05	1.25E-04	− 4182.604	− 4.18E+03	294.1638
CSOSSA5	**2.14E-08**	**2.14E-06**	**1.71E-06**	− 4187.684	− 4.19E+03	530.671
CSOSSA6	1.66E-03	1.66E-03	2.06E-03	**−1356.309**	**−1.3E+03**	**432.9544**
CSOSSA7	1.24E-03	1.24E-03	3.79E-03	− 2139.457	− 1.54E+03	510.4484
CSOSSA8	1.17–05	1.10E-05	1.54E-04	− 4178.275	− 4.18E+03	1108.434
CSOSSA9	2.44E-04	2.44E-04	7.01E-01	− 1592.721	− 1.59E+03	440.7878
CSOSSA10	2.58E-04	2.58E-04	1.38E-04	− 4186.782	− 4.19E+03	403.4007
OCSSA1	2.04E-06	6.28E-04	7.36E-04	− 21,777.57	− 12,304.42	3402.744
OCSSA2	7.00E-06	4.34E-04	6.05E-04	− 17,624.57	− 10,567.80	3418.34
OCSSA3	8.96E-06	3.96E-04	5.62E-04	− 14,301.47	− 12,155.98	2163.005
OCSSA4	7.20E-06	5.18E-04	6.90E-04	− 26,072.40	− 11,567.62	5019.843
OCSSA5	1.46E-07	3.46E-05	2.23E-05	− 20,970.76	− 11,363.73	4055.461
OCSSA6	1.61E-06	6.82E-04	6.87E-04	− 14,301.44	− 11,523.50	2611.771
OCSSA7	9.78E-06	5.75E-04	6.13E-04	− 8379.658	− 8379.658	0
OCSSA8	1.13E-05	4.95E-04	7.89E-04	− 20,473.33	− 11,424.89	4154.301
OCSSA9	5.48E-06	2.43E-04	2.93E-04	− 14,300.68	− 10,097.61	2540.583
OCSSA10	1.91E-05	1.85E-02	1.94E-02	− 16,857.83	− 13,571.51	2249.651
SSAPSO	5.97E-05	9.12E-04	8.48E-04	− 2689.49	−2480.9	1.74E+02
	F9			**F10**		
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**
SSA	5.97E+00	2.92E+01	8.73E+02	1.28E-05	9.69E-01	2.91E+01
CSOSSA1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**0.00E+00**
CSOSSA2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**0.00E+00**
CSOSSA3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**2.06E-31**
CSOSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**0.00E+00**
CSOSSA5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**0.00E+00**
CSOSSA6	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**0.00E+00**

(continued)

Table 3 (continued)

Algorithm	F1			F2		
	Best	Mean	Std	Best	Mean	Std
CSOSSA7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**0.00E+00**
CSOSSA8	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**0.00E+00**
CSOSSA9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**0.00E+00**
CSOSSA10	**0.00E+00**	**0.00E+00**	**0.00E+00**	**8.88E-16**	**8.88E-16**	**0.00E+00**
OCSSA1	0.00E+00	1.34E+00	2.85E+00	8.88E-16	6.44E-15	5.63E-15
OCSSA2	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.88E-16	7.44E-15	5.60E-15
OCSSA3	0.00E+00	2.49E-01	6.34E-01	8.88E-16	4.55E-15	2.59E-15
OCSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.88E-16	1.11E-15	6.38E-16
OCSSA5	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.88E-16	2.81E-08	7.98E-08
OCSSA6	0.00E+00	3.42E-01	8.78E-01	8.88E-16	4.33E-15	2.15E-15
OCSSA7	0.00E+00	4.97E-01	1.43E+00	8.88E-16	2.89E-15	1.76E-15
OCSSA8	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.88E-16	1.55E-15	1.32E-15
OCSSA9	**0.00E+00**	**0.00E+00**	**0.00E+00**	8.88E-16	1.44E-15	1.21E-15
OCSSA10	6.40E-11	8.05E+00	1.46E+01	4.71E-06	1.01E-05	4.55E-06
SSAPSO	**0.00E+00**	1.47E-14	3.43E-14	8.88E-16	1.62E-15	1.46E-15
	F11			**F12**		
	Best	**Mean**	**Std**	**Best**	**Mean**	**Std**
SSA	7.14E-08	2.29E-02	2.11E-02	4.64E-01	3.56E+00	1.87E+00
CSOSSA1	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.46E-12	2.46E-12	9.33E-11
CSOSSA2	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.30E-12	3.30E-12	7.93E-11
CSOSSA3	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.50E-12	4.50E-12	6.65E-11
CSOSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.16E-12	5.16E-12	5.99E-11
CSOSSA5	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.88E-12	2.88E-12	3.78E-11
CSOSSA6	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.97E-13	3.98E-11	8.11E-11
CSOSSA7	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.25E-12	1.25E-11	4.68E-11
CSOSSA8	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.06E-11	1.06E-11	2.02E-11
CSOSSA9	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.33E-12	1.33E-10	4.70E-11
CSOSSA10	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.30E-12	4.30E-12	6.18E-11
OCSSA1	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.06E-11	1.14E-01	1.66E-01
OCSSA2	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.50E-11	1.26E-01	1.90E-01
OCSSA3	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.31E-11	5.90E-02	8.47E-02
OCSSA4	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.34E-14	**1.55E-12**	**1.56E-12**
OCSSA5	**0.00E+00**	**1.08E-09**	**2.84E-09**	8.50E-11	7.26E-02	1.30E-01
OCSSA6	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.65E-12	1.71E-02	4.15E-02

(continued)

Table 3 (continued)

Algorithm	F1			F2		
	Best	Mean	Std	Best	Mean	Std
OCSSA7	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.04E-13	3.11E-11	3.76E-11
OCSSA8	0.00E+00	6.73E-18	1.99E-17	4.60E-11	3.98E-04	9.85E-04
OCSSA9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**3.48E-15**	3.71E-12	3.88E-12
OCSSA10	3.63E-10	1.87E-03	4.39E-03	1.00E-09	6.93E-01	1.14E+00
SSAPSO	**0.00E+00**	3.30E-13	9.35E-13	6.21E-02	1.50E-01	4.26E-02
	F13					
	Best		**Mean**		**Std**	
SSA	1.04E-07		1.05E-01		4.11E-01	
CSOSSA1	4.36E-11		4.36E-11		2.70E-12	
CSOSSA2	4.10E-10		4.10E-07		2.61E-08	
CSOSSA3	2.70E-08		2.70E-06		3.03E-07	
CSOSSA4	5.62E-15		5.62E-08		1.96E-08	
CSOSSA5	2.33E-10		2.33E-06		3.01E-06	
CSOSSA6	7.10E-12		9.45E-12		1.15E-12	
CSOSSA7	9.96E-11		9.96E-11		1.22E-11	
CSOSSA8	2.71 E-11		2.71E-11		2.95E-11	
CSOSSA9	7.94E-11		7.94E-11		8.26E-12	
CSOSSA10	4.12E-11		4.12E-11		2.42E-11	
OCSSA1	1.23E-10		1.28E-09		1.50E-09	
OCSSA2	1.36E-10		1.71E-08		4.73E-08	
OCSSA3	1.15E-11		3.18E-10		2.18E-10	
OCSSA4	**6.03E-17**		3.36E-02		1.60E-01	
OCSSA5	1.67E-10		1.32E-03		7.10E-03	
OCSSA6	1.43E-13		2.48E-10		2.58E-10	
OCSSA7	5.89E-11		3.20E-01		6.54E-01	
OCSSA8	1.51E-11		1.51E-08		3.81E-08	
OCSSA9	3.89E-14		**3.00E-11**		**2.53E-11**	
OCSSA10	6.51E-09		4.78E-03		7.65E-03	
SSAPSO	2.38E-07		1.76E-05		1.50E-05	

restricted to the square $-2.048 < x_i < 2.048$, $i = 1, 2$. Its global minimum is $f(x) = 0$, which is acquired by $x_i = 1$.

The third function of De Jong is a step function, representing the problem of flat surfaces. Flat surfaces are obstacles to optimization algorithms because they give no information as to which direction is favorable. Unless an algorithm has variable step sizes, it can get stuck on one of the flat plateaus. The function is multidimensional and multimodal with an uncertain number of local optimums. The test area is usually

Table 4 De Jong's functions

	Type	Function	Limits	Dim	Number of local minima
F1	Convex function	$\sum_{i=1}^{3} {x_i}^2$	$-5.12 \leq x_i \leq 5.12$	3	1
F2	Banana function	$100.\left({x_1}^2 - x_2\right)^2 + (1 - x_1)^2$	$-2.048 \leq x_i \leq 2.048$	2	1
F3	Step function	$30 + \sum_{i=1}^{5} \lfloor x_j \rfloor$	$-5.12 \leq x_i \leq 5.12$	5	Uncertain
F4	Stochastic function	$\sum_{i=1}^{30} i{x_i}^4 + Gauss(0, 1)$	$-1.28 \leq x_i \leq 1.28$	30	Uncertain
F5	Foxholes function	$\frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{i + \sum_{j=0}^{1} (x_j - a_{ij})^6}}$	$-65.536 \leq x_i \leq 65.536$	2	25

restricted to the polytope $-5.12 < x_i < 5.12$, $i = 1, \ldots, 5$. Its global minimum $f(x) = 0$ is obtainable for $x_i = 0$.

The fourth function of De Jong is a stochastic 30-dimensional function. In this type of function, the algorithm never gets the same value on the same point. Although the function without the added Gaussian noise is a simple unimodal function, the noise makes multimodal function with an uncertain number of local optimums. We limit the test area to the polytope $-1.28 < x_i < 1.28$, $i = 1, \ldots, 30$. However, in the fourth function without the Gaussian noise, the global minimum $f(x) = 0$ is obtainable for $x_i = 0$.

The fifth function of De Jong, also called Foxholes function, is a multimodal test function with 25 local minimum in the square $-65.536 < x_i < 65.536$, $i = 1, 2$. Many standard optimization algorithms get stuck in the first peak they find. Its global optimum is $f(x) = 0$ for $x_i = -32$.

5.2 Algorithms for Comparison

Our algorithm is compared with a conventional genetic algorithm. The GA employs simple, uniform crossover and mutation operators with the probabilities 0.6 and 0.05, respectively. For a fair judgment, the population size is considered to be four, similar to the proposed approach. Besides, our algorithm is compared with the Simulated annealing (SA) proposed by Kirkpatrick and his colleagues [96]. SA is a generic probabilistic metaheuristic for the global optimization problem in a large search space. The algorithm consists of a slow cooling process, which is similar to the mutation process in GA. Eventually, our results are compared with the imperialist competitive algorithm (ICA) [97]. ICA is a computational method utilized to solve

different types of optimization problems. Like GA and most evolutionary computation methods, ICA does not need the gradient of the function in its optimization process. From a specific point of view, ICA can be considered as the social counterpart of genetic algorithms (GAs). ICA is a mathematical model and a computer simulation of human social evolution, while GAs are based on the biological evolution of species.

5.3 Experimental Results on De Jong's Function

To evaluate and to compare the performance of the proposed approach, we have examined it on De Jong's function test suite. Because of the random nature of the algorithms, the results are the average of ten repetitions of each algorithm. The sign "*" in Table 5 indicates that the algorithm is not converged. The results demonstrate that SA is incapable to provide a solution for F3 (step function), F4 (stochastic function), and F5 (multimodal function). Additionally, ICA does not converge in the case of step function. By and large, the results in Table 5 confirm that the proposed approach is significantly better than SSA, ICA, GA, and SA. Comparing CSOSSA with SSA shows that CSOSSA is better than SSA, except for the Banana function F2.

Table 5 Obtained optimum fitness value in optimizing the De Jong's functions

	F1	F2	F3	F4	F5
Global minimum	0.0E+0	0.0E+0	0.0E+0	0+noise	0.0E+0
Genetic algorithm [98]	5.9E-02	8.39E-02	0.0E+0	1.2E-01	9.9E-01
Simulated annealing [96]	9.7E-02	3.41E-01	*	*	*
Imperialist competitive algorithm [97]	5.2E-02	9.56e-02	*	9.1E-01	9.9E-01
SSA [15]	3.5E-12	7.23E-03	0.0E+0	4.1E-01	9.9E-01
CSOSSA1	0.0E+0	6.6E-03	0.0E+0	0.0E+0	8.6E-02
CSOSSA2	0.0E+0	2.09E-03	0.0E+0	0.0E+0	9.9E-02
CSOSSA3	0.0E+0	1.53E-03	0.0E+0	0.0E+0	9.9E-02
CSOSSA4	**0.0E+0**	**1.5E-04**	**0.0E+0**	**0.0E+0**	**5.7E-2**
CSOSSA5	0.0E+0	2.98E-03	0.0E+0	0.0E+0	9.5E-02
CSOSSA6	2.5E-48	5.9E-03	9.9E-03	2.5E-93	9.9E-01
CSOSSA7	4.7E-84	5.75E-03	9.9E-03	6.3E-10	8.2E-01
CSOSSA8	0.0E+0	1.77E-03	0.0E+0	0.0E+0	9.4E-02
CSOSSA9	6.1E-51	3.77E-03	9.9E-03	1.1E-9	9.9E-01
CSOSSA10	0.0E+0	6.65E-03	0.0E+0	0.0E+0	9.1E-02

6 Practical Problem: Carbon Footprint Minimization

In this section, the CSOSSA is utilized to minimize the carbon footprint (CFP) of a hospital using electricity consumption of medical equipment in the patient flow, and after that the performance of CSOSSA is evaluated and compared with SSA.

Our case study is Bushehr Heart Hospital in southern Iran. The hospital has twelve care units including triage, cardiopulmonary resuscitation (CPR), inpatient emergency department (IED), coronary care unit I and II (CCUI and CCU II), post coronary care unit (PCCU), intensive care unit I and II (ICU I and ICU II), Catheterization Laboratory (Cath lab) (angiography, angioplasty and recovery), and operating rooms (ORs). Also, it has two administration units, including reception and discharge units. All the care units have been conceptualized as workstations.

Based on Emergency Severity Index (ESI), patients are categorized into resuscitation (i.e. ESI 1), emergent (i.e. ESI 2), urgent (i.e. ESI 3), less urgent (i.e. ESI 4), and non-urgent (i.e. ESI 5) as a standard classification of patients [99]. A patient in ESI1 category enters to CPR. If treatment is successful, the patient is transferred to inpatient units and then Cath lab. A patient with ESI2 is titled as acute cardiovascular disease (ACS) who has severe pain in the chest, is taken to Cath lab for angiography and if necessary, the Percutaneous Coronary Intervention (PCI) will be applied. A patient having ESI3 is transferred to inpatient units, while an ESI4 patient is admitted to the emergency department for up to six hours' stay and then will be discharged if the treatment is successful. A patient in category ESI5 is referred to an outpatient clinic.

The data covers those patients who have visited the hospital between August 2016 and August 2017. In that period, 7807 patients were referred to ED in total, from which 5% have been diagnosed with ESI 1 and another 5% to ESI 2. The ESI for others was found to be 30%, 30%, and 30% for ESI 3, ESI 4, and ESI 5, respectively. The amount of CO_2 emitted is based on the electricity consumption of equipment which available from its technical specification [100–107].

6.1 Mathematical Formulation to Calculate CFP

The care units that the patient has to go to during his/her treatment processes (based on his/her ESI). According to this table, ESI 1 and ESI 2 patients have three therapeutic pathways (routes) based on physician diagnosis. In general, each patient has to go through a number of care units from the time of admission to discharge. Therefore, in each care unit, there is a set of medical equipment used for the patient who undergoes a series of treatment processes. Each medical equipment consumes electrical power, and with more electrical power be consumed, more CO_2 is produced.

As described before, the hospital has 14 units. In each unit, there is a maximum of 21 medical equipment for patients' treatment. The emission factor (EF) of different fuels is expressed as kilograms of CO_2 per kWh, (or $kgCO_2/kWh$). EF average is

0.61 for the year 2019 in Iran. The amount of produced $kgCO_2$ in the hospital is calculated as follows:

Indices:

i Medical equipment index $(1, 2, \ldots, m)$
j Patient index $(1, 2, \ldots, n)$
k Care and administration units index (1, 2,…, o)

Parameters:

m Total number of medical equipment
n Total number of patients (ENP)

Note:

n_1: ENP of ESI 1 (Route 11), n_2: ENP of ESI 1 (Route 12), n_3: ENP of ESI 1 (Route 13),

n_4: ENP of ESI 2 (Route 21), n_5: ENP of ESI 2 (Route 22), n_6: ENP of ESI 2 (Route 23),

n_7: ENP of ESI 3 (Route 3), n_8: ENP of ESI 4 (Route 4), n_9: ENP of ESI 5 (Route 5)

$$n = n_1 + n_2 + n_3 + n_4 + n_5 + n_6 + n_7 + n_8 + n_9$$

$$\text{s.t } n_i = n * percentage\, n_i$$

$P_{i,j,k}$ Processing time of the ith medical equipment of kth unit on patient j
$Pm_{i,k}$ Power consumption of ith medical equipment of kth unit in the processing condition
EF Quantity of emitted carbon footprint per kilowatt-hour

Decision variables:

Z The total emitted carbon footprint

$$X_{i,j,k} = \begin{cases} 1, \text{ If the } i\text{th medical equipment of } k\text{th unit is executed treatment process on the } j\text{th patient} \\ 0, \ otherwise \end{cases}$$

Objective function:

$$MinZ = \sum_{j=1}^{n} \sum_{k=1}^{o} \sum_{i=1}^{m} \text{EF}.Pm_{i,k}.P_{i,j,k}.X_{i,j,k}$$

6.2 Solution Representation

Each member of the population in the CSOSSA is a chromosome with 14 * 21 = 296 bits where 14 is the number of units and 21 is the maximum number of medical equipment in each unit. The initial value of each bit for population members is a real number greater than or equal to zero. Each bit represents the usage time of medical equipment i in care unit k. That is, the chromosome is divided into fourteen 21-bit units as follows.

The upper and lower bound for each chromosome corresponds to the maximum and minimum time of use of the medical equipment. Therefore, the minimization problem of CFP is converted as follows:

$$\min T(X)$$

$$lb \leq X \leq ub$$

where lb and ub represent the upper and lower bound of the decision variable.

6.3 Experimental Results of CFP Minimization

The SSA and CSOSSA are run ten times for CFP minimization problem. Table 6 compares the performance of the original SSA with the CSOSSA in terms of mean and standard deviation based on the average produced CO_2 in the treatment process of each hospitalized patient of different ESI. Depending on the patient conditions and treatment requirements, patients of ESI 1 and ESI 2 have three possible routes. The

Table 6 Average $kgCO_2$ produced based on ESI

Algorithm	Route	SSA		CSOSSA	
Patient category		Mean	STD	Mean	STD
ESI 1	11	30,991	0.306	21,406	0.093
	12	16,209	0.176	11,860	0.034
	13	6774	0.130	5278	0.077
ESI 2	21	17,504	0.151	12,652	0.083
	22	7000	0.057	5339	0.011
	23	2559	0.041	1821	0.006
ESI 3	3	51,209	0.122	46,948	0.011
ESI 4	4	13,277	0.159	11,200	0.032
ESI 5	5	271	0.006	205	0.003
Total $kgCO_2$		145,794		116,709	

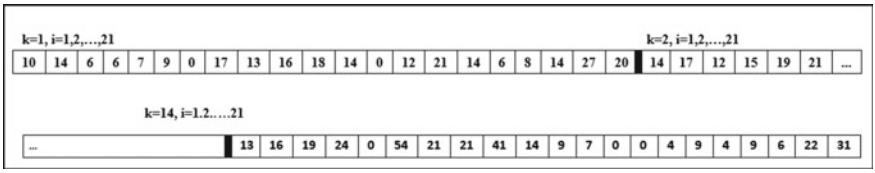


Fig. 1 General scheme of any solution

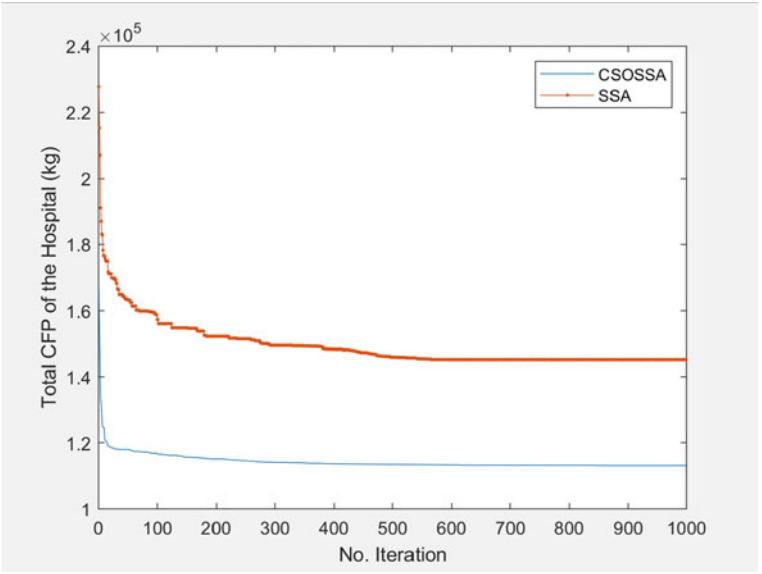


Fig. 2 Convergence graph for SSA and CSOSSA for CFP minimization

total CFP of the hospital in the patient flow with SSA and CSOSSA is 145,794 kg and 116,709 kg, respectively. Based on the results obtained from Table 6 and Fig. 2, the CSOSSA has better performance than the SSA based on ESI (therapeutic pathways) in terms of the performance measure: mean and STD.

Applying a Wilcoxon signed-rank test on the results of SSA and CSOSSA, the p-value is 0.0039. It shows that CSOSSA outperforms the original SSA with a statistically significant difference.

7 Conclusions and Future Insights

Climate change is occurring as a result of the accumulation of greenhouse gases in the atmosphere arising from the combustion of fossil fuels. The delivery of healthcare services produces exceptional greenhouse gas emissions, and with more electrical

power be consumed, more greenhouse gases, specifically CO_2, are produced. To solve these problems, SSA is integrated with the chaotic maps, sine cosine algorithm, and opposition-based learning. The proposed algorithm (CSOSSA) is compared with that of the original SSA and some other metaheuristic algorithms on CEC2005, and five De Jong's functions. The experimental results show that the performance of CSOSSA is better than or comparable with the SSA and other meta-heuristic algorithms. Besides, the problem of CO_2 minimization is solved using CSOSSA. The results indicate that the CSOSSA effective in reducing CO_2 in the patient flow.

For future works, it is suggested that the CSOSSA is extended for multi or many-objective optimization problems. The algorithm could be used in other applications such as renewable energy, optimal distributed generation placement (ODGP) problems, and flexible job scheduling.

Appendix

See Tables 7 and 8.

Table 7 Unimodal benchmark functions

Function	Dim	Range	Shift position	f_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	20	[–100,100]	[–30, –30,…, –30]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	20	[–10,10]	[–3, –3,…, –3]	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	20	[–100,100]	[–30, –30,…, –30]	0
$F_4(x) = \max_i \{ \lvert x_i \rvert, 1 \leq i \leq n \}$	20	[–100,100]	[–30, –30,…, –30]	0
$F_5(x) = \sum_{i=1}^{n-1} \lfloor 100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \rfloor$	20	[–30,30]	[–15, –15,…, –15]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	20	[–100,100]	[–750, –750,…, –750]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	20	[–1.28,1.28]	[–0.25, –0.25,…, –0.25]	0

Table 8 Multimodal benchmark functions

Function	Dim	Range	Position	f_{min}
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	20	[–500,500]	[–300,–300,…,–300]	– 418.9829 * 5
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	20	[–5.12, 5.12]	[–2,–2,…,–2]	0
$F_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$	20	[–32,32]		0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	20	[–600,600]	[–400,–400,…,–400]	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2\left[1 + 10sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 - a < x_i < a \\ k(-x_i - a)^m x_i < -a \end{cases}$	20	[–50,50]	[–30,–30,…,–30]	0
$F_{13}(x) =$ $0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2\left[1 + sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5100, 4)$	20	[–50,50]	[–100,–100,…,–100]	0

Index of Key Terms

CFP (Carbon Footprint)

CO2 (Carbon Dioxide)

CSOSSA (Chaotic Sine Cosine Opposition Based Salp Swarm Algorithm)

OBL (Opposition Based Learning)

SSA (Salp Swarm Algorithm)

References

1. Abdel-Basset, M., Abdel-Fatah, L., Sangaiah, A.K.: Metaheuristic algorithms: a comprehensive review. In: Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, pp. 185–231. Elsevier (2018)
2. Beheshti, Z., Shamsuddin, S.M.H.: A review of population-based meta-heuristic algorithms. Int. J. Adv. Soft Comput. Appl **5**(1), 1–35 (2013)
3. Gogna, A., Tayal, A.: Metaheuristics: review and application. J. Exp. Theor. Artif. Intell. **25**(4), 503–526 (2013)
4. Lin, J.-H., et al.: A chaotic Levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems. Comput. Inf. Technol. **2**(2), 56–63 (2012)
5. Gholizadeh, S., Baghchevan, A.: Multi-objective seismic design optimization of steel frames by a chaotic meta-heuristic algorithm. Eng. Comput. **33**(4), 1045–1060 (2017)
6. Yuan, X., et al.: Hydrothermal scheduling using chaotic hybrid differential evolution. Energy Convers. Manag. **49**(12), 3627–3633 (2008)
7. Ahmadi, M., Mojallali, H.: Chaotic invasive weed optimization algorithm with application to parameter estimation of chaotic systems. Chaos Solitons Fractals **45**(9–10), 1108–1120 (2012)
8. Arora, S., Anand, P.: Chaotic grasshopper optimization algorithm for global optimization. Neural Comput. Appl. **31**(8), 4385–4405 (2019)
9. Gao, S., et al.: Gravitational search algorithm combined with chaos for unconstrained numerical optimization. Appl. Math. Comput. **231**, 48–62 (2014)
10. Gandomi, A.H., et al.: Chaos-enhanced accelerated particle swarm optimization. Commun. Nonlinear Sci. Numer. Simul. **18**(2), 327–340 (2013)
11. Gandomi, A.H., Yang, X.-S.: Chaotic bat algorithm. J. Comput. Sci. **5**(2), 224–232 (2014)
12. Gandomi, A.H., et al.: Firefly algorithm with chaos. Commun. Nonlinear Sci. Numer. Simul. **18**(1), 89–98 (2013)
13. Mirjalili, S., Gandomi, A.H.: Chaotic gravitational constants for the gravitational search algorithm. Appl. Soft Comput. **53**, 407–419 (2017)
14. Wang, G.-G., et al.: Chaotic krill herd algorithm. Inf. Sci. **274**, 17–34 (2014)
15. Mirjalili, S., et al.: Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. **114**, 163–191 (2017)
16. Zhao, X., et al.: An opposition-based chaotic salp swarm algorithm for global optimization. IEEE Access **8**, 36485–36501 (2020)
17. Ali, T.A.A., et al.: Optimal design of IIR wideband digital differentiators and integrators using salp swarm algorithm. Knowl.-Based Syst. **182**, 104834 (2019)
18. Sun, Z.-X., et al.: Salp swarm algorithm based on blocks on critical path for reentrant job shop scheduling problems. In: International Conference on Intelligent Computing. Springer (2018)

19. Hasanien, H.M., El-Fergany, A.A.: Salp swarm algorithm-based optimal load frequency control of hybrid renewable power systems with communication delay and excitation cross-coupling effect. Electr. Power Syst. Res. **176**, 105938 (2019)
20. Ibrahim, A., et al.: Fish image segmentation using salp swarm algorithm. In: International Conference on Advanced Machine Learning Technologies and Applications. Springer (2018)
21. Verma, S., Shiva, C.K.: A novel salp swarm algorithm for expansion planning with security constraints. Iran. J. Sci. Technol. Trans. Electr. Eng. 1–10 (2020)
22. Kansal, V., Dhillon, J.S.: Emended salp swarm algorithm for multiobjective electric power dispatch problem. Appl. Soft Comput. **90**, 106172 (2020)
23. Messaoud, R.B.: Extraction of uncertain parameters of single and double diode model of a photovoltaic panel using salp swarm algorithm. Measurement **154**, 107446 (2020)
24. Chen, P., You, C., Ding, P.: Event classification using improved salp swarm algorithm based probabilistic neural network in fiber-optic perimeter intrusion detection system. Opt. Fiber Technol. **56**, 102182 (2020)
25. Ma, B., et al.: A comprehensive improved salp swarm algorithm on redundant container deployment problem. IEEE Access **7**, 136452–136470 (2019)
26. Elkassas, A.M., ElWakil, M.: Facility layout problem using salp swarm algorithm. In: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT). IEEE (2019)
27. Baygi, S.M.H., Karsaz, A.: A hybrid optimal PID-LQR control of structural system: a case study of salp swarm optimization. In: 2018 3rd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC). IEEE (2018)
28. Shi, X., et al.: A wireless sensor network node location method based on salp swarm algorithm. In: 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). IEEE (2019)
29. Das, S., Bhattacharya, A., Chakraborty, A.K.: Short-term hydro-thermal-wind scheduling using salp swarm algorithm. In: 2018 International Electrical Engineering Congress (iEECON). IEEE (2018)
30. Kumari, S., Shankar, G.: A novel application of salp swarm algorithm in load frequency control of multi-area power system. In: 2018 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES). IEEE (2018)
31. Fathy, A., Rezk, H., Nassef, A.M.: Robust hydrogen-consumption-minimization strategy based salp swarm algorithm for energy management of fuel cell/supercapacitor/batteries in highly fluctuated load condition. Renew. Energy **139**, 147–160 (2019)
32. Faris, H., et al.: An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. Knowl.-Based Syst. **154**, 43–67 (2018)
33. Ibrahim, R.A., et al.: Improved salp swarm algorithm based on particle swarm optimization for feature selection. J. Ambient Intell. Hum. Comput. **10**(8), 3155–3169 (2019)
34. Hegazy, A.E., Makhlouf, M., El-Tawel, G.S.: Improved salp swarm algorithm for feature selection. J. King Saud Univ.-Comput. Inf. Sci. **32**(3), 335–344 (2020)
35. Jumani, T.A., et al.: Salp swarm optimization algorithm-based controller for dynamic response and power quality enhancement of an islanded microgrid. Processes **7**(11), 840 (2019)
36. Patnana, N., Pattnaik, S., Singh, V.: Salp swarm optimization based PID controller tuning for Doha reverse osmosis desalination plant. Int. J. Pure Appl. Math. **119**, 12707–12720 (2018)
37. Saxena, P., et al.: Three dimensional route planning for multiple unmanned aerial vehicles using salp swarm algorithm (2019). arXiv:1911.10519
38. Hekimoğlu, B., et al.: Speed control of DC motor using PID controller tuned by salp swarm algorithm. In: Proceeding of IENSC (2018)
39. Panda, N., Majhi, S.K.: Improved salp swarm algorithm with space transformation search for training neural network. Arab. J. Sci. Eng. **45**(4), 2743–2761 (2020)
40. Qais, M.H., Hasanien, H.M., Alghuwainem, S.: Enhanced salp swarm algorithm: application to variable speed wind generators. Eng. Appl. Artif. Intell. **80**, 82–96 (2019)
41. El-Fergany, A.A.: Extracting optimal parameters of PEM fuel cells using salp swarm optimizer. Renew. Energy **119**, 641–648 (2018)

42. Bairathi, D., Gopalani, D.: Salp swarm algorithm (SSA) for training feed-forward neural networks. In: Soft Computing for Problem Solving, pp. 521–534. Springer (2019)
43. Sahu, P.C., et al.: Improved-salp swarm optimized type-II fuzzy controller in load frequency control of multi area islanded AC microgrid. Sustain. Energy Grids Netw. **16**, 380–392 (2018)
44. Yodphet, D., et al.: Electrical distribution system reconfiguration for power loss reduction by the Salp Swarm algorithm. Int. J. Smart Grid Clean Energy **2**, 156–163 (2019)
45. El-Fergany, A.A., Hasanien, H.M.: Salp swarm optimizer to solve optimal power flow comprising voltage stability analysis. Neural Comput. Appl. **32**(9), 5267–5283 (2020)
46. Tolba, M., et al.: A novel robust methodology based salp swarm algorithm for allocation and capacity of renewable distributed generators on distribution grids. Energies **11**(10), 2556 (2018)
47. Zhang, J., Wang, Z., Luo, X.: Parameter estimation for soil water retention curve using the salp swarm algorithm. Water **10**(6), 815 (2018)
48. Hussien, A.G., Hassanien, A.E., Houssein, E.H.: Swarming behaviour of salps algorithm for predicting chemical compound activities. In: 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS). IEEE (2017)
49. Asasi, M.S., Ahanch, M., Holari, Y.T.: Optimal allocation of distributed generations and shunt capacitors using salp swarm algorithm. In: Iranian Conference on Electrical Engineering (ICEE). IEEE (2018)
50. Liu, X., Xu, H.: Application on target localization based on salp swarm algorithm. In: 2018 37th Chinese Control Conference (CCC). IEEE (2018)
51. Kamel, S., et al.: Radial distribution system reconfiguration for real power losses reduction by using salp swarm optimization algorithm. In: 2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia). IEEE (2019)
52. Singh, A., Sharma, V.: Salp swarm algorithm-based model predictive controller for frequency regulation of solar integrated power system. Neural Comput. Appl. **31**(12), 8859–8870 (2019)
53. Zhong, C., et al.: Structural reliability assessment by salp swarm algorithm–based FORM. Qual. Reliab. Eng. Int. **36**(4), 1224–1244 (2020)
54. Wang, H., et al.: A novel nonlinear Arps decline model with salp swarm algorithm for predicting pan evaporation in the arid and semi-arid regions of China. J. Hydrol. **582**, 124545 (2020)
55. Liu, W., Wang, R., Su, J.: An image impulsive noise denoising method based on salp swarm algorithm. Int. J. Educ. Manag. Eng. **10**(1), 43 (2020)
56. Mallikarjuna, B., Reddy, Y., Kiranmayi, R.: Salp swarm algorithm to combined economic and emission dispatch problems. Int. J. Eng. & Technol. **7**(3.29), 311–315 (2018)
57. Hosseinpour, M., Mansoori, S., Shayeghi, H.: Selective harmonics elimination technique in cascaded H-bridge multi-level inverters using the salp swarm optimization algorithm. J. Oper. Autom. Power Eng. **8**(1), 32–42 (2020)
58. Wang, S., Jia, H., Peng, X.: Modified salp swarm algorithm based multilevel thresholding for color image segmentation. Math. Biosci. Eng. **17**, 700–724 (2020)
59. Wang, D., et al.: A simplex method-based salp swarm algorithm for numerical and engineering optimization. In: International Conference on Intelligent Information Processing. Springer (2018)
60. Sambaiah, K.S., Jayabarathi, T.: Optimal allocation of renewable distributed generation and capacitor banks in distribution systems using salp swarm algorithm. Int. J. Renew. Energy Res. **9**, 96–107 (2019)
61. Karnavas, Y.L., et al.: Application of salp swarm algorithm for DC motor parameter estimation in an industry 4.0 control systems IoT framework. In: 2019 12th International Conference on Developments in eSystems Engineering (DeSE). IEEE (2019)
62. Jiang, Y., et al.: Optimal nonlinear adaptive control for voltage source converters via memetic salp swarm algorithm: design and hardware implementation. Processes **7**(8), 490 (2019)
63. Ahmed, S., et al.: Feature selection using salp swarm algorithm with chaos. In: Proceedings of the 2nd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (2018)

64. Majhi, S.K., Mishra, A., Pradhan, R.: A chaotic salp swarm algorithm based on quadratic integrate and fire neural model for function optimization. Prog. Artif. Intell. **8**(3), 343–358 (2019)
65. Hegazy, A.E., Makhlouf, M., El-Tawel, G.S.: Feature selection using chaotic salp swarm algorithm for data classification. Arab. J. Sci. Eng. **44**(4), 3801–3816 (2019)
66. Sayed, G.I., Khoriba, G., Haggag, M.H.: A novel chaotic salp swarm algorithm for global optimization and feature selection. Appl. Intell. **48**(10), 3462–3481 (2018)
67. Bairathi, D., Gopalani, D.: Opposition based salp swarm algorithm for numerical optimization. In: International Conference on Intelligent Systems Design and Applications. Springer (2018)
68. Tubishat, M., et al.: Improved salp swarm algorithm based on opposition based learning and novel local search algorithm for feature selection. Expert Syst. Appl. **145**, 113122 (2020)
69. Meraihi, Y., et al.: A chaotic binary salp swarm algorithm for solving the graph coloring problem. In: International Symposium on Modelling and Implementation of Complex Systems. Springer (2018)
70. Khamees, M., Albakry, A., Shaker, K.: Multi-objective feature selection: hybrid of salp swarm and simulated annealing approach. In: International Conference on New Trends in Information and Communications Technology Applications. Springer (2018)
71. Yodphet, D., et al.: Network reconfiguration and capacitor placement for power loss reduction using a combination of salp swarm algorithm and genetic algorithm
72. Abd Elaziz, M., et al.: Multiobjective big data optimization based on a hybrid salp swarm algorithm and differential evolution. Appl. Math. Model. **80**, 929–943 (2020)
73. Wang, J., Gao, Y., Chen, X.: A novel hybrid interval prediction approach based on modified lower upper bound estimation in combination with multi-objective salp swarm algorithm for short-term load forecasting. Energies **11**(6), 1561 (2018)
74. Gholami, K., Parvaneh, M.H.: A mutated salp swarm algorithm for optimum allocation of active and reactive power sources in radial distribution systems. Appl. Soft Comput. **85**, 105833 (2019)
75. Aljarah, I., et al.: Asynchronous accelerating multi-leader salp chains for feature selection. Appl. Soft Comput. **71**, 964–979 (2018)
76. Alwerfali, H.S.N., et al.: A multilevel image thresholding based on hybrid salp swarm algorithm and fuzzy entropy. IEEE Access **7**, 181405–181422 (2019)
77. Singh, N., Chiclana, F., Magnot, J.-P.: A new fusion of salp swarm with sine cosine for optimization of non-linear functions. Eng. Comput. **36**(1), 185–212 (2020)
78. Sayed, G.I., Khoriba, G., Haggag, M.H.: Hybrid quantum salp swarm algorithm for contrast enhancement of natural images. Int. J. Intell. Eng. Syst. **12**(6), 225–235 (2019)
79. Chen, R., et al.: QSSA: quantum evolutionary salp swarm algorithm for mechanical design. IEEE Access **7**, 145582–145595 (2019)
80. Yaseen, Z.M., Faris, H., Al-Ansari, N.: Hybridized extreme learning machine model with salp swarm algorithm: a novel predictive model for hydrological application. Complexity **2020** (2020)
81. Kang, F., Li, J., Dai, J.: Prediction of long-term temperature effect in structural health monitoring of concrete dams using support vector machines with Jaya optimizer and salp swarm algorithms. Adv. Eng. Softw. **131**, 60–76 (2019)
82. Anderson, P., Bone, Q.: Communication between individuals in salp chains. II. Physiology. In: Proceedings of the Royal Society of London. Series B. Biological Sciences **210**(1181), 559–574 (1980)
83. Sutherland, K.R., Weihs, D.: Hydrodynamic advantages of swimming by salp chains. J. R. Soc. Interface **14**(133), 20170298 (2017)
84. Mirjalili, S.: SCA: a sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. **96**, 120–133 (2016)
85. Hongwu, L.: An adaptive chaotic particle swarm optimization. In: 2009 ISECS International Colloquium on Computing, Communication, Control, and Management. IEEE (2009)
86. dos Santos Coelho, L.: A quantum particle swarm optimizer with chaotic mutation operator. Chaos Solitons Fractals **37**(5), 1409–1418 (2008)

87. Tatsumi, K., Yamamoto, H., Tanino, T.: A perturbation based chaotic particle swarm optimization using multi-type swarms. In: 2008 SICE Annual Conference. IEEE (2008)
88. He, Y., et al.: A precise chaotic particle swarm optimization algorithm based on improved tent map. In: 2008 Fourth International Conference on Natural Computation. IEEE (2008)
89. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999)
90. Caponetto, R., et al.: Chaotic sequences to improve the performance of evolutionary algorithms. IEEE Trans. Evol. Comput. **7**(3), 289–304 (2003)
91. De Jong, K.A.: Analysis of the behavior of a class of genetic adaptive systems (1975)
92. Juneja, K., Gill, N.S.: Optimization of Dejong function using GA under different selection algorithms. Int. J. Comput. Appl. **64**(7) (2013)
93. Kapoor, M., Wadhwa, V.: Optimization of De Jong's function using genetic algorithm approach. Int. J. Adv. Res. Comput. Sci. Electron. Eng. **1**(5), 35–38 (2012)
94. Karaboğa, D., Ökdem, S.: A simple and global optimization algorithm for engineering problems: differential evolution algorithm. Turk. J. Electr. Eng. Comput. Sci. **12**(1), 53–60 (2004)
95. Kim, H.-S., et al.: Application of real-type Tabu search in function optimization problems. In: ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570). IEEE (2001)
96. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
97. Atashpaz-Gargari, E., Lucas, C.: Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: 2007 IEEE Congress on Evolutionary Computation. IEEE (2007)
98. Houck, C.R., Joines, J., Kay, M.G.: A genetic algorithm for function optimization: a Matlab implementation. NCSU-IE TR **95**(09), 1–10 (1995)
99. Tanabe, P., et al.: The Emergency Severity Index (version 3) 5-level triage system scores predict ED resource consumption **30**(1), 22–29 (2004)
100. WHO (2015) Quantitative risk assessment of the effects of climate change on selected causes of death, 2030s and 2050s. WHO, Geneva, Switzerland
101. Pollard, A.S., et al.: Mainstreaming carbon management in healthcare systems: A bottom-up modeling approach. Environ. Sci. Technol. **47**(2), 678–686 (2013)
102. Eckelman, M.J., Sherman, J.D., MacNeill, A.J.: Life cycle environmental emissions and health damages from the Canadian healthcare system: an economic-environmental-epidemiological analysis **15**(7), e1002623 (2018)
103. Chung, J.W., Meltzer, D.O.: Estimate of the carbon footprint of the US health care sector. JAMA **302**(18), 1970–1972 (2009)
104. The Emission Gas Report, November 2012 (2012)
105. Pollard, A., et al.: The carbon footprint of acute care: how energy intensive is critical care? Public Health **128**(9), 771–776 (2014)
106. Becker (2012) 18 statistics on hospital energy consumption. BECKER'S HOSPITAL REVIEW
107. CBECS (2012) Energy Characteristics and Energy Consumed in Large Hospital Buildings in the United States in 2007 in EIA

Design and Performance Evaluation of Objective Functions Based on Various Measures of Fuzzy Entropies for Image Segmentation Using Grey Wolf Optimization

Baljit Singh Khehra, Arjan Singh, and Lovepreet Kaur

Abstract Image segmentation is the most significant pre-processing phase of computer vision. Thresholding is one of most suitable approach used for the segmentation of image. Searching an optimal value of threshold for image segmentation has got more attention in recent years due to its accuracy and robustness. Over the years, number of researchers has successfully used Shannon and non-Shannon measures of fuzzy entropy based objective function in various evolutionary algorithms to find the optimal value of threshold for image segmentation. In this paper, grey wolf optimization (GWO) algorithm is used along with Shannon and non-Shannon measures of fuzzy entropy for selecting optimal threshold value to segment image in reasonable amount of time. The objective of the proposed research work is to design objective functions based on Shannon and non-Shannon measures of fuzzy entropy for GWO and using them for optimal threshold value selection which will further be used to segment images. Non-Shannon measures of fuzzy entropy used for design of objective functions are Renyi, Havrda-Charvat, Kapur, and M. Masi. The performance of Shannon and non-Shannon fuzzy entropies using GWO is compared with Shannon fuzzy entropy using GA, BBO and recursive approach to find the optimal threshold value. Comparison is done based on the computation time and four image segmentation quality evaluation parameters i.e. PSNR; uniformity; SSIM and MSSIM. The experiments are conducted on a set of ten different kinds of benchmark images that has variant characteristics. Experimental results indicate that the performance of fuzzy Shannon entropy using GWO is better than Shannon fuzzy entropies using GA, BBO, and recursive approach. From overall performances point of view, Havrda-Charvat fuzzy entropy using GWO is performing better than all other approaches in terms of quality of the segmented image and computational time point of view as well.

B. S. Khehra
BAM Khalsa College, Garhshankar, Hoshiarpur, Punjab 144527, India
e-mail: baljitkhehra@ieee.org

A. Singh (✉) · L. Kaur
Punjabi University, Patiala, Punjab 147002, India
e-mail: arjanpu@gmail.com

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_2

Keywords Image segmentation · Grey wolf optimization (GWO) · Shannon measures of fuzzy entropy · Havrda-Charvat fuzzy entropy · Structural Similarity index (SSIM) and Mean Structural Similarity index (MSSIM)

1 Introduction

Image segmentation is the process to divide an image into different subparts or subregions and is considered as the important pre-processing phase of computer vision. The main objective of image segmentation is to represent the image in a meaningful way so that it can be easily analyzed and interpreted for computer vision applications like autonomous object recognition, disease diagnosis in medical imaging, geographical imaging, etc. [1]. Image segmentation techniques can be classified into thresholding based, edge detection based, graph based, histogram based, region growing based, stochastic based models, fuzzy rule based, artificial neural network based and clustering based techniques [2]. Among all these segmentation techniques, thresholding technique is used widely due to its simplicity, accuracy and robustness [3]. Bi-level thresholding is one of the most frequently used thresholding-based techniques in which target is extracted from the input image using one threshold value. It segments the image into two segments: Target and Background. But, the main challenging task in bi-level thresholding technique is to select an optimal threshold value in a reasonable amount of time.

Different researchers have proposed different methods and techniques for appropriate selection of threshold value for image segmentation. Otsu [4] introduced a method to select the threshold value based on the variation of gray level values of the image. Pun [5] proposed a new approach to classify the image into classes based on Shannon entropy. Kapur et al. [6] proposed an approach to segment the image by selecting threshold value using the entropy of the histogram. Sahoo et al. [7] introduced a method to divide the image into regions by selecting threshold value using Renyi's measure of entropy. Cheng et al. [8] proposed a technique to select threshold value by using the concept of fuzzy c-partition entropy. Albuquerque et al. [9] proposed an approach for selection of threshold value to segment images based on Tsallis entropy. Shitong and Chung [10] disclose the equivalence relationship between Sahoo et al. and Albuquerque et al. methods. All such techniques suffer from serious drawback of selection of accurate threshold value in reasonable amount of time. Benabdelkader and Boulemden [11] proposed a new method based on fuzzy 2-partition entropy for selection of threshold value to segment image and a recursive algorithm has been applied to select appropriate parameters of fuzzy 2-partition entropy. Tang et al. [12] developed another recursive approach to select appropriate parameters of fuzzy 2-partition entropy to segment images. This approach tried to eliminate repeated computations for reducing the time complexity of approach. But, this method again suffers from computational burden. So over the years, fuzzy entropy has been widely used for discovering thresholding approaches for image segmentation. The main goal of fuzzy entropy based approaches is to select optimal

values of parameters of fuzzy entropy in reasonable amount of time under some constraints. So, selection of optimal values of parameters of fuzzy 2-partition entropy for image segmentation is an optimization problem.

Bio-inspired evolutionary algorithms are widely used for providing near to optimal solutions for vast range of optimization problems. Genetic algorithm [13], cuckoo-search [14, 15], bacterial foraging optimization [16], whale optimization [17], ant colony optimization [18], honey bee optimization [19] are some of the examples of such bio-inspired evolutionary algorithms. Moreover, these evolutionary algorithms have been used by researchers to find parameters of fuzzy 2-partition entropy for image segmentation. In 2013, Huang et al. [20] applied genetic algorithm (GA) to select parameters of fuzzy 2-partition entropy for segmentation of Ripe Fuji Apple image. Khehra et al. [21] applied big bang–big crunch optimization algorithm to select parameters of fuzzy 2-partition entropy for extracting object from back ground. Tosta et al. [22] converted input image into wavelet domain and then applied genetic algorithm to select parameters of fuzzy 2-partition entropy for segmentation of chronic lymphocytic leukemia images. Naidu et al. [2] used swarm intelligence algorithm to obtain threshold value for segmenting the foreground from background using Shannon fuzzy entropy. Pare et al. [23] applied Lévy flight firefly algorithm on modified fuzzy entropy for image segmentation. Wenbing Tao et al. [24] applied ant colony optimization (ACO) to obtain the optimal parameters of fuzzy 2-partition entropy for image segmentation. Chatterjee et al. [25] applied biogeography based optimization (BBO) approach on fuzzy entropy to segment CT-scan images**.** Khehra et al. [26, 27] used Teaching–Learning-based Optimization Algorithm and Fuzzy Entropy for image segmentation.

From literature, following four different observations have been made:

(a) Fuzzy entropy plays important role for image segmentation and the concept has been widely and successfully applied by various researchers for selecting threshold values.
(b) Researchers have applied hybrid approaches by using fuzzy entropy as objective function in evolutionary algorithms to search optimal threshold value for image segmentation.
(c) Most of the researchers have used Shannon fuzzy entropy as objective function in evolutionary algorithms for selection of optimal threshold value.
(d) Development of robust and automatic method for selection of optimal threshold value for image segmentation in reasonable amount of time is still an open problem.

Moreover, according to No Free Lunch (NFL) Theorem [28], there is no single metaheuristic technique that can be used to provide solutions for all kind of optimization problems. Each metaheuristic technique has its own advantages and disadvantages. All the metaheuristics techniques provide near to optimal solution instead of exact solution to a given optimization problems. All above mentioned factors have motivated us to explore new approaches for selection of optimal threshold value for the segmentation of a given image.

More recently, Mirjalili et al. [29] have developed a new metaheuristic optimization technique named as Grey Wolf Optimizer (GWO) algorithm. GWO is having the capability to produce quality solution to optimization problems than that of other metaheuristic technique as reported in the recent literature. GWO works very well in keeping the balance between exploration and exploitation which lead to global optimal solution rather than going to local optima [30]. GWO simulates social leadership hierarchy and intelligent hunting mechanism of grey wolves. From simulation point of view, four different types of grey wolves named as alpha, beta, delta, and omega has been introduced by Mirjalili et al. [29] to represent their social leadership hierarchy and intelligent hunting mechanism. Three main hunting strategies: searching for prey, encircling prey and attacking prey are implemented. During last five years, GWO algorithm has been widely applied to solve many optimization problems [31–35]. GWO has also been used by researchers for image segmentation [36, 37]. This inspires us to design objective functions based on Shannon and non-Shannon measures of fuzzy entropy for GWO which can be used for selection of optimal threshold value to segment images. Non-Shannon measures of entropy used here are Renyi et al. [38, 39]. The major contributions of this paper are summarized as:

1. Grey Wolf Optimization is proposed to find optimal threshold values for bi-level image segmentation.
2. Five different objective functions are designed to check the performance of the proposed method. One is based on Shannon fuzzy entropy and other four objective functions are based on non-Shannon measures of fuzzy entropy: Renyi, Havrda-Charvat, Kapur and M. Masi.
3. The performance of the proposed framework is compared with other well-known techniques for image segmentation i.e. GA-based, BBO-based and Recursive-based.
4. The performance is evaluated using a set of different kind of images that has variant characteristics.

The rest of the paper is organized as follows: Sect. 2 defines the problem based on Shannon and Non-Shannon fuzzy entropies. Section 3 introduces the Grey Wolf Optimization (GWO). Section 4 describes the proposed research work in detail. Experiment results are illustrated in Sect. 5. Final Sect. 6 concludes the research paper with some future research directions.

2 Problem Formulation Based on Shannon and Non-Shannon Fuzzy Entropies

Image segmentation partitioned a given image into different segments. Thresholding is one of the most commonly used techniques for image segmentation. However, the selection of optimal threshold value is most important to partition the image into appropriate segments. It requires a suitable optimization method to search an optimal

threshold value from *0* to $L_{\max}$ for optimum segmentation of the image; where $L_{\max}$ = 255 for gray scale image having pixel size of 1 byte. Suppose *g(x,y)* be input image of size $M \times N$; where g is the gray level value of the image at spatial coordinate position *(x, y)*. The range of gray levels of input image is $\{0, 1, 2, \ldots, j, \ldots, L_{\max}\}$. Let T be one of the value from $\{0, 1, 2, \ldots, j, \ldots, L_{\max}\}$ that segments the input image into two segments: S_1 and S_2, where $S_1 = \{0, 1, 2, \ldots, T\}$ and $S_2 = \{T + 1, T + 2, \ldots, L_{\max}\}$. Thus, segmented image *Seg (x, y)* is defined as:

$$Seg(x, y) = \begin{cases} 0 & g(x, y) \leq T \\ L_{\max} & Otherwise \end{cases} \tag{1}$$

In this paper, grey wolf optimization (GWO) algorithm is used to search optimal threshold value from $\{0, 1, 2, \ldots, j, \ldots, L_{\max}\}$. Fuzzy 2-partion entropy approach [8] has been applied for the design and development of objective functions which have to be maximized using GWO to find optimal threshold value for image segmentation. For the identification of two segments, two functions S and Z are used to quantify the degrees of association of each gray level j. Such functions are defined as follows:

$$S(j; a, b, c) = \begin{cases} 0 & j \leq a \\ \frac{(j-a)^2}{(c-a)(b-a)} & a < j \leq b \\ 1 - \frac{(j-c)^2}{(c-a)(c-b)} & b < j \leq c \\ 1 & j > c \end{cases} \tag{2}$$

$$Z(j; a, b, c) = \begin{cases} 1 & j \leq a \\ 1 - \frac{(j-a)^2}{(c-a)(b-a)} & a < j \leq b \\ \frac{(j-c)^2}{(c-a)(c-b)} & b < j \leq c \\ 0 & j > c \end{cases} \tag{3}$$

The variables *a, b* and *c* are used to find the membership degree ψ. The membership degree is evaluated as:

$$\psi_{S_1}(j) = S(j; a, b, c) \tag{4}$$

$$\psi_{S_2}(j) = Z(j; a, b, c) \tag{5}$$

The parameters *a, b* and *c* determine the distribution of the association degrees i.e. the shape of S and Z functions. Both fuzzy membership functions ψ_{S_1} and ψ_{S_2} intersect at a point as shown in Fig. 1. The parameters *a, b* and *c* must be satisfied the constraint: $0 \leq a \leq b \leq c \leq L_{\max}$. The gray level value corresponding to this intersection point is the threshold. Gray levels between *0* and intersection point have a large association to S_1. While gray levels between intersection point and $L_{\max}$ have a large association to S_2.

The probability distributions of segments S_1 and S_2 are defined as:

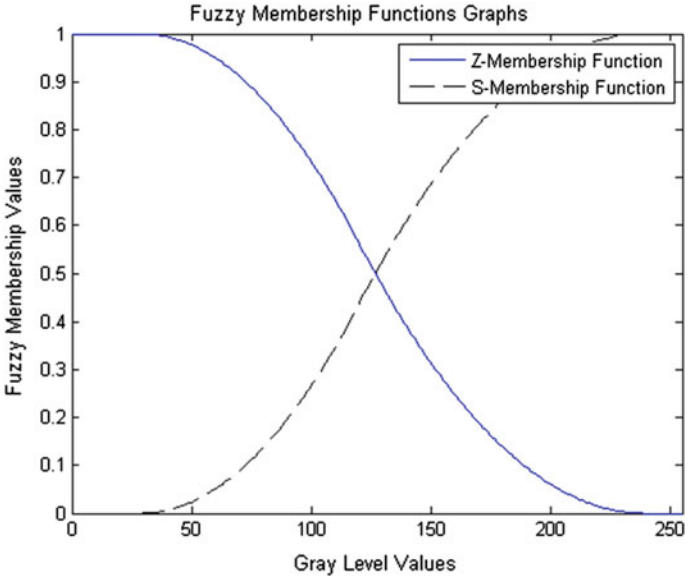


Fig. 1 Representation of Z and S fuzzy membership functions graphically

$$PD_{S_1} = \sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j \tag{6}$$

$$PD_{S_2} = \sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j \tag{7}$$

where p_j is the probability of occurrence of gray level j in the input image. It is called normalized histogram of the image and is defined as:

$$p_j = \frac{h_j}{M \times N} \tag{8}$$

where h_j is the frequency of occurrence of gray level j in the input image and is known as the histogram of the image.

The fuzzy entropy of each segment can be calculated using various measures of entropies [38].

2.1 Shannon Measure of Entropy

Shannon measure of entropy for both segments S_1 and S_2 is defined as:

$$H_S(a,b,c) = -\sum_{i=1}^{2} PD_{S_i} \log PD_{S_i} \tag{9}$$

$$H_S(a,b,c) = -[PD_{S_1} \log PD_{S_1} + PD_{S_2} \log PD_{S_2}] \tag{10}$$

$$H_S(a,b,c) = -\sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j \log \left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j \right) - \sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j \log \left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j \right) \tag{11}$$

Equation (11) represents the objective function based on Shannon measure of entropy.

2.2 Havrda-Charvat Measure of Entropy

Havrda-Charvat measure of entropy for segments S_1 and S_2 is defined as:

$$H_{HC}(a,b,c) = \frac{1}{1-\alpha} \left[\sum_{i=1}^{2} PD_{S_i}^{\alpha} - 1 \right] \tag{12}$$

where $\alpha > 0$, $\alpha \neq 1$

$$H_{HC}(a,b,c) = \frac{1}{1-\alpha} \left[PD_{S_1}^{\alpha} + PD_{S_2}^{\alpha} - 1 \right] \tag{13}$$

$$H_{HC}(a,b,c) = \frac{1}{1-\alpha} \left[\left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j \right)^{\alpha} + \left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j \right)^{\alpha} - 1 \right] \tag{14}$$

Equation (14) represents the objective function based on Havrda-Charvat measure of entropy.

2.3 Renyi Measure of Entropy

Renyi measure of entropy for segments $S_1$and S_2 is defined as:

$$H_R(a,b,c) = \frac{1}{1-\alpha}\log\left[\sum_{i=1}^{2} PD_{S_i}^{\alpha}\right] \tag{15}$$

where $\alpha > 0$, $\alpha \neq 1$

$$H_R(a,b,c) = \frac{1}{1-\alpha}\log\left[PD_{S_1}^{\alpha} + PD_{S_2}^{\alpha}\right] \tag{16}$$

$$H_R(a,b,c) = \frac{1}{1-\alpha}\log\left[\left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j)p_j\right)^{\alpha} + \left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j)p_j\right)^{\alpha}\right] \tag{17}$$

Equation (17) represents the objective function based on Renyi measure of entropy.

2.4 Kapur Measure of Entropy

Kapur measure of entropy for segments $S_1$and S_2 is defined as:

$$H_K(a,b,c) = \frac{1}{\beta-\alpha}\log\left[\frac{\sum_{i=1}^{2} PD_{S_i}^{\alpha}}{\sum_{i=1}^{2} PD_{S_i}^{\beta}}\right] \tag{18}$$

where α and β are two parameters that are constants and $\alpha > 0$, $\beta > 0$, $\alpha \neq \beta$.

$$H_K(a,b,c) = \frac{1}{\beta-\alpha}\log\left[\frac{PD_{S_1}^{\alpha} + PD_{S_2}^{\alpha}}{PD_{S_1}^{\beta} + PD_2^{\beta}}\right] \tag{19}$$

$$H_K(a,b,c) = \frac{1}{\beta-\alpha}\log\left[\frac{\left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j)p_j\right)^{\alpha} + \left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j)p_j\right)^{\alpha}}{\left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j)p_j\right)^{\beta} + \left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j)p_j\right)^{\beta}}\right] \tag{20}$$

Equation (20) represents the objective function based on Kapur measure of entropy.

2.5 M. Masi Measure of Entropy

M. Masi measure of entropy for segments $S_1$and S_2 is defined as:

$$H_{MM}(a,b,c) = \frac{1}{1-\alpha}\log\left[1-(1-\alpha)\left\{\sum_{i=1}^{2} PD_{S_i}\log PD_{S_i}\right\}\right] \tag{21}$$

$$H_{MM}(a,b,c) = \frac{1}{1-\alpha}\log\left[1-(1-\alpha)\left\{PD_{S_1}\log PD_{S_1} + PD_{S_2}\log PD_{S_2}\right\}\right] \tag{22}$$

$$H_{MM}(a,b,c) = \frac{1}{1-\alpha}\log\left[1-(1-\alpha)\left\{\begin{array}{l}\sum_{j=0}^{L_{\max}}\psi_{S_1}(j)p_j\log\left(\sum_{j=0}^{L_{\max}}\psi_{S_1}(j)p_j\right)\\ +\sum_{j=0}^{L_{\max}}\psi_{S_2}(j)p_j\log\left(\sum_{j=0}^{L_{\max}}\psi_{S_2}(j)p_j\right)\end{array}\right\}\right] \tag{23}$$

Equation (23) represents the objective function based on M. Masi measure of entropy.

Equations (11), (14), (17), (20) and (23) represent five different objective functions which are to be optimized using GWO separately. The idea is to measure the performance of each objective function for obtaining optimal threshold value. GWO is used to optimize or maximize *H(a, b, c)* by varying *a*, *b* and *c*.

$$(a^*, b^*, c^*) = \underset{a,b,c=0}{\overset{L_{\max}}{Arg\,\max}}[H(a,b,c)] \tag{24}$$

where $H \in \{H_S, H_{HC}, H_R, H_K, H_{MM}\}$ and a^*, b^* and c^* are the optimal combination of value for which H is maximized.

The optimal threshold value can be calculated by putting the optimal combination of values a^*, b^* and c^* in the following given formula [21, 26]:

$$T = \begin{cases} a^* + \sqrt{(c^*-a^*)(b^*-a^*)/2} & (a^*+c^*)/2 \le b^* \le c^* \\ c^* - \sqrt{(c^*-a^*)(c^*-b^*)/2} & a^* \le b^* \le (a^*+c^*)/2 \end{cases} \tag{25}$$

3 Grey Wolf Optimization (GWO)

GWO algorithm is a population based optimization method inspired by the behavior of grey wolves. It simulates the social leadership hierarchy of grey wolves and their hunting strategies. On the bases of hieratical social leadership setup, grey wolves can be categorized into four different levels: alpha (α), beta (β), delta (δ) and omega (Ω) as shown in Fig. 2. At the first level hierarchy, alpha wolves are leading wolves that are responsible for making all kind of decisions. These decisions are hunting, quiet place, time to wake, and other activities [29]. Beta is the next level in a grey wolf pack that acts as subordinates of alpha in all set of activities. Beta wolf is most suitable wolf that replaces the alpha. Alpha wolves are usually replaced when they die or become old. The next level from beta in the hierarchy of grey wolves is delta. Delta wolves have to submit to alpha wolves and beta wolves. Omega grey wolf is at the bottom of ranking. Omega wolves follow the order of all other categories of wolves. Simulation of social leadership hierarchy and main phases of group hunting mechanism of grey wolves are discussed below [33].

3.1 Simulation of Social Leadership Hierarchy of Grey Wolves

The fittest feasible solution (best feasible solution) to the problem is assumed as alpha (α) wolf whereas beta (β) wolf and delta (δ) wolf represent the second and third best solutions respectively. Omega (Ω) wolves represent the rest of all other feasible solutions. The optimization process is done under the supervision of alpha (α), beta (β) and delta (δ) while omega (Ω) wolves tail alpha (α), beta (β) and delta (δ) wolves.

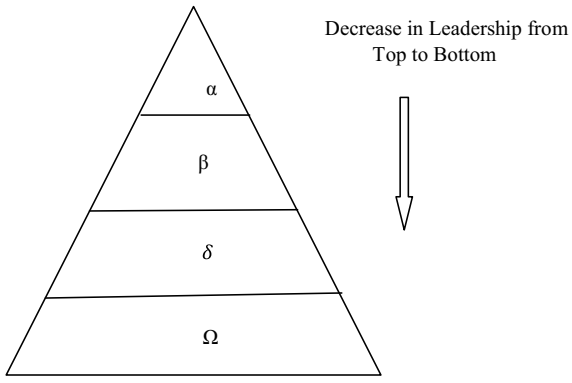


Fig. 2 Grey wolf hierarchy [29]

3.2 Simulation of Strategy of Grey Wolves for Encircling of Prey

During hunting, Grey wolves encircle around the prey. The following mathematical equations are used to model the hunting of grey wolves [37]:

$$\overrightarrow{D} = \left| \overrightarrow{C} . \overrightarrow{X}_{\text{Prey}}(k) - \overrightarrow{X}_{GW}(k) \right| \tag{26}$$

$$\overrightarrow{X}_{GW}(k+1) = \overrightarrow{X}_{\text{Prey}} - \overrightarrow{A} . \overrightarrow{D} \tag{27}$$

where k is the current iteration; $\overrightarrow{A}$ *and* $\overrightarrow{C}$ are coefficient vectors; $\overrightarrow{X}_{\text{Prey}}$ is the position vector of prey; $\overrightarrow{X}_{GW}$ is the position vector of a grey wolf.

Coefficient vectors $\overrightarrow{A}$ *and* $\overrightarrow{C}$ are defined as:

$$\overrightarrow{A} = 2 \cdot \overrightarrow{B} \cdot \overrightarrow{r_1} - \overrightarrow{B} \tag{28}$$

$$\overrightarrow{C} = 2 \cdot r_2 \tag{29}$$

where r_1 and r_2 are two numbers chosen randomly and lies between [0,1]; parameter $\overrightarrow{B}$ is linearly decreasing from 2 to 0 over the course of iterations and is defined as:

$$\overrightarrow{B} = 2 - k \cdot (2/I_{\max}) \tag{30}$$

where $I_{\max}$ is number of iterations.

3.3 Simulation of Strategy of Grey Wolves for Hunting of Prey

Grey wolves have the ability to identify the location of prey. This is done under the guidance of the alpha wolves. Occasionally beta and delta wolves also participate in the hunting process. It is assumed that all these three categories of wolves have better understanding of hunting as compare to omega wolves. Therefore for simulation point of view, initial optimization process starts with storing the three best solutions and they are called the searching agents. All other solutions update their position as per the position of the best solution (search agent) to improve the quality of solutions. To further improve the quality of the best solutions, the first three solutions (search agents) are modified as follows by updating their score and positions [30]:

$$\overrightarrow{D}_{\alpha} = \left| \overrightarrow{C}_{1} \cdot \overrightarrow{X}_{\alpha} - \overrightarrow{X} \right| \tag{31}$$

$$\overrightarrow{D}_{\beta} = \left| \overrightarrow{C}_{2} \cdot \overrightarrow{X}_{\beta} - \overrightarrow{X} \right| \tag{32}$$

$$\overrightarrow{D}_{\delta} = \left| \overrightarrow{C}_{3} \cdot \overrightarrow{X}_{\delta} - \overrightarrow{X} \right| \tag{33}$$

The prey's position vector w. r. t α, β, and δ wolves is modifies as follows:

$$\overrightarrow{X}_{1} = \overrightarrow{X}_{\alpha} - \overrightarrow{A}_{1} \cdot \overrightarrow{D}_{\alpha} \tag{34}$$

$$\overrightarrow{X}_{2} = \overrightarrow{X}_{\beta} - \overrightarrow{A}_{2} \cdot \overrightarrow{D}_{\beta} \tag{35}$$

$$\overrightarrow{X}_{3} = \overrightarrow{X}_{\delta} - \overrightarrow{A}_{3} \cdot \overrightarrow{D}_{\delta} \tag{36}$$

The best position can be updated by taking mean of α, β, and δ wolves as follows:

$$\overrightarrow{X}(k+1) = \frac{\overrightarrow{X}_{1} + \overrightarrow{X}_{2} + \overrightarrow{X}_{3}}{3} \tag{37}$$

3.4 General GWO Algorithm

The general GWO algorithm is described as follows:

Step 1: Initialization

- Grey Wolves Population: N_{sa}
- Iterations: $I_{\max}$
- Population of search agents: $X[1,2,\ldots,N_{sa}]$

Step 2: Set Iteration Counter, $k = 0$
Step 3: Evaluate of fitness of each search agent in Grey Wolf population based on objective functions.
Step 4: Identify the best, 2nd best and 3rd best search agents from the population based on their fitness values. Such search agents are called alpha, beta and delta search agents.
X_{α} = 1st Best search agent
X_{β} = 2nd Best search agent
X_{δ} = 3rd Best search agent
Step 5: Calculate the value of parameter B that is linearly decreasing from 2 to 0 during iterations process

$$B = 2 - k * (2/I_{\max})$$

Step 6: Repeat from 1 to Nsa

Step 6.1: Update the Score and position of alpha, beta and delta search agents

$$D_\alpha = |C_1 * X_\alpha - X|$$

$$D_\beta = \left|C_2 * X_\beta - X\right|$$

$$D_\delta = |C_3 * X_\delta - X|$$

$$C = 2 * r_2$$

where r_2 is a random number lies between [0,1]
Step 6.2: Update the position of prey w.r.t alpha, beta and delta wolves

$$X_1 = X_\alpha - A_1 * D_\alpha$$

$$X_2 = X_\beta - A_2 * D_\beta$$

$$X_3 = X_\delta - A_3 * D_\delta$$

$$A = 2 * B * r_1 - B$$

where r_1 is a random number lies between [0,1]
Step 6.3: Update the position of search agent

$$X = (X_1 + X_2 + X_3)/3$$

Step 7: End of Step 6
Step 8: Increment in iteration i.e. $k = k + 1$
Step 9: Go to Step 3 until $k < I_{\max}$
Step 10: Get Optimal Solution $= X_\alpha$

General GWO algorithm is also described in flowchart form shown in Fig. 3.

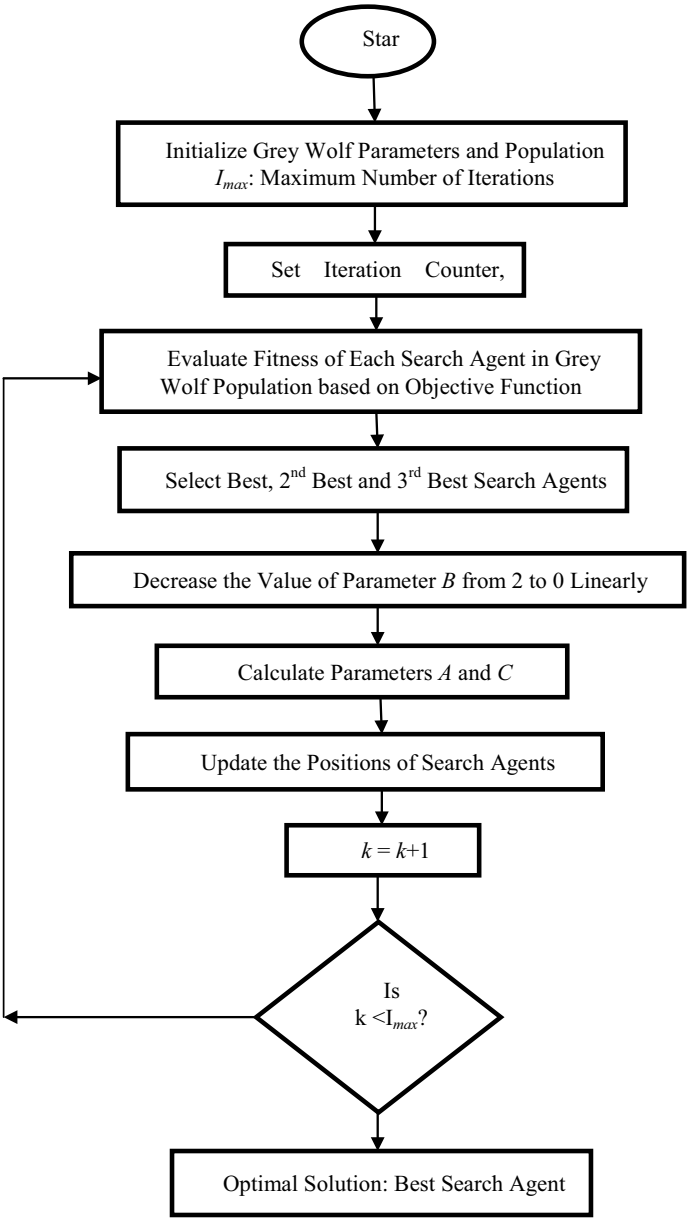


Fig. 3 Flowchart of general GWO algorithm

4 Proposed Approach

Grey wolf optimization (GWO) approach has been proposed to search optimal values of parameters of fuzzy entropy. Searched optimal parameters of fuzzy entropy are used for searching optimal values of threshold to segment images. Normally, Shannon measure of entropy is used by researchers for finding entropy of image in fuzzy domain. In this research work, non-Shannon measures of entropy are also explored. The concept of searching optimal values of parameters of fuzzy Shannon as well as non-Shannon (Havrda-Charvat, Renyi, Kapur and M. Masi) measures of entropy using GWO approach has not been explored for bi-level image segmentation. So, the contribution of this research work is novel. The proposed research work used the concept of fuzzy Shannon and non-Shannon measures of entropy and GWO approach to segment input digital into two segments; one is object and second is background of the object. Following are the five different approaches proposed in the present work:

(a) Shannon Fuzzy Entropy-Based-GWO (SFE-Based-GWO) approach
(b) Havrda-Charvat Fuzzy Entropy-Based-GWO (HCFE-Based-GWO) approach
(c) Renyi Fuzzy Entropy-Based-GWO (RFE-Based-GWO) approach
(d) Kapur Fuzzy Entropy-Based-GWO (KFE-Based-GWO) approach
(e) M. Masi Fuzzy Entropy-Based-GWO (MMFE-Based-GWO) approach.

4.1 GWO Based Image Segmentation Algorithm

Various steps of the proposed framework are given below:

Step 1: Initialization

Step 1.1: Grey wolf Parameters

- Population size (Number of search agents): N_{sa}
- Iterations: $I_{\max}$
- Dimension of Search Space: 0 to $L_{\max}$
- Fuzzy design variables are a, b, and c, where the values of these variables lies between $[0, L_{\max}]$

Step 1.2: Grey wolf Population

Grey wolf Population is generated randomly and depends upon Nsa as well as fuzzy design variables.

$$Pop = X[Nsa, 3]$$

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ . & . & . \\ . & . & . \\ X_{N_{sa}1} & X_{N_{sa}2} & X_{N_{sa}3} \end{bmatrix}$$

$$\{X_{i1}\} \in a, \{X_{i2}\} \in b, \{X_{i3}\} \in c, i = 1, 2, 3, \ldots, N_{sa}$$

Population is generated randomly in which design variables may or may not fall under their limits such that $0 \leq X_{i1} \leq X_{i2} \leq X_{i3} \leq L_{\max}$. To resolve such issue, some mathematical processing is required:

$$X'_{i2} = X_{i2}$$

$$X'_{i3} = X'_{i2} + (255 - X'_{i2}) * (X_{i3}/255)$$

$$X'_{i1} = X'_{i2} * (X_{i1}/255)$$

$$\{X'_{i1}\} \in a, \{X'_{i2}\} \in b, \{X'_{i3}\} \in c$$

Now, each solution is represented as:

$$X'_i \leftarrow [\, X'_{i1} \;\; X'_{i2} \;\; X'_{i3} \,]$$

$$X'_i = [a_i \;\; b_i \;\; c_i]$$

After performing above said operations, the population can be re-written as follows:

$$Pop = X' = \begin{bmatrix} X'_{11} & X'_{12} & X'_{13} \\ X'_{21} & X'_{22} & X'_{23} \\ . & . & . \\ . & . & . \\ X'_{N_{sa}2} & X'_{N_{sa}2} & X'_{N_{sa}3} \end{bmatrix}$$

So population can be represented as follows:

$$X^k = X' = \begin{bmatrix} X_{11}^k & X_{12}^k & X_{13}^k \\ X_{21}^k & X_{22}^k & X_{23}^k \\ . & . & . \\ . & . & . \\ X_{N_{sa}2}^k & X_{N_{sa}2}^k & X_{N_{sa}3}^k \end{bmatrix}$$

where $k = 0, 1, 2, \ldots, I_{\max}$.

Initialize k = 0

Step 2: Evaluation of Fitness of Search Agent in Grey Wolf Population

The following functions are used for the evaluation of fitness of search agent:

(a) **Function based on fuzzy Shannon entropy**

$$F_i^k = H(X_i^k) = -\sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j \log\left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j\right) - \sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j \log\left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j\right)$$

(b) **Functions based on fuzzy non-Shannon measures of entropy**

(i) **Function based on fuzzy Havrda-Charvat entropy**

$$F_i^k = H(X_i^k) = \frac{1}{1-\alpha}\left[\left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j\right)^{\alpha} + \left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j\right)^{\alpha} - 1\right]$$

where α is a parameter that is constant and $\alpha > 0, \alpha \neq 1$

(ii) **Function based on fuzzy Renyi entropy**

$$F_i^k = H(X_i^k) = \frac{1}{1-\alpha} \log\left[\left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j\right)^{\alpha} + \left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j\right)^{\alpha}\right]$$

where α is a parameter that is constant and $\alpha > 0, \alpha \neq 1$

(iii) **Function based on fuzzy Kapur entropy**

$$F_i^k = H(X_i^k) = \frac{1}{\beta-\alpha} \log\left[\frac{\left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j\right)^{\alpha} + \left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j\right)^{\alpha}}{\left(\sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j\right)^{\beta} + \left(\sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j\right)^{\beta}}\right]$$

where α, β are two parameters that are constant and $\alpha, \beta > 0, \alpha \neq \beta$.

(iv) **Function based on fuzzy M. Masi entropy**

$$F_i^k = H(X_i^k) = \frac{1}{1-\alpha}\log\left[1-(1-\alpha)\left\{\begin{array}{l}\sum_{j=0}^{L_{\max}}\psi_{S_1}(j)p_j\log\left(\sum_{j=0}^{L_{\max}}\psi_{S_1}(j)p_j\right)\\ +\sum_{j=0}^{L_{\max}}\psi_{S_2}(j)p_j\log\left(\sum_{j=0}^{L_{\max}}\psi_{S_2}(j)p_j\right)\end{array}\right\}\right]$$

where α is a parameter that is constant and $\alpha > 0, \alpha \neq 1$

On the basis of fitness value, all the search agents are arranged in descending order.

X_i^k and its respective fitness value F_i^k are

$$X^k = \begin{bmatrix} X_{11}^k & X_{12}^k & X_{13}^k \\ X_{21}^k & X_{22}^k & X_{23}^k \\ . & . & . \\ . & . & . \\ X_{N_{sa}1}^k & X_{N_{sa}2}^k & X_{N_{sa}3}^k \end{bmatrix} = \begin{bmatrix} X_1^k \\ X_2^k \\ . \\ . \\ X_{N_{sa}}^k \end{bmatrix}$$

$$F^k = \begin{bmatrix} F_1^k \\ F_2^k \\ . \\ . \\ F_{N_{sa}}^k \end{bmatrix}$$

$$\{X_{i1}^k\} \in a, \{X_{i2}^k\} \in b, \{X_{i3}^k\} \in c$$

Step 3: Selection of best, second best and third best candidate solutions

$$X_\alpha^k = X_1^k = X_{H(X_i^k)=\max}^k$$

$$X_\beta^k = X_2^k = X_{H(X_i^k)=second\ \max}^k$$

$$X_\delta^k = X_3^k = X_{H(X_i^k)=third\ \max}^k$$

Step 4: Calculate *B*

The value of B decreases from 2 to 0 linearly during the iteration process

$$B = 2 - k * (2/I_{\max})$$

Step 5: For each candidate solution (search or hunt agent)

i.e. $for\ i = 1 : N_{sa}\ do$

Step 6: Update the position of each grey wolf according to the position of the prey

For each design variable in candidate solution i.e. $for\ j = 1 : 3\ do$

$$A_1 = 2 * B * r_1 - B$$

$$C_1 = 2 * r_2$$

$$D_\alpha = \left|C_1 * X_\alpha^k(j) - X^k(i, j)\right|$$

$$Y_1 = X_\alpha^k(j) - A_1 * D_\alpha$$

$$A_2 = 2 * B * r_1 - B$$

$$C_2 = 2 * r_2$$

$$D_\beta = \left|C_2 * X_\beta^k(j) - X^k(i, j)\right|$$

$$Y_2 = X_\beta^k(j) - A_2 * D_\beta$$

$$A_3 = 2 * B * r_1 - B$$

$$C_3 = 2 * r_2$$

$$D_\delta = \left|C_3 * X_\delta^k(j) - X^k(i, j)\right|$$

$$Y_3 = X_\delta^k(j) - A_3 * D_\delta$$

$$Y(i, j) = (Y_1 + Y_2 + Y_3)/3$$

where r_1 and r_2 are random numbers and their values lies in [0,1].

Step 7: end for of Step 6.
Step 8: Check the feasibility of generated candidate solution

$$
\begin{aligned}
Y_i = & [\, Y_{i,1}\ Y_{i,2}\ Y_{i,3}\,] \\
& if\ 0 \le Y_{i,1} \le Y_{i,2} \le Y_{i,3} \le L_{\max} \\
& \quad X_i^k = Y_i \\
& else \\
& \quad X_i^k = X_i^k \\
& end\ if
\end{aligned}
$$

Step 9: *end for* of Step 5.
Step 10: Increment in iteration

$k = k + 1$
Step 11: Stopping criteria

Go to Step 2 until $k < I_{\max}$
Step 12: Values of *a, b* and *c*

$$a = X_{11}^{I_{\max}}, b = X_{12}^{I_{\max}}, c = X_{13}^{I_{\max}}$$

Step 13: Optimal threshold value

$$
\begin{aligned}
& If \quad ((a+c)/2) \le b \ \&\ \&\ b \le c \\
& \quad T = a + \sqrt{(c-a)(b-a)/2} \\
& Endif \\
& If \quad a \le b \ \&\ \&\ b \le ((a+c)/2) \\
& \quad T = a + \sqrt{(c-a)(b-a)/2} \\
& Endif
\end{aligned}
$$

Step 14: Stop

4.2 Procedure for Computing Fitness of Search Agents of Population Using Fuzzy Shannon and Non-Shannon Entropies

Five different measures of fuzzy entropies (Shannon, Havrda-Charvat, Renyi, Kapur and M. Masi) are used as objective function in this research work. The complete procedure for evaluating the fitness value of each candidate solution of the population using all five measures of fuzzy entropy is given below:

Step1: Read the image

$g(x, y)$; g is the gray level intensity at a particular coordinator point (x, y)

Step 2: Find the normalized histogram of the input image

$$p_j = \frac{h_j}{M \times N}$$

where, h_j is the histogram of the input image; $M \times N$ is the size of input image; $j = 0, 1, 2, \ldots, L_{\max}$

Step 3: Calculate fuzzy entropy of S_1 and S_2

$$for\ i = 1\ to\ N_{sa}$$

$$a = X[i, 1]$$

$$b = X[i, 2]$$

$$c = X[i, 3]$$

Step 3.1: Converting input image into fuzzy domain using S-fuzzy membership function

$$\begin{aligned}
&for\ \ j = 0\ to\ L_{\max}\\
&\quad if\quad j \le a\\
&\qquad \psi_{S_1}(j) = 0\\
&\quad endif\\
&\quad if\quad a < j \le b\\
&\quad \psi_{S_1}(j) = (j-a)^2/((c-a)(b-a))\\
&\quad endif\\
&\quad if\quad b < j \le c\\
&\qquad \psi_{S_1}(j) = 1 - (j-c)^2/((c-a)(c-b))\\
&\quad endif\\
&\quad if\quad j > c\\
&\qquad \psi_{S_1}(j) = 1\\
&\quad endif\\
&endfor
\end{aligned}$$

Step 3.2: Converting input image into fuzzy domain using Z-fuzzy membership function i.e. background of object in fuzzy domain

$$
\begin{aligned}
&for \;\; j = 0 \; to \; L_{\max} \\
&\quad if \quad j \le a \\
&\quad \psi_{S_2}(j) = 1 \\
&\quad endif \\
&\quad if \quad a < j \le b \\
&\qquad \psi_{S_2}(j) = 1 - (j-a)^2/((c-a)(b-a)) \\
&\quad endif \\
&\quad if \quad b < j \le c \\
&\qquad \psi_{S_2}(j) = (j-c)^2/((c-a)(c-b)) \\
&\quad endif \\
&\quad if \quad j > c \\
&\qquad \psi_{S_2}(j) = 0 \\
&\quad endif \\
&endfor
\end{aligned}
$$

Step 3.3: Probability Distribution of S_1

$$PD_{S_1} = \sum_{j=0}^{L_{\max}} \psi_{S_1}(j) p_j$$

Step 3.4: Probability Distribution of S_2

$$PD_{S_2} = \sum_{j=0}^{L_{\max}} \psi_{S_2}(j) p_j$$

Step 3.5: Find fuzzy entropy of S1 region and S2 region using different measures of entropy

(i) **Shannon measure of entropy**

$$H = -PD_{S_1} \log PD_{S_1} - PD_{S_2} \log PD_{S_2}$$

(ii) **Havrda-Charvat measure of entropy**

$$H = \frac{1}{1-\alpha}\left[\left(PD_{S_1}\right)^{\alpha} + \left(PD_{S_2}\right)^{\alpha} - 1\right]$$

where α is a parameter that is constant and $\alpha > 0, \alpha \neq 1$

(iii) **Renyi measure of entropy**

$$H = \frac{1}{1-\alpha} \log\left[\left(PD_{S_1}\right)^{\alpha} + \left(PD_{S_2}\right)^{\alpha}\right]$$

where α is a parameter that is constant and $\alpha > 0, \alpha \neq 1$

(iv) **Kapur measure of entropy**

$$H = \frac{1}{\beta-\alpha}\left[\frac{\left(PD_{S_1}\right)^{\alpha} + \left(PD_{S_2}\right)^{\alpha}}{\left(PD_{S_1}\right)^{\beta} + \left(PD_{S_2}\right)^{\beta}}\right]$$

where α, β are two parameters that are constants and $\alpha, \beta > 0, \alpha \neq \beta$

(v) **M. Masi measure of entropy**

$$H = \frac{1}{1-\alpha} \log\left[1 - (1-\alpha)\left\{PD_{S_1} \log PD_{S_1} + PD_{S_2} \log PD_{S_2}\right\}\right]$$

where α is a parameter that is constant and $\alpha > 0, \alpha \neq 1$

Step 3.6: *endfor*

Step 4: Stop

5 Experimental Results and Performance Analysis

The simulation has been done in MATLAB (R2019a) on a computer having Intel(R) Core(TM) i5-5300u CPU @ 2.30 GHz and 8 GB RAM to validate the proposed approach. The experiments are conducted on a set of ten different kinds of benchmark images that has variant characteristics. The name, type and resolution of all the images are tabulated in the Table 1. Images I_3 and I_7 are taken from [40] whereas rest of the images is taken from MATLAB standard test images dataset. These original input images and their respective histograms are shown in Figs. 4a, 5a, 6a, 7a, 8a, 9a, 10a, 11a, 12a, 13a and Figs. 4b, 5b, 6b, 7b, 8b, 9b, 10b, 11b, 12b, 13b respectively.

The performance evaluation of all the five approaches i.e. SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO and MMFE-Based-GWO is done against two well-known metaheuristics techniques and one non- metaheuristics approach available in the literature. Metaheuristics techniques selected for comparison are Shannon fuzzy entropy using GA and BBO whereas non- metaheuristics approach selected for comparison is Shannon fuzzy entropy using Recursive approaches.

Evaluation process is important as well as complex. Therefore, the evaluation of experimental results has been done from qualitative as well as quantitative point of view. From qualitative view point, Figs. 4(c–g), 5(c–g), 6(c–g), 7(c–g), 8(c–g), 9(c–g), 10(c–g), 11(c–g), 12(c–g), 13(c–g) clearly show the visual acceptability of

Table 1 Name, type and resolution of the images used for performance analysis

Serial number of the image	Name of the image	Type of the image	Resolution of the image
I_1	Circuit	tif	(280 × 272)
I_2	Coins	png	(246 × 300)
I_3	Dining Table (#7)	gif	(512 × 512)
I_4	Shadow	tif	(223 × 298)
I_5	Eight	tif	(242 × 308)
I_6	Glass	png	(181 × 282)
I_7	Bird Image (#9)	gif	(512 × 512)
I_8	Westconcordorthophoto	png	(394 × 369)
I_9	Peppers	png	(384 × 512)
I_{10}	Onion	png	(135 × 198)

all the five proposed approaches to effectively segment all the ten input images. Peak signal to noise ratio (PSNR) [41, 42], uniformity [43, 44], structural similarity index (SSIM) [45] and mean structural similarity index (MSSIM) [45] are four most popular measures used to check the efficiency of an approach from quantitative view point.

PSNR is used as a measure to show the dissimilarity between the input image and the segmented image. Decibels (dB) are the units of PSNR. Higher PSNR points toward better quality of the segmented image. Formula to calculate the PSNR is given as follows:

$$PSNR = 20\log_{10}\left(\frac{255}{RMSE}\right) \tag{38}$$

where RMSE is the root mean-squared error and the formula to calculate RMSE is given as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[g(i,j) - h(i,j)]^2}{M * N}} \tag{39}$$

where $g(i, j)$ is the input image and $h(I, j)$ is the segmented images. Both g and h are of size $M \times N$.

Uniformity is the parameter which helps in measuring the region homogeneity in an image. Higher the value of uniformity better is the quality of segmented image. The value of the uniformity lies between 0 and 1. Formula to calculate the uniformity is given as follows:

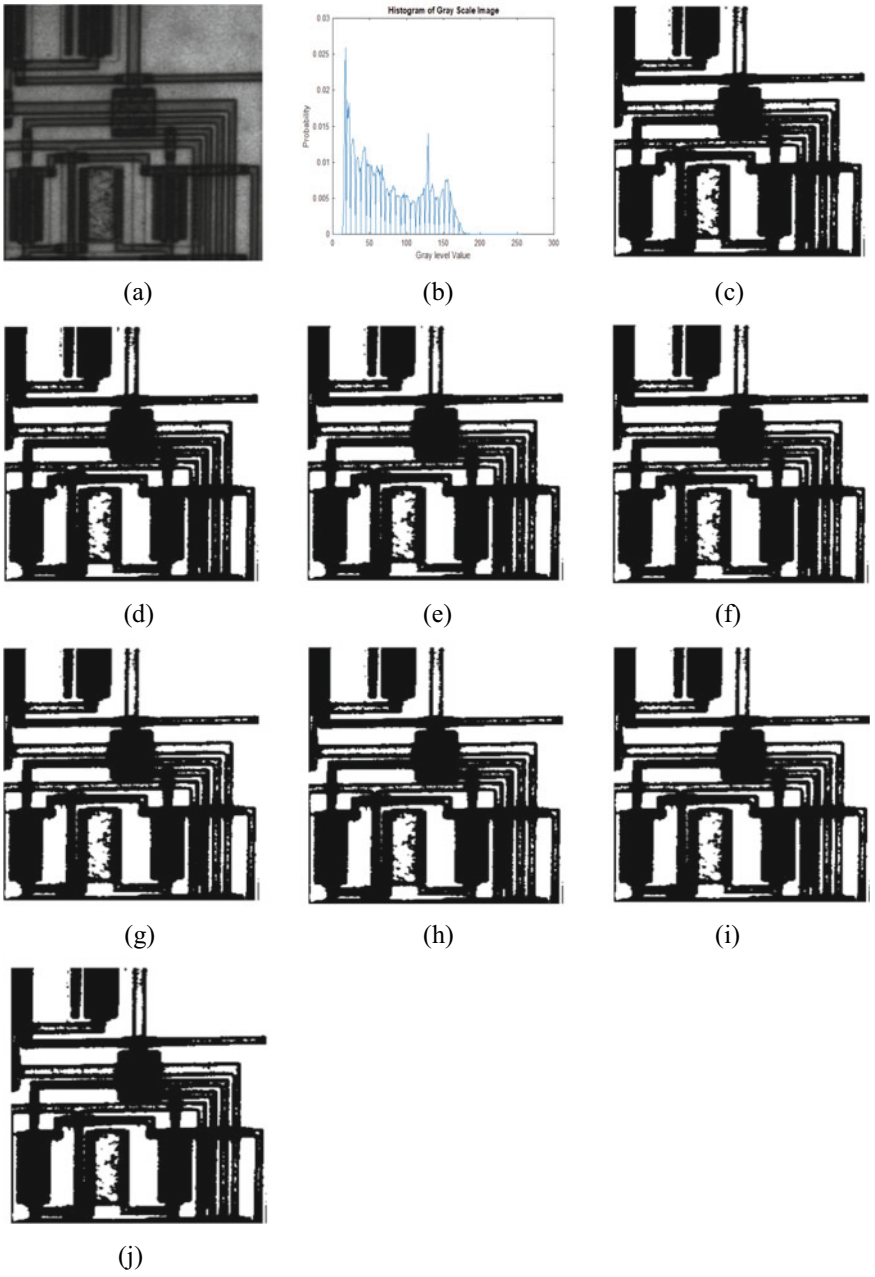


Fig. 4 "circuit.tif (280 × 272)" Image **a** Original image; **b** Histogram; **c–j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

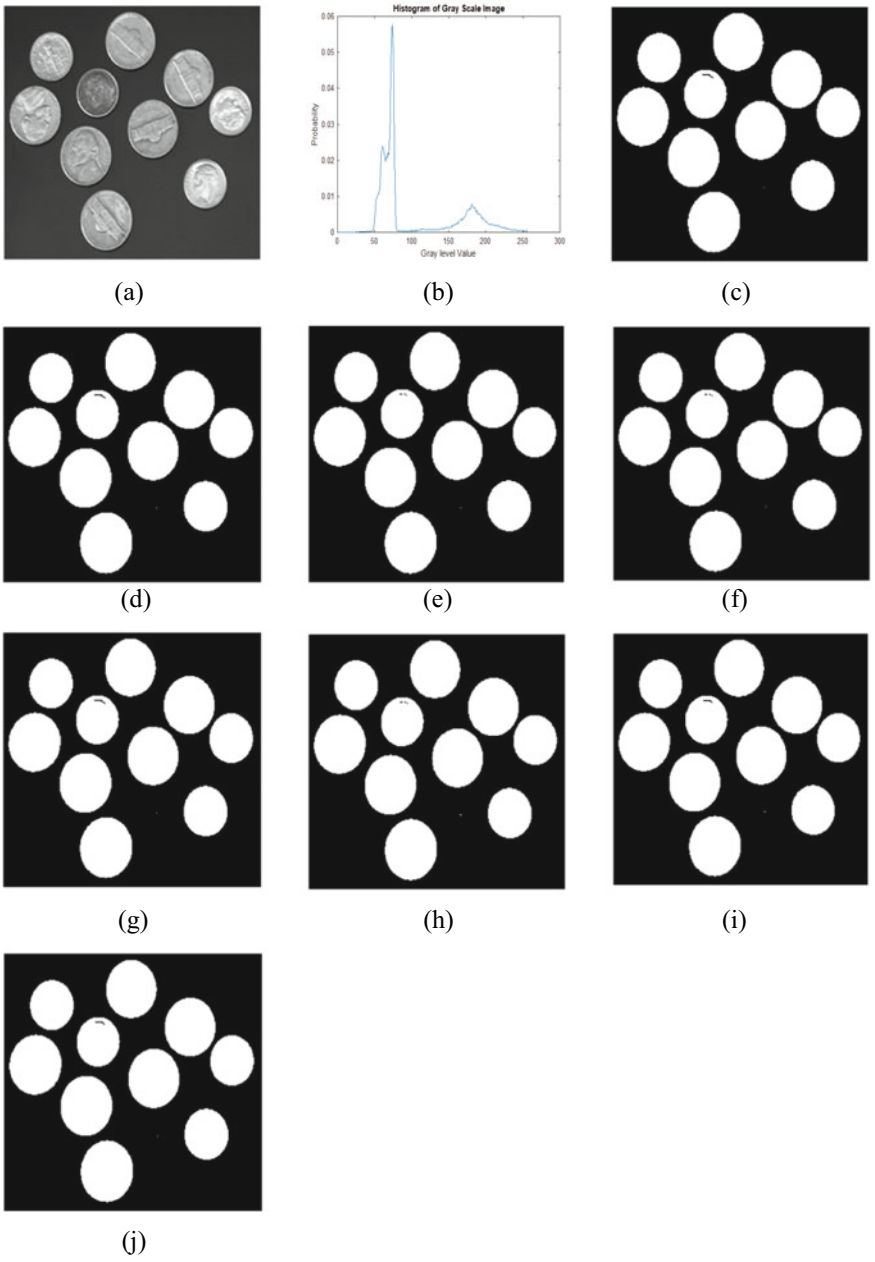


Fig. 5 "coins.png (246 × 300)" image **a** Original image; **b** Histogram; **c–j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

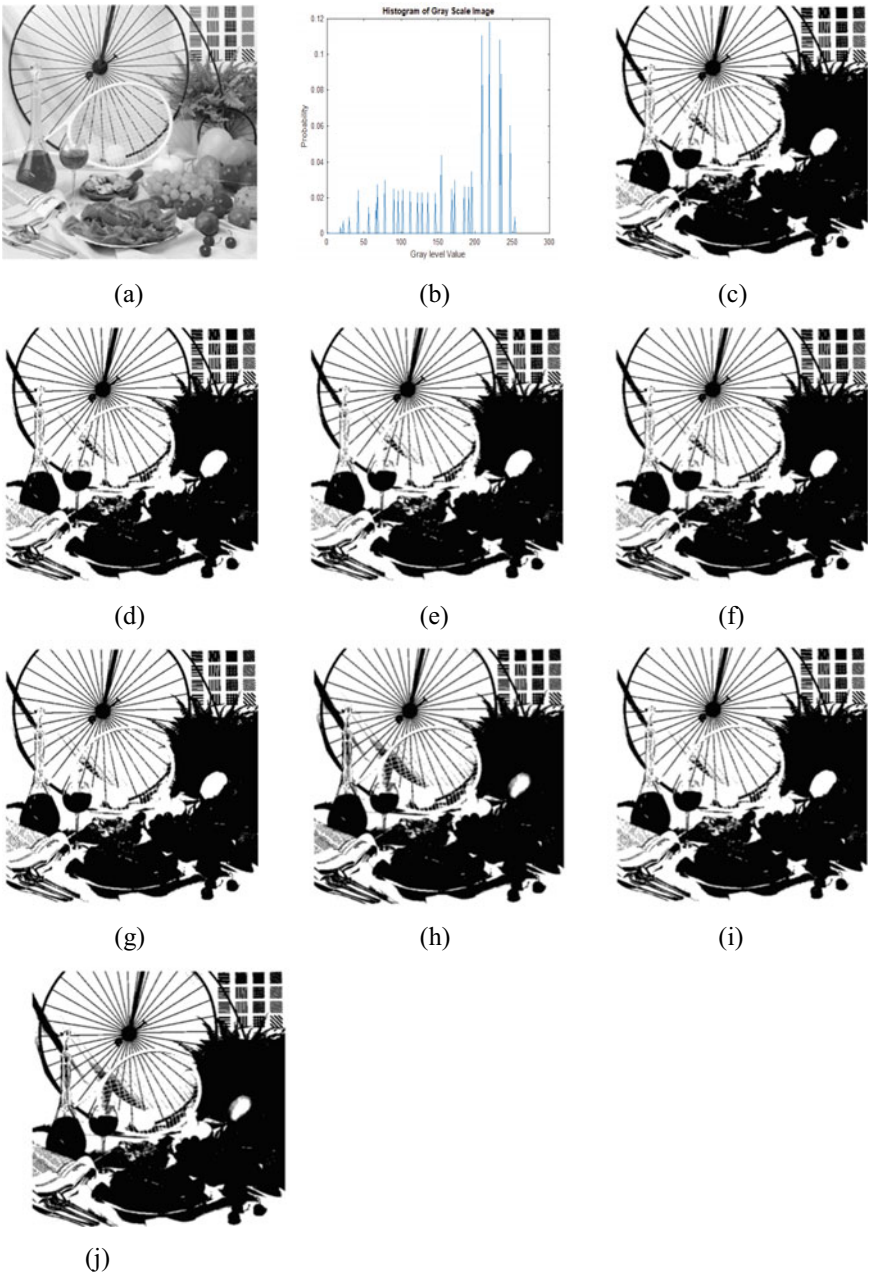


Fig. 6 "#7.gif (Dining Table) (512 × 512)" image **a** Original image; **b** Histogram; **c–j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

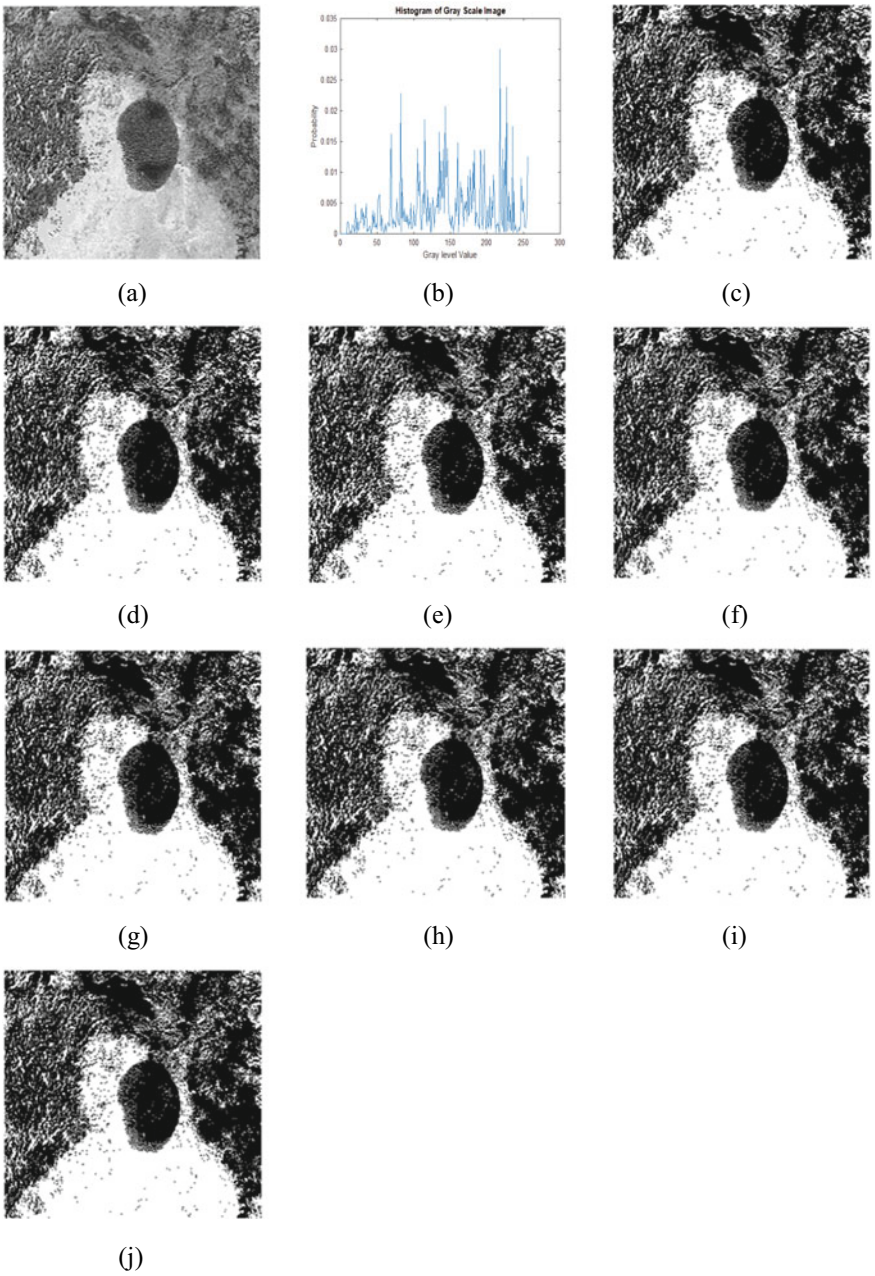


Fig. 7 "Shadow.tif (223 × 298)" image **a** Original image; **b** Histogram; **c–j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

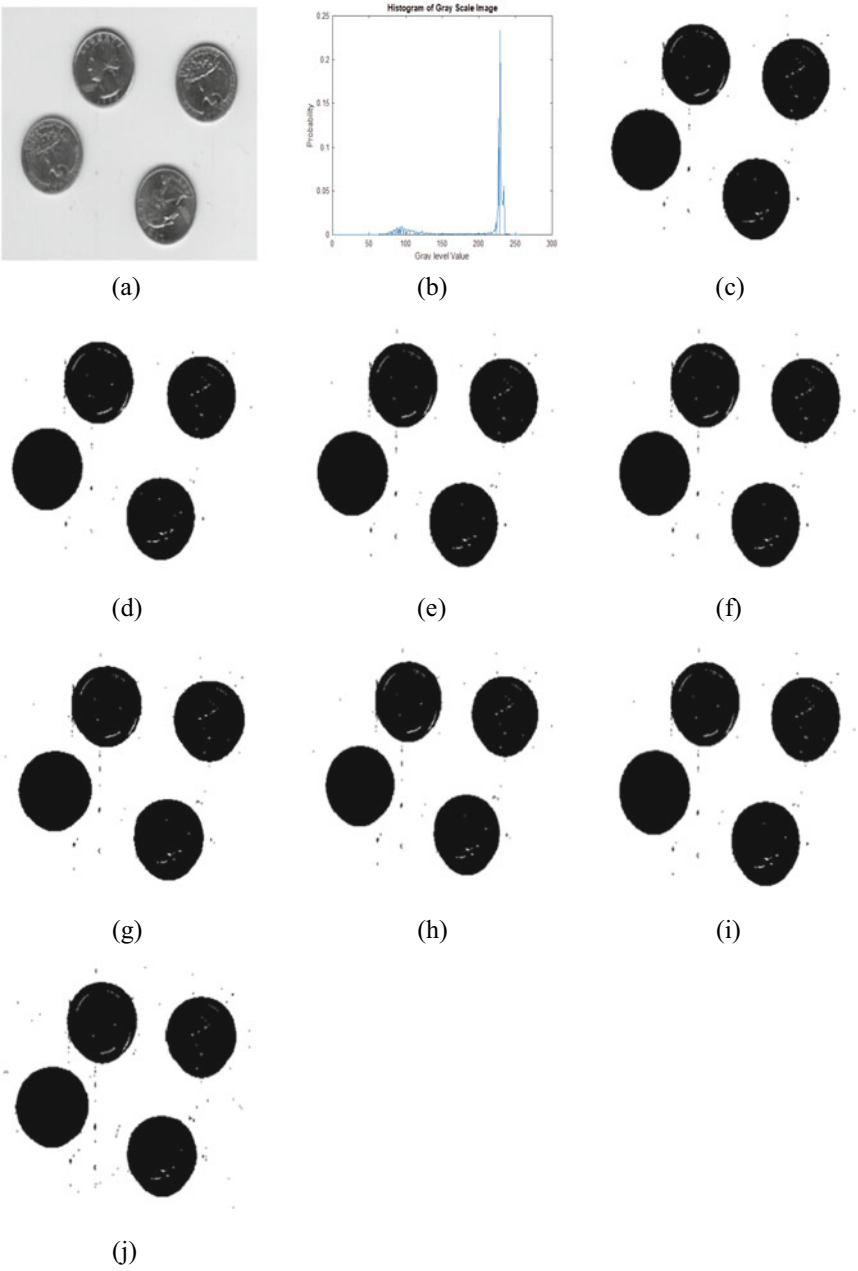


Fig. 8 "eight.tif (242 × 308)" image **a** Original image; **b** Histogram; **c**–**j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

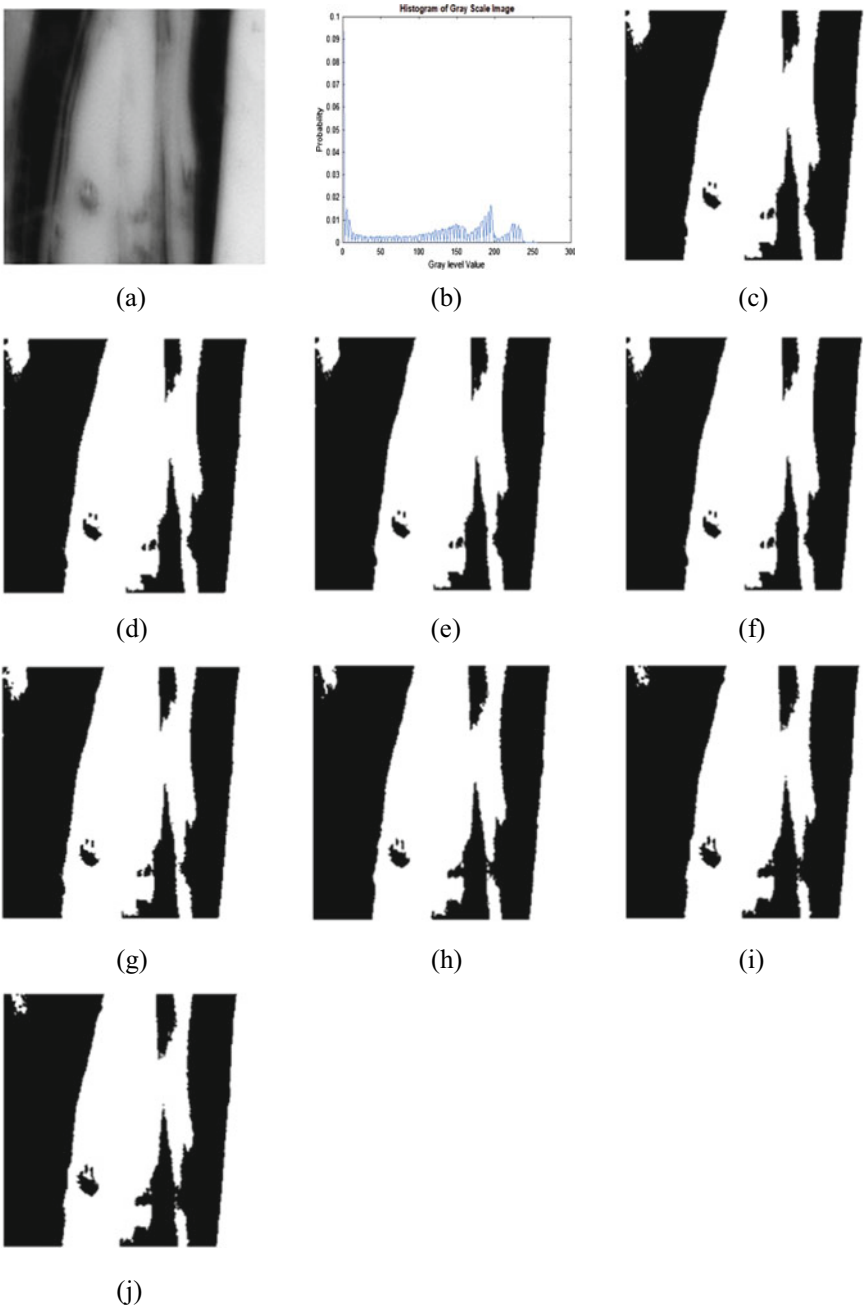


Fig. 9 "glass.png (181 × 282)" image **a** Original image; **b** Histogram; **c–j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

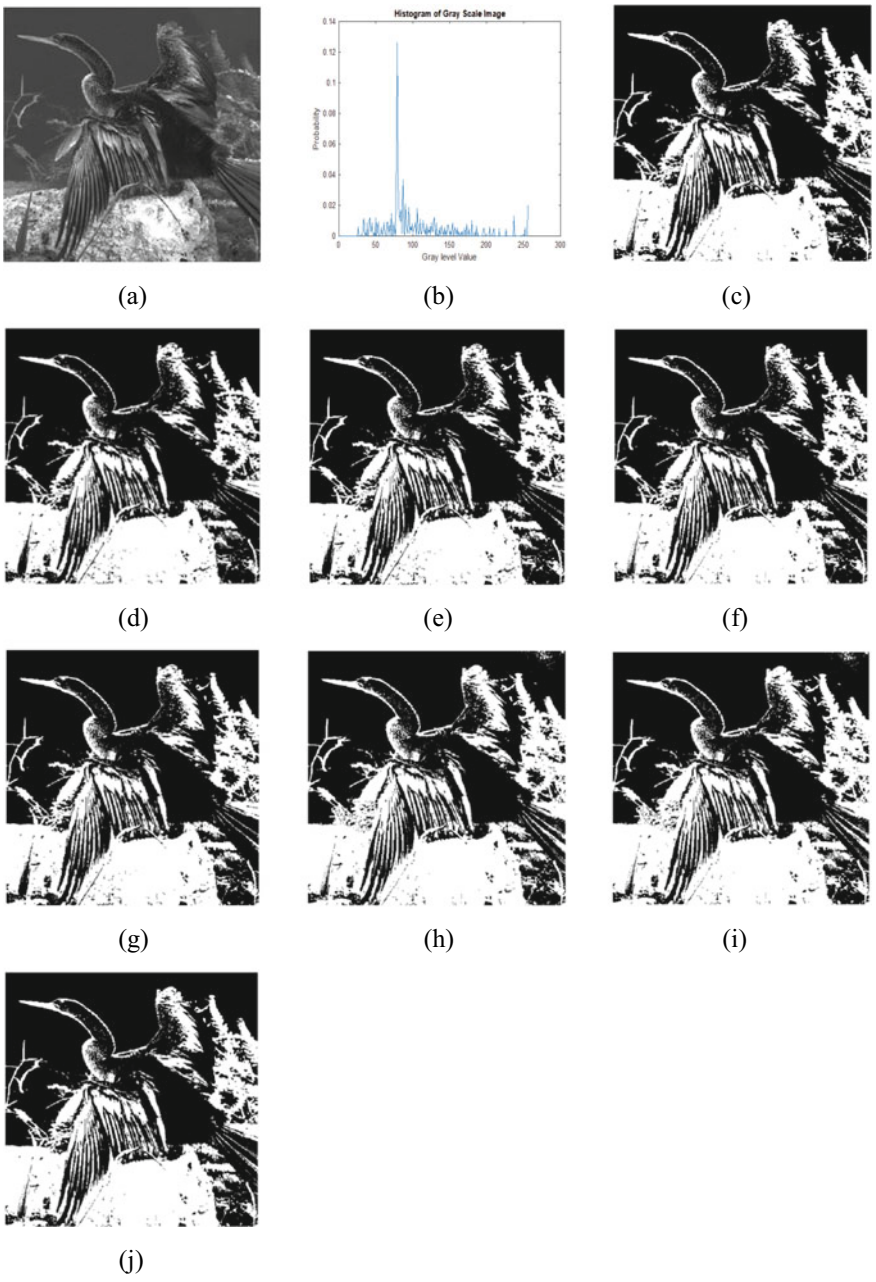


Fig. 10 "#9.gif (Bird Image) (512 × 512)" image **a** Original image; **b** Histogram; **c–j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

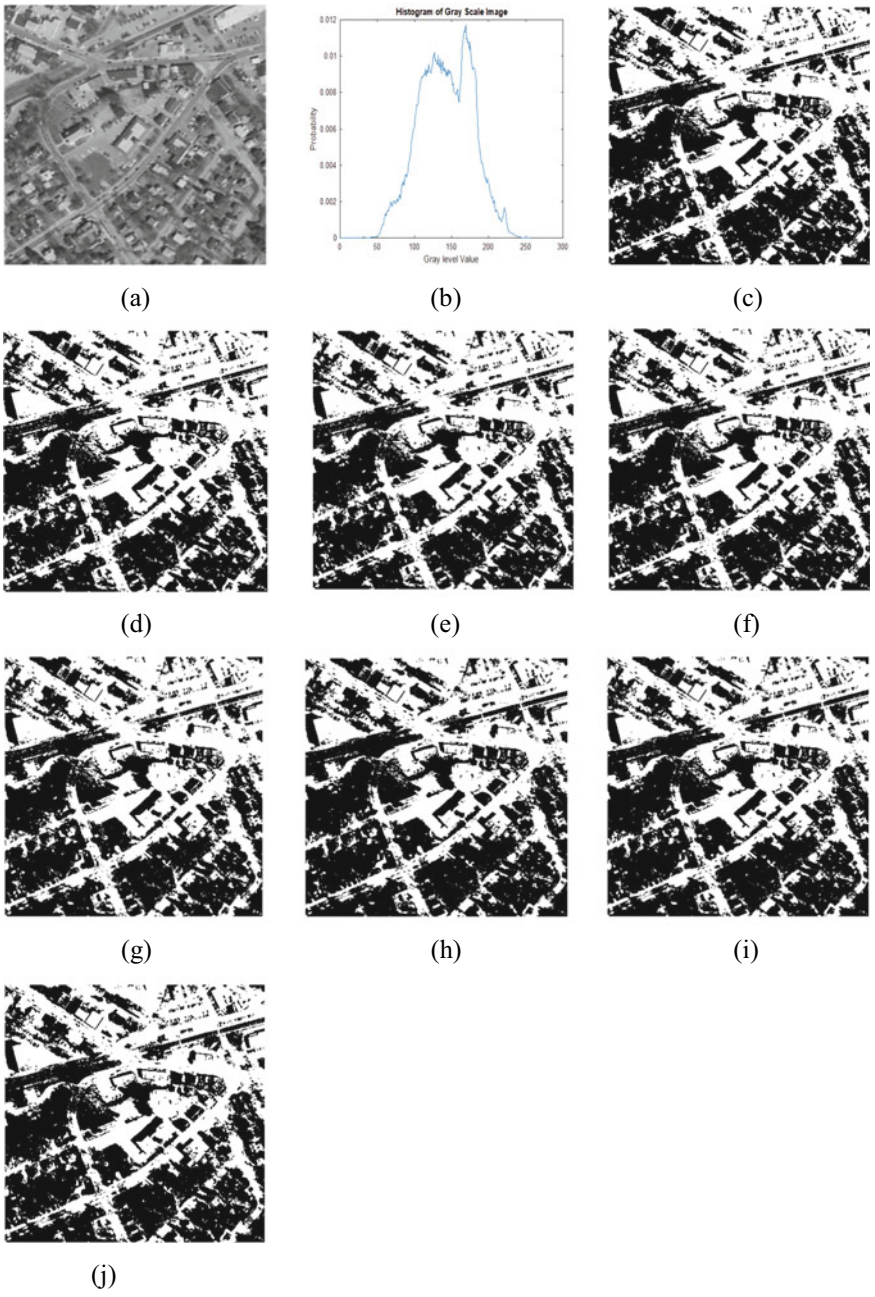


Fig. 11 "westconcordorthophoto.png (394 × 369)" image **a** Original image; **b** Histogram; **c**–**j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

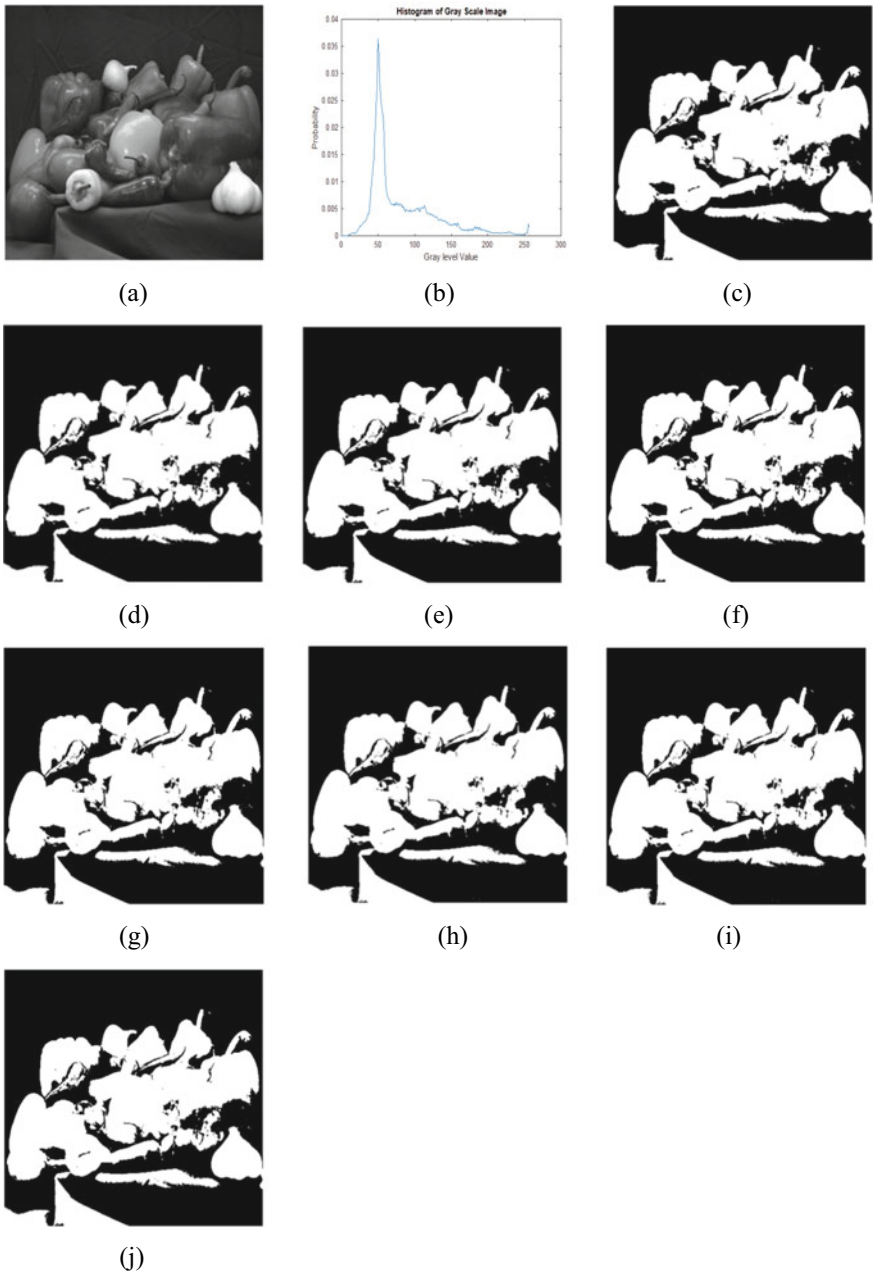


Fig. 12 "peppers.png (384 × 512)" image **a** Original image; **b** Histogram; **c–j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

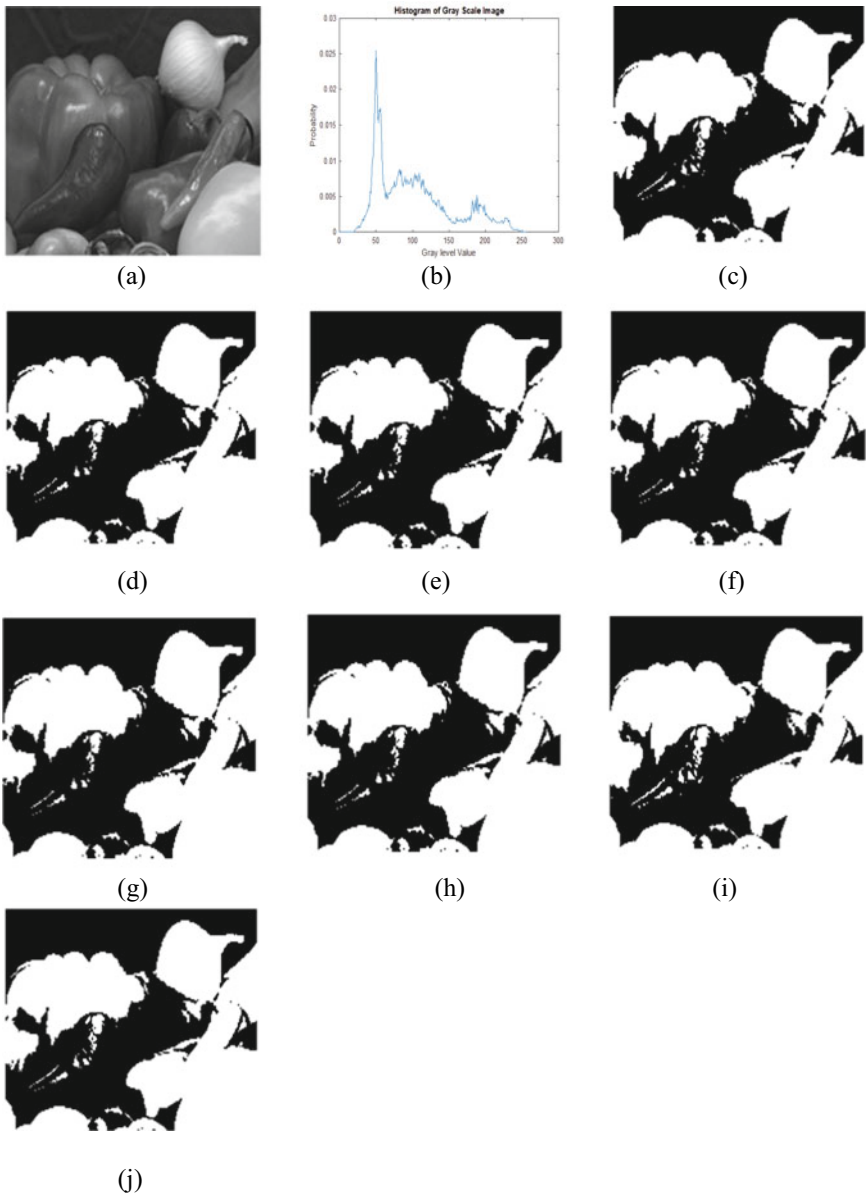


Fig. 13 "onion.png (135 × 198)" image **a** Original image; **b** Histogram; **c–j** Segmented image achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively

$$U = 1 - \frac{\left[2\sum_{k=0}^{1}\sum_{(i,j)\in R_k}\{g(i,j) - \mu_k\}^2\right]}{M * N * (g_{\max} - g_{\min})^2} \tag{40}$$

where $g(i, j)$ represents the intensity value of the pixel (i, j); R_k represents the kth segmented region; $g_{\max}$ and g_{min} represents the maximum and minimum intensity values of the image; μ_k represents the mean intensity value of pixels in the kth region and μ_k is defined as:

$$\mu_k = \frac{\sum_{(i,j)\in R_k} g(i,j)}{n_k} \tag{41}$$

where n_k represents the total number of pixels in R_k.

Structural Similarity (SSIM) Index is used to measure quality of segmented image based on the three terms: luminance, contrast and structural. It is defined as:

$$SSIM(g, Seg) = \frac{(2\mu_g\mu_{Seg} + C_1) * (2\sigma_{g,Seg} + C_2)}{(\mu_g^2 + \mu_{Seg}^2 + C_1) * (\sigma_g^2 + \sigma_{Seg}^2 + C_2)} \tag{42}$$

where, $\mu_g, \mu_{Seg}, \sigma_g, \sigma_{Seg}\, and\, \sigma_{g,Seg}$ are mean of input image, mean of segmented image, standard deviation of input image, standard of segmented image, cross-covariance for input and segmented images and constant parameters.

Mean of input image, mean of segmented image, standard deviation of input image, standard of segmented image, cross-covariance for input and segmented images are defined as:

$$\mu_g = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} g(i,j)}{M * N} \tag{43}$$

$$\mu_{Seg} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} Seg(i,j)}{M * N} \tag{44}$$

$$\sigma_g^2 = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left[g(i,j) - \mu_g\right]^2}{M * N - 1} \tag{45}$$

$$\sigma_{Seg}^2 = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left[Seg(i,j) - \mu_{Seg}\right]^2}{M * N - 1} \tag{46}$$

$$Cross\_Cov(g, Seg) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left[g(i,j) - \mu_g\right] * \left[Seg(i,j) - \mu_{Seg}\right]}{M * N - 1} \tag{47}$$

Mean Structural Similarity (MSSIM) Index is another measure to measure the quality of segmented that is defined as

$$MSSIM(g, Seg) = \frac{\sum_{i=1}^{M-(W_1-1)} \sum_{j=1}^{N-(W_2-1)} SSIM^{L}_{g,Seg}(i, j)}{[M-(W_1-1)] * [N-(W_2-1)} \quad (48)$$

where, $SSIM^{L}_{g,Seg}$ is local Structural Similarity (MSSIM) of input and segmented images and $W_1 \times W_2$ is size of window that is used for local definition. In the present experimental setup, the size of local window is set as 15×15.

Three different approaches i.e. Shannon fuzzy 2-partition entropy using GA, Shannon fuzzy 2-partition entropy using BBO and Shannon fuzzy 2-partition entropy using Recursive algorithm have been implemented for comparative investigation of the proposed approaches. GWO, GA and BBO approaches are metaheuristic techniques and these techniques require setting of some parameters for their execution. All the parameters required for these approaches are specified as below:

- The size of the population for all the approaches is set to as 10
- Number of iterations for GWO and GA based approaches is set to as 100 and 1000 to 2000 for BBO based approach
- Elitism parameter is set to as 1 for GA as well as BBO based approaches
- Mutation probability is set to as 0.15 for GA as well as BBO based approaches
- For GA based approach, crossover probability is set to as 0.75
- For BBO-based approach, maximum immigration and maximum emigration rate are set to as 1.

Entropy parameters of all the different measures of entropy used in the present work has been defined as follows:

- For Kapur fuzzy entropy values of $\alpha = 0.4$ and $\beta = 0.9$
- For Havrda-Charvat fuzzy entropy, Renyi fuzzy entropy and M. Masi fuzzy entropy value of $\alpha = 0.75$.

All the eight approaches are applied on ten standard test input images to find the optimal combination of values of the parameters a, b and c for each input image. Tables 2 and 3 demonstrate comparative analysis of the values of parameters (a, b, c) obtained through all the different algorithms for standard test images. These optimal combinations of value are further used to discover the optimal threshold value for each respective image. The optimal threshold values for all the standard test images obtained by different approaches are shown in the Table 4. Figures 4(c–j), 5(c–j), 6(c–j), 7(c–j), 8(c–j), 9(c–j), 10(c–j), 11(c–j), 12(c–j), 13(c–j) describe segmented images achieved by SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches.

Table 5 shows the comparative results of PSNR values achieved through SFE-Based-BBO, SFE-Based-GA, SFE-Based-Recursive approach and the five proposed approaches for all the ten segmented images. From Table 5, it is clearly evident that HCFE-Based-GWO has outperformed all other approaches in terms of achieving higher PSNR value. SFE-Based-GWO achieves the second place in term of overall performance based on PSNR values. RFE-Based-GWO performs better in eight

Table 2 Values of (a, b, c) obtained through all the different algorithms for test images

Image	SFE-Based-GWO			HCFE-Based-GWO			RFE-Based-GWO			KFE-Based-GWO		
	a	b	c	a	b	c	a	b	c	a	b	c
I_1	3	75	134	5	84	121	15	74	118	13	62	135
I_2	26	36	239	13	51	233	6	97	157	19	62	201
I_3	153	190	238	126	201	245	84	234	239	157	191	235
I_4	60	156	232	94	128	228	57	182	208	74	141	234
I_5	133	246	253	129	247	254	141	243	252	150	239	252
I_6	30	124	230	46	117	225	55	118	218	60	115	217
I_7	3	105	178	6	128	147	2	67	237	11	105	168
I_8	59	132	222	84	124	205	60	132	221	93	129	190
I_9	15	79	117	5	11	236	18	28	192	13	21	210
I_{10}	2	62	230	13	79	191	8	76	202	6	55	235

Table 3 Values of (a, b, c) obtained through all the different algorithms for test images

Image	MMFE-Based-GWO			SFE-Based-BBO			SFE-Based-GA			SFE-Based-Recursive		
	a	b	c	a	b	c	a	b	c	a	b	c
I_1	2	65	147	27	67	110	17	68	122	22	83	102
I_2	20	45	233	38	44	201	0	101	163	2	87	186
I_3	124	197	249	160	214	214	151	206	227	140	202	239
I_4	75	158	215	22	177	246	41	200	205	59	192	199
I_5	157	236	252	158	240	247	152	242	247	164	233	255
I_6	10	136	233	13	153	216	1	187	193	37	149	206
I_7	3	59	248	49	61	183	53	96	123	35	55	215
I_8	58	133	222	9	173	228	11	157	243	27	149	239
I_9	19	25	195	1	59	165	7	88	100	32	34	164
I_{10}	6	78	202	44	97	129	39	41	208	42	55	184

Table 4 Threshold values obtained through all the different algorithms for standard test images

Image	SFE-Based-GW0	HCFE-Based-GWO	RFE-Based-GWO	KFE-Based-GWO	MMFE-Based-GWO	SFE-Based-BB)	SFE-Based-GA	SFE-Based-Recursive
I_1	71.6731	72.6905	70.1226	68.2692	69.8962	67.7567	68.7553	71.3964
I_2	91.9643	91.5080	88.8885	88.5322	91.5009	87.8828	90.7276	90.5642
I_3	192.8336	192.8019	191.8193	193.5754	191.5463	198.1838	196.7165	195.3986
I_4	150.8625	146.1465	154.1468	147.7446	151.2234	153.7574	155.1841	155.4883
I_5	215.3408	214.8778	216.2396	217,3721	218.2577	218.4070	217.3835	220.0312
I_6	127.0437	126.6842	127.7226	127.5182	128.5285	132.2057	134.6263	134.2831
I_7	97.4722	98.7416	95.6671	96.9011	95.8405	92.5898	91.7943	95
I_8	136.3554	134.9964	136.3566	135.6079	136.5717	143.0075	141.1384	140.7190
I_9	72.1314	74.7936	72.5513	73.5577	72.6889	71.7691	68.3718	71.3717
I_{10}	91.6092	91.1601	91.4468	91.4382	91.7639	91.4605	89.2082	88.2973

Table 5 Comparative analysis of PSNR values resulting from all the different algorithms for standard test images

Image	SFE-Based-GW0	HCFE-Based-GWO	RFE-Based-GWO	KFE-Based-GWO	MMFE-Based-GWO	SFE-Based-BB)	SFE-Based-GA	SFE-Based-Recursive
I_1	8.2385	8.3204	8.1601	8.0685	8.1601	7.9684	8.0685	8.2385
I_2	11.0229	11.0229	11.0053	11.0053	11.0229	10.9986	11.0155	11.0155
I_3	9.0686	9.0686	9.0686	9.0686	9.0686	8.4797	8.4797	8.4797
I_4	9.3362	9.3880	9.2863	9.3677	9.3161	9.2952	9.2740	9.2740
I_5	11.1757	11.3353	11.0865	11.0865	10.9681	10.9681	11.0865	10.6442
I_6	10.8297	10.8297	10.8297	10.8297	10.8287	10.8108	10.7822	10.7822
I_7	8.9430	9.0084	8.8614	8.9130	8.8614	8.5798	8.5647	8.8466
I_8	9.3501	9.3788	9.3501	9.3659	9.3501	9.2085	9.2553	9.2775
I_9	8.3891	8.5430	8.3891	8.4633	8.3891	8.3120	8.0885	8.3120
I_{10}	8.6420	8.6420	8.6420	8.6420	8.6420	8.6420	8.5170	8.4451

images in each case as compared to SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches. KFE-Based-GWO performs better in nine images in case of SFE-Based-BBO, seven images in case of SFE-Based-GA and eight images in case of SFE-Based-Recursive. MMFE-Based-GWO performs better in eight images as compared to SFE-Based-BBO and nine images in each case as compared to SFE-Based-GA and SFE-Based-Recursive approaches.

Table 6 shows the comparative results of uniformity values achieved through SFE-Based-BBO, SFE-Based-GA, SFE-Based-Recursive approach and the five proposed approaches for all the ten segmented images. From Table 6, it is clearly evident that HCFE-Based-GWO has outperformed all other approaches in terms of achieving higher Uniformity value. SFE-Based-GWO achieves the second place in term of overall performance based on Uniformity values. RFE-Based-GWO performs better in eight images as compared to SFE-Based-BBO and six images in each case as compared to SFE-Based-GA and SFE-Based-Recursive approaches. KFE-Based-GWO performs better in eight images in case of SFE-Based-BBO, six images in case of SFE-Based-GA and seven images in case of SFE-Based-Recursive. MMFE-Based-GWO performs better in seven images as compared to SFE-Based-BBO and eight images in each case as compared to SFE-Based-GA and SFE-Based-Recursive approaches.

Table 7 shows the comparative results of SSIM values achieved through SFE-Based-BBO, SFE-Based-GA, SFE-Based-Recursive approach and the five proposed approaches for all the ten segmented images. From Table 6, it is clearly evident that HCFE-Based-GWO has outperformed all other approaches in terms of achieving higher SSIM value. SFE-Based-GWO achieves the second place in term of overall performance based on SSIM values. RFE-Based-GWO performs better in eight images in each case as compared to SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches. KFE-Based-GWO performs better in nine, seven and eight images as compare to SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively. MMFE-Based-GWO performs better in eight images as compared to SFE-Based-BBO and nine images in each case as compared to SFE-Based-GA and SFE-Based-Recursive approaches.

Table 8 shows the comparative results of MSSIM values achieved through SFE-Based-BBO, SFE-Based-GA, SFE-Based-Recursive approach and the five proposed approaches for all the ten segmented images. From Table 7, it is clearly evident that HCFE-Based-GWO has outperformed all other approaches in terms of achieving higher MSSIM value. SFE-Based-GWO achieves the second place in term of overall performance based on MSSIM values. RFE-Based-GWO performs better in eight, seven and six images as compared to SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively. KFE-Based-GWO performs better in nine, six and seven images as compare to SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches respectively. MMFE-Based-GWO performs better in eight images in each case as compared to SFE-Based-BBO and SFE-Based-GA; and seven images compared to SFE-Based-Recursive approaches.

From above discussion, it is clearly evident that HCFE-Based-GWO has outperformed all other approaches in terms of achieving higher PSNR, uniformity, SSIM

Table 6 Comparative analysis of Uniformity values resulting from all the different algorithms for standard test images

Image	SFE-Based-GWO	HCFE-Based-GWO	RFE-Based-GWO	KFE-Based-GWO	MMFE-Based-GWO	SFE-Based-BBO	SFE-Based-GA	SFE-Based-Recursive
I_1	0.9676	0.9681	0.9671	0.9664	0.9671	0.9656	0.9664	0.9676
I_2	0.9881	0.9881	0.9878	0.9878	0.9881	0.9877	0.9880	0.9880
I_3	0.9631	0.9631	0.9631	0.9631	0.9631	0.9580	0.9580	0.9580
I_4	0.9617	0.9617	0.9616	0.9617	0.9617	0.9617	0.9616	0.9616
I_5	0.9771	0.9790	0.9761	0.9761	0.9747	0.9747	0.9761	0.9707
I_6	0.9506	0.9506	0.9506	0.9506	0.9499	0.9474	0.9456	0.9456
I_7	0.9631	0.9638	0.9621	0.9627	0.9621	0.9590	0.9588	0.9620
I_8	0.9695	0.9695	0.9695	0.9695	0.9695	0.9694	0.9695	0.9695
I_9	0.9735	0.9744	0.9735	0.9739	0.9735	0.9731	0.9717	0.9731
I_{10}	0.9627	0.9627	0.9627	0.9627	0.9627	0.9627	0.9618	0.9612

Table 7 Comparative analysis of SSIM values resulting from all the different algorithms for standard test images

Image	SFE-Based-GWO	HCFE-Based-GWO	RFE-Based-GWO	KFE-Based-GWO	MMFE-Based-GWO	SFE-Based-BBO	SFE-Based-GA	SFE-Based-Recursive
I_1	0.5146	0.5192	0.5102	0.5050	0.5102	0.4993	0.5050	0.5146
I_2	0.7102	0.7102	0.7093	0.7093	0.7102	0.7090	0.7099	0.7099
I_3	0.6534	0.6534	0.6534	0.6534	0.6534	0.6224	0.6224	0.6224
I_4	0.6440	0.6455	0.6421	0.6450	0.6432	0.6424	0.6415	0.6415
I_5	0.6964	0.7041	0.6921	0.6921	0.6862	0.6862	0.6921	0.6696
I_6	0.7651	0.7651	0.7651	0.7651	0.7643	0.7607	0.7577	0.7577
I_7	0.5375	0.5419	0.5318	0.5354	0.5318	0.5115	0.5103	0.5308
I_8	0.6130	0.6135	0.6130	0.6133	0.6130	0.6081	0.6100	0.6108
I_9	0.4961	0.5061	0.4961	0.5010	0.4961	0.4911	0.4762	0.4911
I_{10}	0.5480	0.5480	0.5480	0.5480	0.5480	0.5480	0.5407	0.5365

Table 8 Comparative analysis of MSSIM values resulting from all the different algorithms for standard test images

Image	SFE-Based-GWO	HCFE-Based-GWO	RFE-Based-GWO	KFE-Based-GWO	MMFE-Based-GWO	SFE-Based-BBO	SFE-Based-GA	SFE-Based-Recursive
I_1	0.2422	0.2427	0.2412	0.2399	0.2412	0.2382	0.2399	0.2422
I_2	0.2343	0.2343	0.2333	0.2333	0.2343	0.2330	0.2339	0.2339
I_3	0.3793	0.3793	0.3793	0.3793	0.3793	0.3540	0.3540	0.3540
I_4	0.4067	0.4145	0.4018	0.4097	0.4046	0.4026	0.4007	0.4007
I_5	0.2884	0.2943	0.2855	0.2855	0.2812	0.2812	0.2855	0.2631
I_6	0.1323	0.1323	0.1323	0.1323	0.1315	0.1276	0.1246	0.1246
I_7	0.1968	0.1974	0.1896	0.1947	0.1895	0.1878	0.1875	0.1896
I_8	0.4177	0.4220	0.4177	0.4200	0.4177	0.3999	0.4053	0.4081
I_9	0.1173	0.1191	0.1173	0.1182	0.1173	0.1160	0.1129	0.1160
I_{10}	0.1976	0.1976	0.1976	0.1976	0.1976	0.1976	0.1976	0.1976

and MSSIM values for each segmented image. Thus, the proposed HCFE-Based-GWO approach has the more ability to produce quality segmented image as compared to SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches as well as other four proposed approaches. SFE-Based-GWO has achieved second place in term of overall performance point of view after HCFE-Based-GWO. The performance of other three proposed approaches i.e. RFE-Based-GWO, KFE-Based-GWO and MMFE-Based-GWO is satisfactory as compare to SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches.

Moreover from computational complexity and number iteration point of view, the proposed Havrda-Charvat Fuzzy Entropy-Based-GWO (HCFE-Based-GWO) approach is also providing segmented images in less time than that of provided by SFE-Based-BBO, SFE-Based-GA, SFE-Based-Recursive approaches and other four proposed approaches as concluded from Tables 9, 10 to Figs. 14, 15, 16, 17, 18, 19, 20, 21, 22 and 23.

6 Conclusions

Grey Wolf Optimization has been proposed to find optimal threshold values for image segmentation. Five different objective functions have been designed to check the performance of the proposed method. These objective functions are based on Shannon as well as non-Shannon measures of fuzzy entropy. Non-Shannon measures of fuzzy entropy are Renyi fuzzy entropy, Havrda-Charvat fuzzy entropy, Kapur fuzzy entropy and M. Masi fuzzy entropy. The performance evaluation of all the five approaches i.e. SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO and MMFE-Based-GWO is done against two well-known metaheuristics techniques and one non-metaheuristics approach available in the literature. Metaheuristics techniques selected for comparison are Shannon fuzzy entropy using GA and BBO whereas non-metaheuristics approach selected for comparison is Shannon fuzzy entropy using Recursive approaches. The performance is evaluated using a set of ten different kinds of images that has variant characteristics. PSNR, uniformity, SSIM and MSSIM are four measures that have been used to check the efficiency of the proposed approaches. HCFE-Based-GWO has outperformed all other approaches in terms of achieving higher PSNR, uniformity, SSIM and MSSIM values for each segmented image. From experimental results, it is concluded that the proposed Havrda-Charvat Fuzzy Entropy-Based-GWO (HCFE-Based-GWO) approach has the more ability to produce quality segmented image as compared to SFE-Based-BBO, SFE-Based-GA and SFE-Based-Recursive approaches as well as other four proposed approaches. Shannon Fuzzy Entropy-Based-GWO (SFE-Based-GWO) has achieved second place in term of overall performance point of view after HCFE-Based-GWO. From computational complexity and number iteration point of view, the proposed Havrda-Charvat Fuzzy Entropy-Based-GWO (HCFE-Based-GWO) approach is also providing segmented images in less time than all other approaches.

Table 9 Computation time taken by all the different algorithms to segment standard test images

Image	SFE-Based-GWO	HCFE-Based-GWO	RFE-Based-GWO	KFE-Based-GWO	MMFE-Based-GWO	SFE-Based-BBO	SFE-Based-GA	SFE-Based-Recursive
I_1	0.3718	0.3028	0.4912	0.4288	0.3318	2.4382	0.3919	1.5031
I_2	0.3615	0.2924	0.3714	0.4808	0.4696	2.2009	0.6188	1.6691
I_3	1.3954	1.3381	1.5362	1.6795	1.5678	12.2871	1.2437	2.2503
I_4	0.4102	0.2968	0.4267	0.4024	0.4515	5.6082	0.4016	1.5671
I_5	0.4718	0.3035	0.4768	0.4934	0.4363	3.1720	0.3576	1.8175
I_6	0.3345	0.3168	0.3759	0. 3915	0.3778	2.6813	0.3836	1.5327
I_7	0.6983	0.6578	1.2564	1.5573	1.2975	10.4987	1.2854	2.4749
I_8	0.5281	0.4238	0.4975	0.5903	0.5678	3.9874	0.5766	1.7111
I_9	1.1091	0.9853	1.0984	1.1475	1.0975	10.2948	1.2676	1.8290
I_{10}	0.4632	0.4498	0.4893	0.5931	0.6256	4.5587	0.5317	2.2136

Table 10 Number of iterations taken by all the different Metaheuristic approaches to segment standard test images

Image	SFE-Based-GWO	HCFE-Based-GWO	RFE-Based-GWO	KFE-Based-GWO	MMFE-Based-GWO	SFE-Based-BBO	SFE-Based-GA
I_1	23	10	40	27	14	584	28
I_2	20	8	27	38	40	166	40
I_3	20	19	23	28	28	964	12
I_4	17	7	21	11	59	1545	14
I_5	84	7	83	73	61	540	21
I_6	10	6	27	25	26	321	25
I_7	7	5	12	23	15	444	13
I_8	12	5	10	17	17	562	16
I_9	22	15	25	17	19	668	41
I_{10}	12	10	15	18	26	264	14

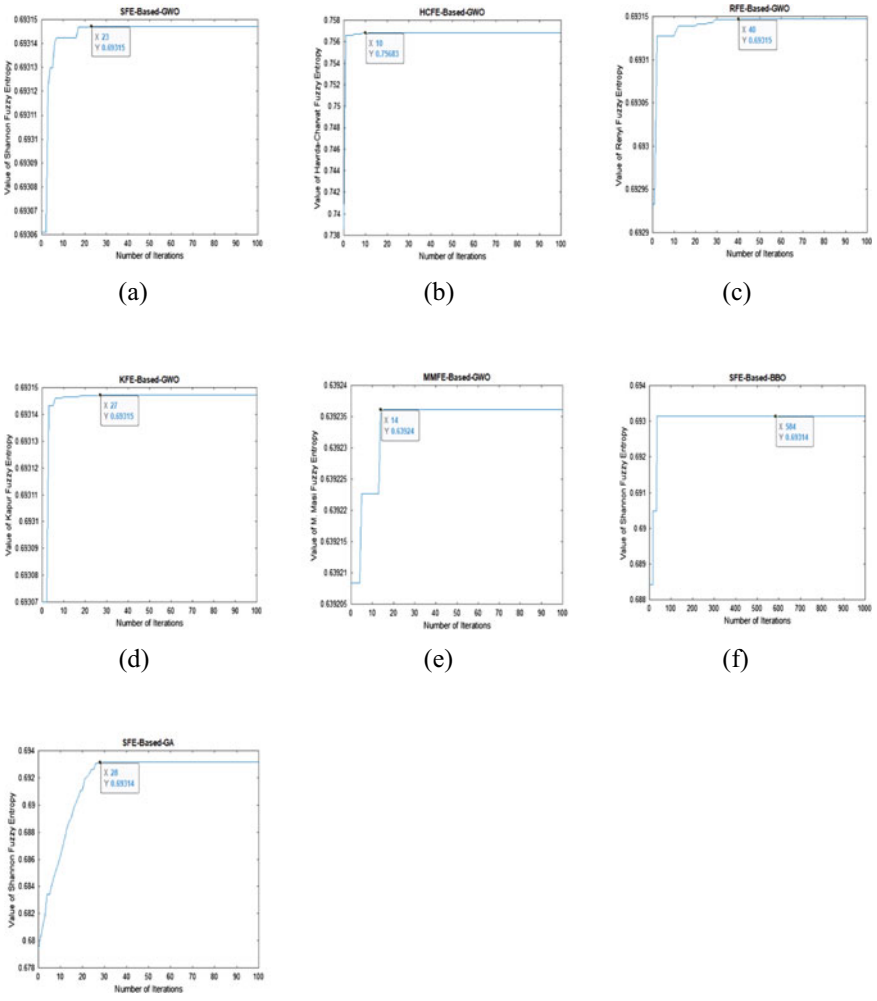


Fig. 14 Convergence rate of "circuit.tif (280 × 272)" image **a**–**g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively

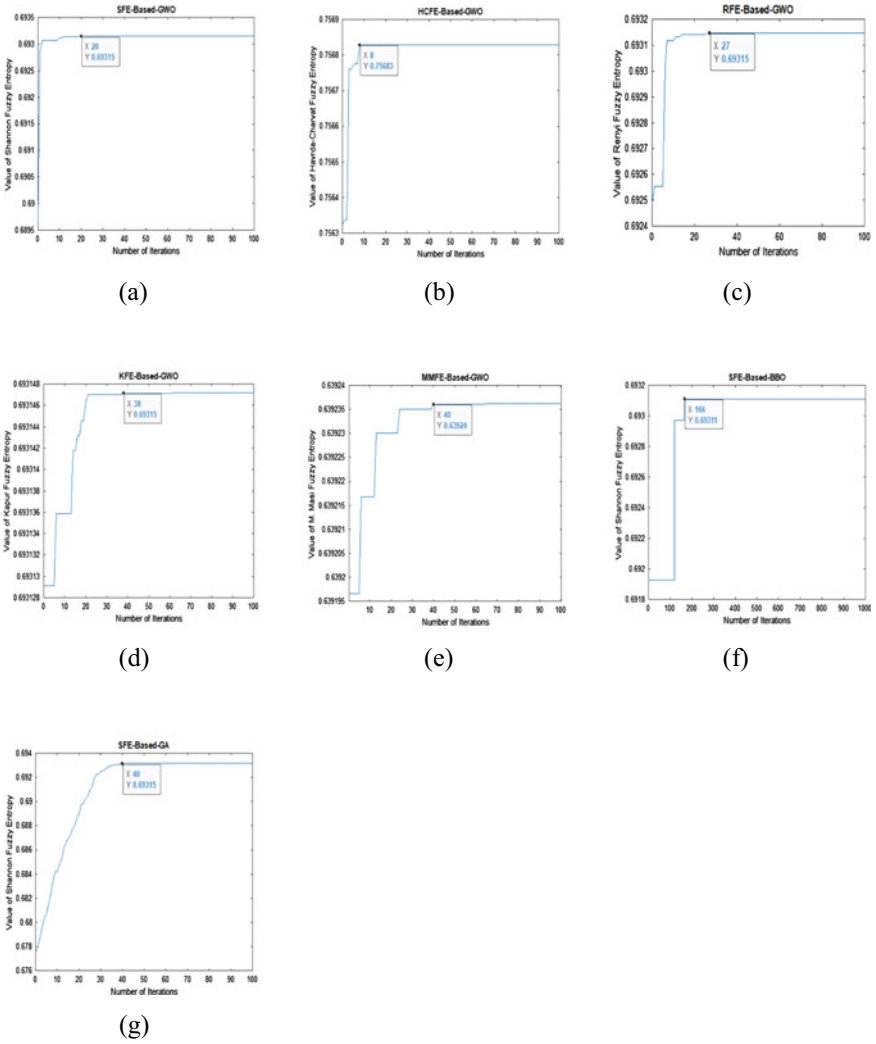


Fig. 15 Convergence rate of "coins.png (246 × 300)" image **a**–**g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively

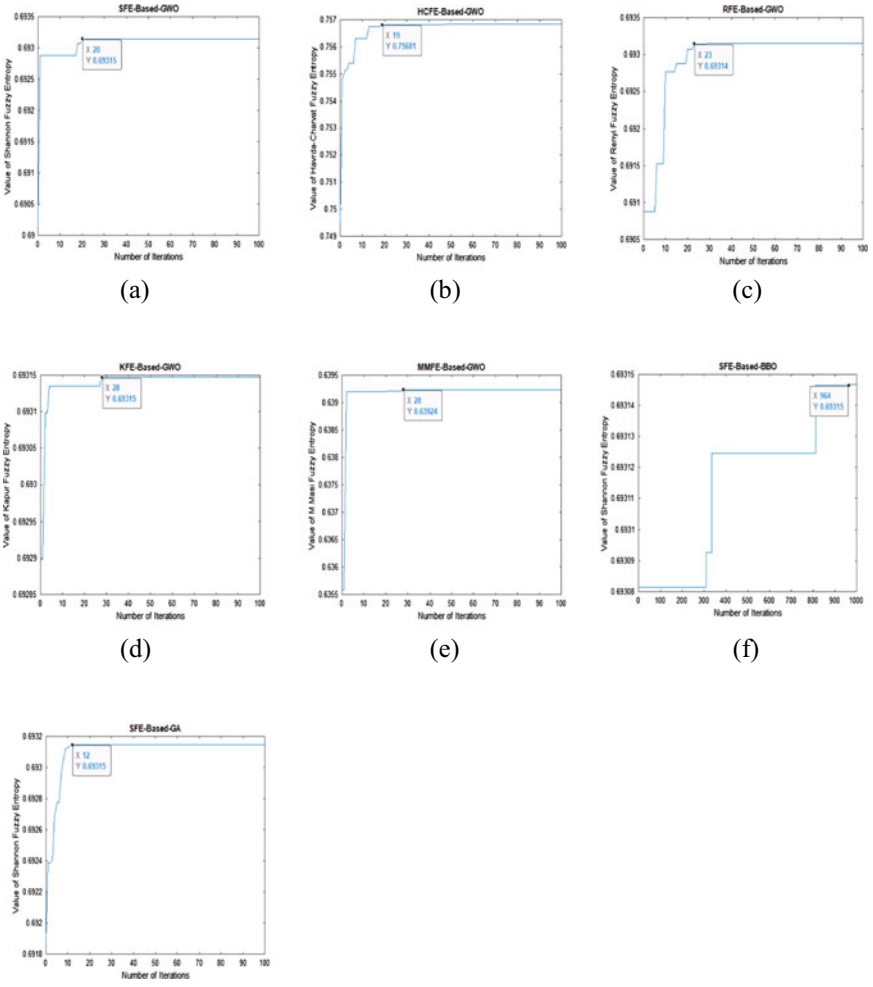


Fig. 16 Convergence rate of "#7.gif (Dining Table) (512 × 512)" image **a**–**g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively

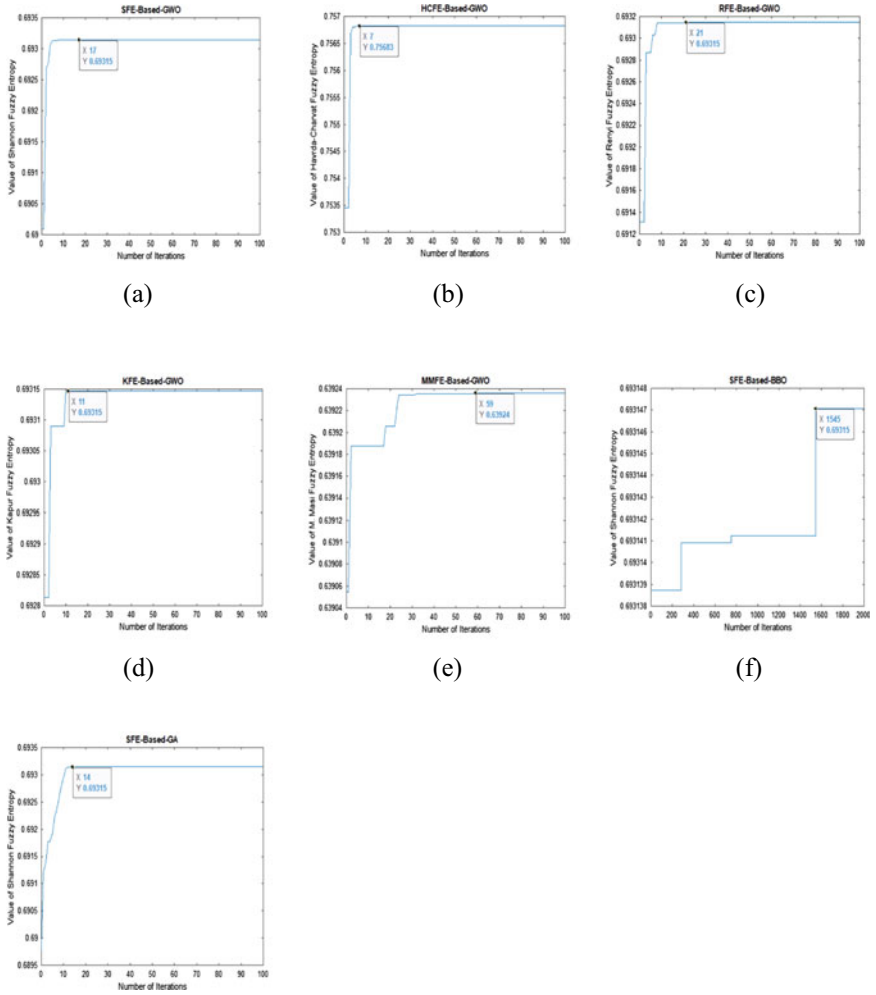


Fig. 17 Convergence rate of "Shadow.tif (223 × 298)" image **a–g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively

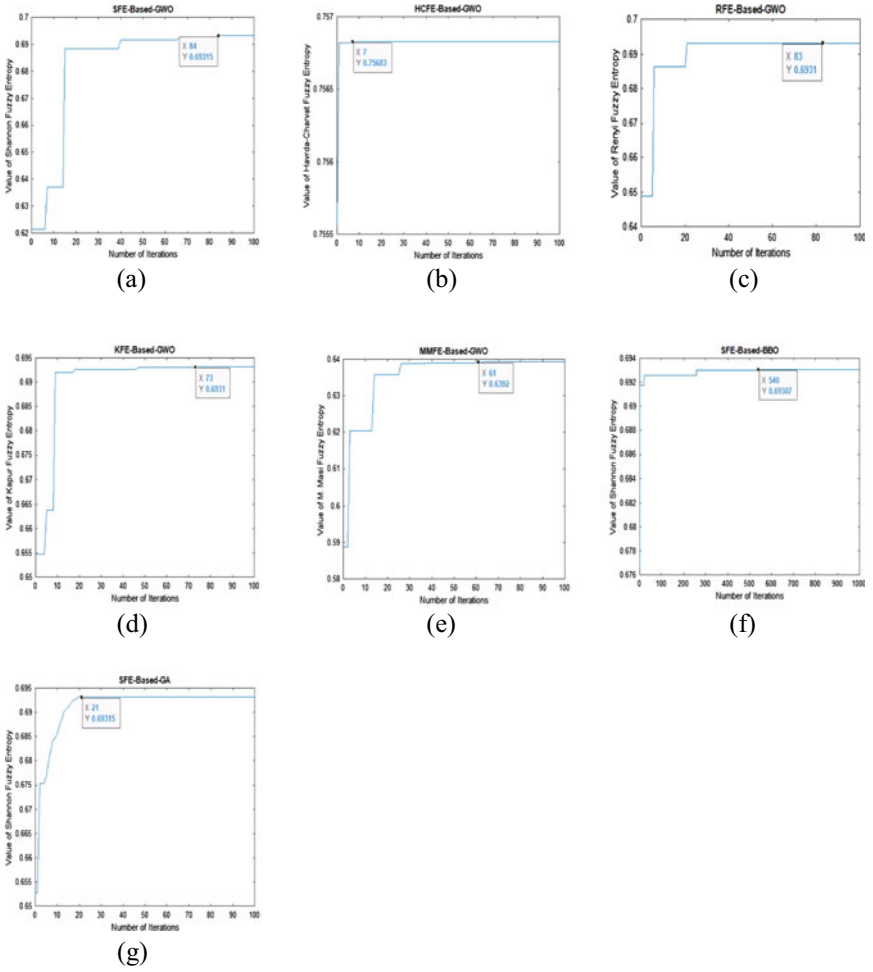


Fig. 18 Convergence rate of "eight.tif (242 × 308)" image **a**–**g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively

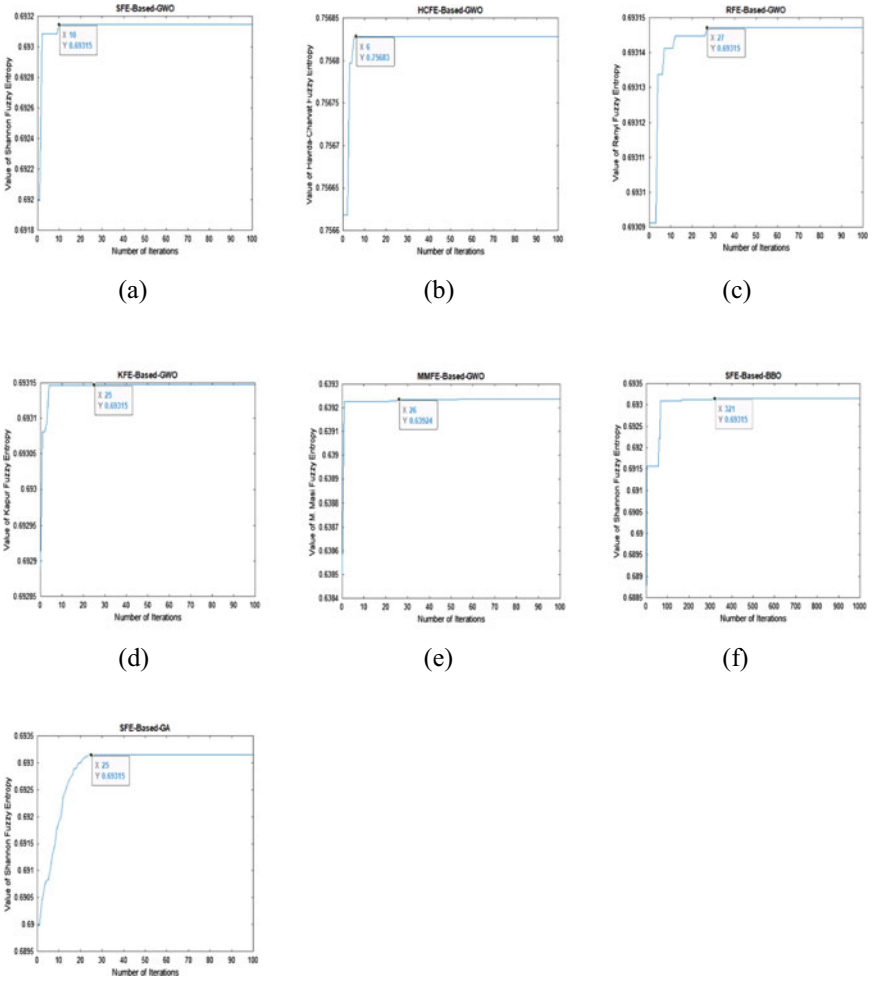
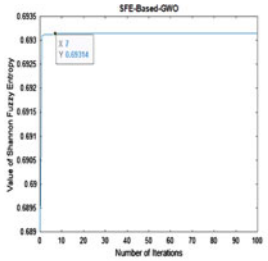
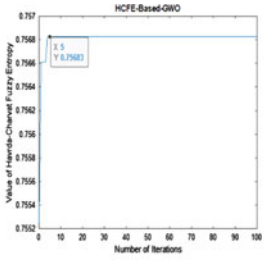


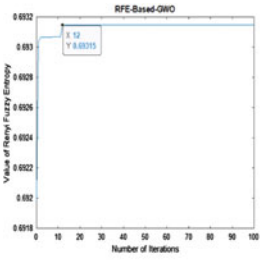
Fig. 19 Convergence rate of "glass.png (181 × 282)" image **a**–**g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively



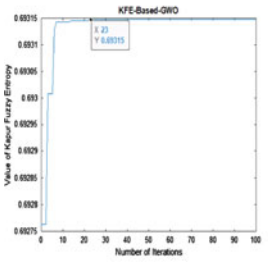
(a)



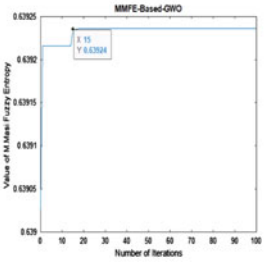
(b)



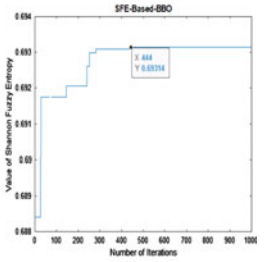
(c)



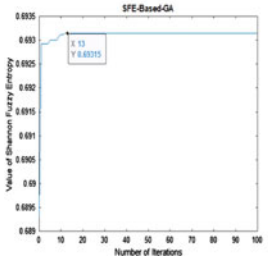
(d)



(e)



(f)



(g)

Fig. 20 Convergence rate of "#9.gif (Bird Image) (512 × 512)" image **a**–**g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively

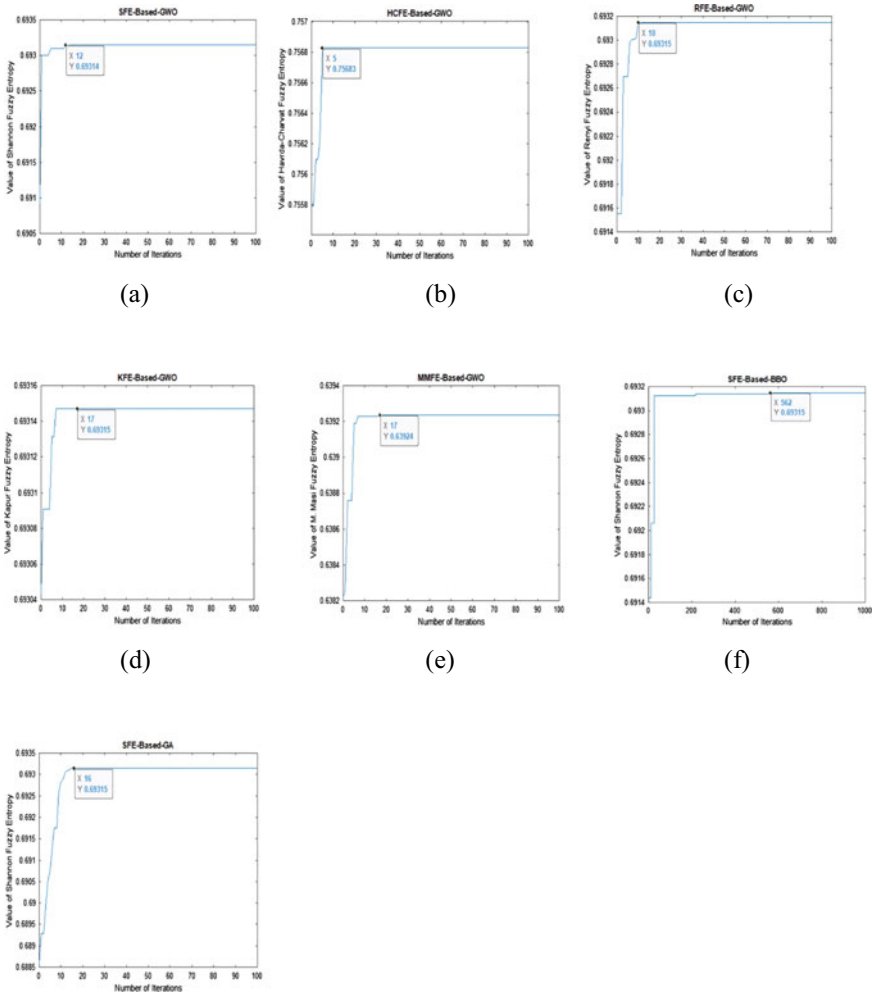


Fig. 21 Convergence rate of "westconcordorthophoto.png (394 × 369)" image **a**–**g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively

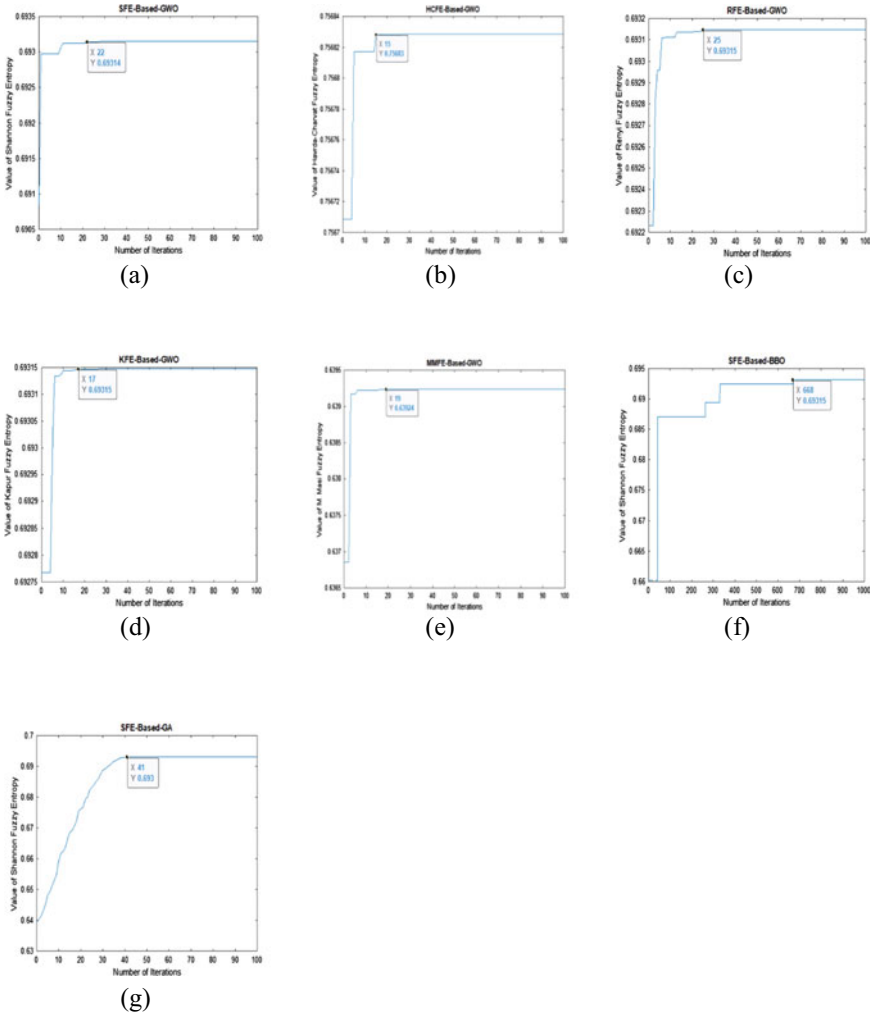


Fig. 22 Convergence rate of "peppers.png (384 × 512)" image **a**–**g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively

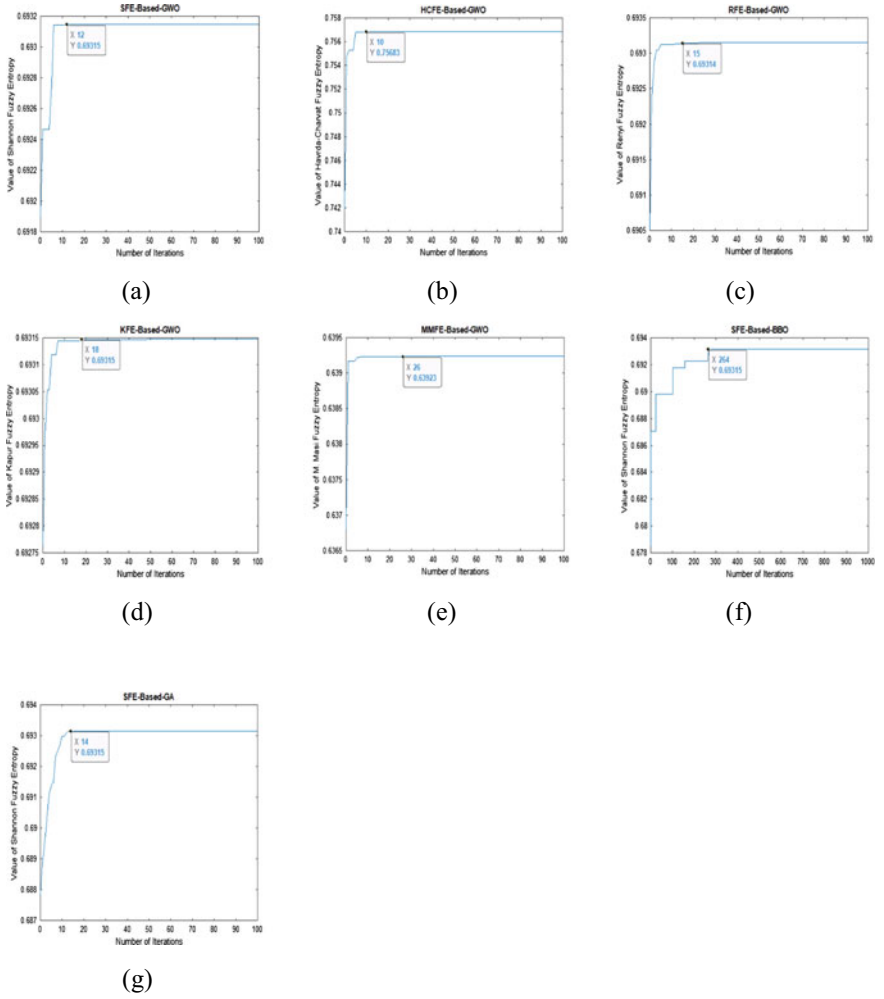


Fig. 23 Convergence rate of "onion.png (135 × 198)" image **a**–**g** SFE-Based-GWO, HCFE-Based-GWO, RFE-Based-GWO, KFE-Based-GWO, MMFE-Based-GWO, SFE-Based-BBO and SFE-Based-GA approaches respectively

References

1. Baby Resma, K.P., Nair, M.S.: Multilevel thresholding for image segmentation using Krill Herd optimization algorithm. J. King Saud Univ.-Comput. Inf. Sci. **33**(5), 528–541 (2021)
2. Naidu, M.S.R., Kumar, P.K., Chiranjeevi, K.: Shannon and fuzzy entropy based evolutionary image thresholding for image segmentation. Alex. Eng. J. **57**(3), 1643–1655 (2018)
3. Liao, P.S., Chen, T.S., Chung, P.C.: A fast algorithm for multilevel thresholding. J. Inf. Sci. Eng. **17**(5), 713–727 (2001)
4. Otsu, N.: A threshold selection from gray level histograms. IEEE Trans. Syst. Man Cybern.: Syst. **9**(1), 62–66 (1979)

5. Pun, T.: Entropies thresholding, a new approach. Comput. Vis. Graph. Image Process. **16**(2), 210–239 (1981)
6. Kapur, J.N., Sahoo, P.K., Wong, A.K.C.: A new method for gray level picture thresholding using the entropy of the histogram. Comput. Vis. Graph. Image Process. **29**, 273–285 (1985)
7. Sahoo, P., Wilkins, C., Yeager, J.: Threshold selection using Renyi's entropy. Pattern Recogn. **30**(1), 71–84 (1997)
8. Cheng, H.D., Cheng, J.R.C., Li, J.: Threshold selection based on fuzzy c-partition entropy approach. Pattern Recogn. **31**(7), 857–870 (1998)
9. de Albuquerque, M.P., Esquef, I.A., Mello, A.R.G.: Image thresholding using Tsallis-entropy. Pattern Recogn. Lett. **25**(9), 1059–1065 (2004)
10. Shitong, W., Chung, F.L.: Note on the equivalence relationship between Renyi-entropy and Tsallis-entropy based thresholding. Pattern Recogn. Lett. **26**(14), 2309–2312 (2005)
11. Benabdelkader, S., Boulemden, M.: Recursive algorithm based on fuzzy 2-partition entropy for 2-level image thresholding. Pattern Recogn. **38**(8), 1289–1294 (2005)
12. Tang, Y., Weiwei, M., Zhang, Y., Zhang, X.: A fast recursive algorithm based on fuzzy 2-partition entropy approach for threshold selection. Neuro Comput. **74**(8), 3072–3078 (2011)
13. Wang, Y.: Improved OTSU and adaptive genetic algorithm for infrared image segmentation. In: Proceedings of IEEE Conference on Chinese Control and Decision, Shenyang, China, pp. 5644–5648 (2018)
14. Pare, S., Kumar, A., Bajaj, V., Singh, G.K.: A multilevel color image segmentation technique based on cuckoo search algorithm and energy curve. Appl. Soft Comput. **47**, 76–102 (2016)
15. Suresh, S., Lal, S.: An efficient cuckoo search algorithm based multilevel thresholding for segmentation of satellite images using different objective functions. Expert Syst. Appl. **58**, 184–209 (2016)
16. Pan, Y., Xia, Y., Zhou, T., Fulham, M.: Cell image segmentation using bacterial foraging optimization. Appl. Soft Comput. **58**, 770–782 (2017)
17. Aziz, M.A.E.A., Ewees, A.A., Hassanien, A.E.: Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. Expert Syst. Appl. **83**, 242–256 (2017)
18. Arnay, R., Fumero, F., Sigut, J.: Ant colony optimization-based method for optic cup segmentation in retinal images. Appl. Soft Comput. **52**, 409–417 (2017)
19. Jiang, Y., Tsai, P., Yeh, W.C., Cao, Y.: A honey-bee-mating based algorithm for multilevel image segmentation using Bayesian theorem. Appl. Soft Comput. **52**, 1181–1190 (2017)
20. Huang, L., He, D., Yang, S.X.: Segmentation on ripe fuji apple with fuzzy 2D entropy based on 2D histogram and GA optimization. Intell. Autom. Soft Comput. **19**(3), 239–251 (2013)
21. Khehra, B.S., Pharwaha, A.P.S., Kaushal, M.: Fuzzy 2-partition entropy threshold selection based on big bang-big crunch optimization algorithm. Egypt. Inform. J. **16**(1), 133–150 (2015)
22. Tosta, T.A.A., Faria, P.R., Batista, V.R., Neves, L.A., Nascimento, M.Z.D.: Using wavelet sub-band and fuzzy2-partitionentropy to segment chronic lymphocytic leukaemia images. Appl. Soft Comput. **64**, 49–58 (2018)
23. Pare, S., Bhandari, A.K., Kumar, A., Singh, G.K.: A new technique for multilevel color image thresholding based on modified fuzzy entropy and Lévy flight firefly algorithm. Comput. Electr. Eng. **70**, 476–495 (2018)
24. Tao, W., Jin, H., Liu, L.: Object segmentation using ant colony optimization algorithm and fuzzy entropy. Pattern Recogn. Lett. **28**(7), 788–796 (2007)
25. Chatterjee, A., Siarry, P., Nakib, A., Blanc, R.: An improved biogeography based optimization approach for segmentation of human head CT-scan images employing fuzzy entropy. Eng. Appl. Artif. Intell. **25**(8), 1698–1709 (2012)
26. Khehra, B.S., Singh, A.P.: Image segmentation using teaching-learning-based optimization algorithm and fuzzy entropy. In: Proceedings of the 15th International Conference on Computational Science and its applications (ICCSA 2015), 22–25 June 2015, Banff, Calgary, Canada, pp. 67–71 (2015)
27. Khehra, B.S, Singh, A., Hura, G.S., Kaur, L.: Fuzzy 2-partition Kapur entropy for image segmentation using teaching-learning-based optimization algorithm. In: Proceedings of the 3rd

IEEE International Conference on Image Processing, Applications and System (IPAS-2018), 12–14 December 2018, Inria Sophia Antipolis, France, pp. 198–203 (2018)
28. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 67–82 (1997)
29. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
30. Kamboj, V.K., Bath, S.K., Dhillon, J.S.: Solution of non-convex economic load dispatch problem using grey wolf optimizer. Neural Comput. Appl. **27**, 1301–1316 (2016)
31. Long, W., Jiao, J., Liang, X., Tang, M.: Inspired grey wolf optimizer for solving large-scale function optimization problems. Appl. Math. Model. **60**, 112–126 (2018)
32. Chawla, V.K., Chanda, A.K., Angra, S.: Automatic guided vehicles fleet size optimization for flexible manufacturing system by grey wolf optimization algorithm. Manag. Sci. Lett. **8**, 79–90 (2018)
33. Medjahed, A., Saadi, T.A., Benyettou, A., Ouali, M.: Gray wolf optimizer for hyper spectral band selection. Appl. Soft Comput. **40**, 178–186 (2016)
34. Nayak, B., Mohapatra, A., Mohanty, K.B.: Parameter estimation of single diode PV module based on GWO algorithm. Renew. Energy Focus **30**, 1–12 (2019)
35. Saxena, A., Soni, B.P., Kumar, R., Gupta, V.: Intelligent grey wolf optimizer – development and application for strategic bidding in uniform price spot energy market. Appl. Soft Comput. **69**, 1–13 (2018)
36. Khairuzzaman, A.K.M., Chaudhury, S.: Multilevel thresholding using grey wolf optimizer for image segmentation. Expert Syst. Appl. **86**, 64–76 (2017)
37. Kapoor, S., Zeya, I., Singhal, C., Nanda, S.J.: A grey wolf optimizer based automatic clustering algorithm for satellite image segmentation. In: Proceedings of 7th International Conference on Advances in Computing & Communications, ICACC-2017, 22–24 August 2017, Cochin, India, vol. 115, pp. 415–422 (2017).
38. Kapur, J.N.: Measures of Information and Their Applications. Wiley, New Delhi (1994)
39. Zhang, F.N.P.F., Li, J., Ding, D.: A novel generalized entropy and its application in image thresholding. Signal Process. **134**, 23–34 (2017)
40. http://decsai.ugr.es/cvg/CG/base.html
41. Horng, M.H.: Multilevel minimum cross entropy threshold selection based on the honey bee mating optimization. Expert Syst. Appl. **37**(6), 4580–4592 (2010)
42. Horng, M.H., Liou, R.J.: Multilevel minimum cross entropy threshold selection based on the firefly algorithm. Expert Syst. Appl. **38**(12), 14805–14811 (2011)
43. Nie, F., Gao, C., Guo, Y., Gan, M.: Two-dimensional minimum local cross-entropy thresholding based on co-occurrence matrix. Comput. Electr. Eng. **37**(5), 757–767 (2011)
44. Tang, K., Yuan, X., Sun, T., Yang, J., Gao, S.: An improved scheme for minimum cross entropy threshold selection based on genetic algorithm. Knowl.-Based Syst. **24**(8), 1131–1138 (2011)
45. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Processing **13**(4), 1–14 (2004)

Improved Artificial Bee Colony Algorithm with Adaptive Pursuit Based Strategy Selection

Osman Gokalp

Abstract Adaptive parameter setting techniques are called parameter control and they are more advantageous than offline tuning approaches. Because parameter tuning itself is an optimization problem and may be time-consuming. Moreover, optimization is a dynamic process and the parameter set-up at the initial stage of the search may not perform well in the later stages. Artificial Bee Colony (ABC) is one of the important algorithms in metaheuristic optimization. In order to contribute to the literature on parameter control in ABC, this study presents an Adaptive Pursuit (AP) selection rule-based improved ABC algorithm (IABC-AP) that automatically controls among 9 search strategies while the optimization process is running. The AP updates the selection probabilities of alternative operators based on their performance and reflects the changes in search space. In order to evaluate the performance of the developed algorithm, an experimental study was performed using a common continuous optimization benchmark called CEC 2017. The numerical results showed that the adaptive selection among 9 search strategies works successfully that the IABC-AP outperforms each of the individual strategies employed. Also, it was seen that the proposed method provides competitive results with other ABC algorithms.

Keywords Artificial bee colony · Adaptive pursuit · Continuous optimization

1 Introduction

Inspired by the foraging behavior of honey bees, Artificial Bee Colony (ABC) is one of the mostly used optimization algorithms for continuous optimization. It has numerous applications for solving real-world optimization problems such as the economic dispatch problem of power system operations [1], selection of cluster heads for energy optimization in wireless sensor networks [2], engineering design optimization [3], and optimum design of steel space [4].

O. Gokalp (✉)
Department of Computer Engineering, Izmir Katip Celebi University, Cigli, Izmir, Turkey
e-mail: osman.gokalp@ikcu.edu.tr

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_3

Since it was first proposed in [5], numerous variants of ABC using different search strategies have been proposed so far to improve the capability of the standard ABC search strategy. Some of the important ABC versions are listed as follows. Gbest-guided ABC [6] incorporated the global best solution into the standard ABC search equation as an additional term to strengthen the exploitation ability of the algorithm. Improved ABC (IABC) [7] employed two different search techniques that are ABC/rand/1 and ABC/best/1, and alternated them with a given probability parameter. Chaotic ABC [8] used chaotic maps to replace the random number generation process and improved the solution quality by escaping better from local optima. Incremental ABC [9] increased the size of the population over time and performed a local search on solutions selected.

The selection of good parameter values at the design or initialization stage of algorithms is called parameter tuning, and it is crucial for the optimization performance. However, adjusting parameters each time for a problem at hand is a hard task and requires expert knowledge. Therefore, another technique called parameter control can be used to change the values of parameters automatically while an optimization algorithm keeps running [10].

When adaptive parameter control techniques are applied for the selection of multiple algorithms or search strategies, this is called adaptive operator (strategy, or algorithm) selection (AOS) [11]. AOS can be applied to alternate between different algorithms at a high level [12], as well as to select among different search strategies in the same algorithm [13].

In this study, we present an adaptive IABC algorithm that selects among 9 different search strategies that are produced using 2 search equations. The algorithm employs Adaptive Pursuit (AP) [14], which is one of the AOS techniques, to adapt its selection probabilities using the feedback collected from the search process. In order to validate the proposed method, we conducted an experimental study on CEC 2017 benchmark [15]. Experimental results suggested that the efficacy of the IABC strategies depends on the function to be solved, and the proposed adaptive selection mechanism outperforms per individual strategy. It was also shown that this algorithm can outperform the original ABC, as well as one of the up-to-date ABC variants.

The organization of this paper is as follows. Section 2 discusses the main concepts for the proposed algorithm such as numerical optimization, the original ABC algorithm, the IABC algorithm, and the AP selection mechanism. Then, Sect. 3 explains clearly how the proposed IABC algorithm with AP-based strategy selection mechanism, or IABC-AP, was developed. After that, Sect. 4 presents an experimental study that was conducted to present the performance of the developed algorithm. Finally, Sect. 5 concludes the paper.

2 The Background

This section is devoted to discussing the main concepts for the proposed algorithm. First, the single objective bound-constrained real-parameter numerical optimization problem is formalized. Then, the original ABC algorithm is described by giving its general algorithmic structure. Next, one of the successful variants of ABC, namely improved ABC, is explained. Lastly, the probability matching technique is defined.

2.1 Single Objective Bound-Constrained Real-Parameter Numerical Optimization

A single objective continuous or numerical optimization problem can be defined as follows:

$$\min f(X), X = (x_1, x_2, \ldots, x_D) \tag{1}$$

where, $x_i \in R, i = 1, 2, \ldots, D$, and D is the problem dimension or the number of parameters to be optimized. In other words, the goal is to minimize the single objective function $f{:}R^D \rightarrow R$ by finding the best $X = (x_1, x_2, \ldots, x_D)$ real number vector in the search space. When the upper and lower limits for each number in X is specified, i.e. $L_i \leq x_i \leq U_i$, the problem is called bound-constrained.

2.2 Original ABC Algorithm

Inspired by the food search behavior of honey bees, the ABC algorithm has four major components: (i) food sources, (ii) employed bees, (iv) onlooker bees, and (iv) scout bees. Food sources are associated to candidate solutions of the optimization problem at hand. Employed bees are associated with food sources and used for searching the neighborhood to find better sources. The duty of onlooker bees is first to choose among food sources so that the better has a better chance, and then do a search over that solution. Note that this selection may leave some of the food sources unselected, especially low-quality ones. This can be associated the exploitation behavior because the search is concentrated on promising regions. Lastly, scout bees are used to discover completely new food sources with a random search if new neighbor food sources cannot be detected for a long time (i.e. source is exhausted). This can be associated with the exploration behavior.

Algorithm 1 outlines the general structure of the ABC metaheuristic that has three main phases devoted to three types of bees [16]. Food sources are initialized with uniform random numbers within specified search bounds. The algorithm continues until the specified termination condition is satisfied, e.g. the maximum number of iterations, the maximum number of fitness function evaluations, or the maximum time elapsed. At each iteration, firstly, the employed bees phase is applied. As a second step, the onlooker bees phase starts, and high-quality solutions are searched again. Then, the scout bees phase replaces a new solution with an exhausted one if available, i.e. solution that has not been improved for a long time specified by a limit value. Lastly, the best solution found is saved.

Algorithm 1: Pseudocode of ABC

```
Initialize parameters;
Initialize food sources;
while termination condition is not met do
    Employed bees phase;
    Onlooker bees phase;
    Scout bees phase;
    Memorize the best solution found so far;
end
```

For the adaptation of ABC to solve the real-parameter optimization problems, solutions are represented by a real number vector $X = (x_1, x_2, \ldots, x_D)$, where D is the dimension of the problem and each $x_i \in R$. Then, the population of solutions is represented by $P = (X_1, X_2, \ldots, X_{NP})$, where NP is the number of solutions. The search equation that is used in both employed and onlooker bee phases is defined in (2), where $i = 1, 2, \ldots NP$ is the current solution index, j is a randomly chosen dimension index, V is the neighbor solution that is being searched for, $k \neq i$ is a random solution index, and ϕ is the random real number in the range $[-1, 1]$.

$$V_{i,j} = X_{i,j} + \phi_{i,j}\left(X_{i,j} - X_{k,j}\right) \tag{2}$$

The goal of the algorithm is to find $X \in R^D$ such that the value of fitness is maximized. Fitness calculation is done by using (3), where $f(.)$ is the objective function of a minimization problem. Note that fitness and objective functions have different roles in ABC. The objective function is calculated for a given problem and it is tried to be minimized, whereas the fitness function is calculated using the objective value and it is tried to be maximized.

$$fit(\mathrm{X_i}) = \begin{cases} \frac{1}{1+f(\mathrm{X_i})}, \; if \; f(\mathrm{X_i}) > 0 \\ 1 + \left|f(X_\mathrm{i})\right|, \; \text{otherwise} \end{cases} \tag{3}$$

The probabilistic selection that is performed at the onlooker bees phase is proportional to the fitness values of solutions. As it is given in (4), the probability of the ith solution to be selected is calculated using the roulette wheel selection method.

$$p_i = \frac{fit(X_i)}{\sum_{i=1}^{NP} fit(X_i)} \tag{4}$$

Whenever a new neighbor solution V_i is produced in the neighborhood of X_i, it is accepted as a current ith solution if the fitness value of X_i is improved as in (5).

$$X_i = \begin{cases} V_i, \ if \ fit(X_i) \leq fit(V_i) \\ X_i, \ \text{otherwise} \end{cases} \tag{5}$$

Another important part of the ABC algorithm is its scout bees phase at which trial count statistics are used for deciding to replace unpromising solutions with new ones. This mechanism works as follows. At each solution acceptance step, a trial count is updated as in (6). At the end of each iteration, the solution with a maximum trial count is found. After that, the solution found is replaced with a random one if the trial count has reached the $limit$ value specified.

$$\text{triall}_i = \begin{cases} 0, \ if \ fit(\mathrm{X}_i) \leq fit(\mathrm{V}_i) \\ \text{trial}_i + 1, \ \text{otherwise} \end{cases} \tag{6}$$

It can be seen that the ABC algorithm has only 2 main parameters to be specified that are the population size NP and the $limit$ value.

2.3 Improved ABC Algorithm

Different from the original ABC, Improved ABC (IABC) [7] makes use of two solution search equations that are called (i) *ABC/rand/1* and (ii) *ABC/best/1*. These two equations are emulated from Differential Evolution (DE) algorithm [17] which includes solution mutation strategies *DE/rand/1* and *DE/best/1*. IABC also introduces the parameter M which is used to determine how many dimensions will be taken into account at neighbor generation.

The definitions of IABC solution search equations are given as follows.

$$ABC/best/1 : V_{i,m} = X_{best,m} + \phi_{i,m}\left(X_{i,m} - X_{r1,m}\right) \tag{7}$$

$$ABC/rand/1 : V_{i,m} = X_{r1,m} + \phi_{i,m}\left(X_{i,m} - X_{r2,m}\right) \tag{8}$$

where, X_{best} is the best solution found so far (global best), $r1$ and $r2$ are the random solution indices that satisfy the inequality $r1 \neq r2 \neq i$. The updated dimension is represented by m. As in the original ABC, X denotes the current solution, V denotes the neighbor solution, and ϕ is a random real number between $[-1, 1]$.

The general algorithmic structure of IABC is similar to the original ABC that is defined in Sect. 2.2, except for the neighbor solution generation. As it was mentioned at the beginning of this sub-section, IABC uses two new equations in this step. In order to select between the equations, it introduces the parameter $p \in [0, 1]$, which is the probability of selecting *DE/rand/1* strategy. Also, IABC is capable of updating more than one dimension at a time, which is controlled by M. Together, the neighbor solution search mechanism of IABC is outlined in Algorithm 2, where rnd is a random number between [0, 1]. Note that dimensions to be updated are also selected randomly at the beginning, i.e., the random permutation is produced of size D and saved into w.

In IABC, the parameters p and M are of great importance to the performance of the algorithm. So, they should be tuned properly for the problem at hand. However, we know from the no free lunch theorem [18] that there is no single optimization algorithm that outperforms others, i.e., a single IABC parameter set-up may perform well on one problem instance and may not be successful on another instance. Therefore, it becomes crucial to adjust these parameters adaptively for a given optimization problem.

Algorithm 2: IABC neighbor solution generation

$m \leftarrow 1$;
$w \leftarrow randPerm(D)$;
if $rnd<p$ **then**
  while $m \leq M$ **do**
    $\phi_{i,w(m)} \leftarrow 1 - 2 \times rnd$;
    $V_{i,w(m)} \leftarrow X_{r1,w(m)} + \phi_{i,w(m)}(X_{i,w(m)} - X_{r2,w(m)})$;
    $m \leftarrow m + 1$;
  end
else
  while $m \leq M$ **do**
    $\phi_{i,w(m)} \leftarrow 1 - 2 \times rnd$;
    $V_{i,w(m)} \leftarrow X_{best,w(m)} + \phi_{i,w(m)}(X_{i,w(m)} - X_{r1,w(m)})$;
    $m \leftarrow m + 1$;
  end
end

Lastly, we should also mention that IABC also employs the opposition-based learning method along with chaotic systems in its original paper for the initialization of the first population. However, in this study, we prefer using the standard random initialization as in the original ABC because we want to focus more on adaptive search strategy selection.

2.4 Adaptive Pursuit

Adaptive Pursuit (AP) [14] is an adaptive technique to allocate probabilities for a given set of operators, or strategies. In the context of optimization, the term operator corresponds to a discrete parameter such as a crossover operator in Genetic Algorithms [19], vector mutation strategy in DE, neighbor solution generation strategy in ABC, or even the algorithm itself in case of high-level hybrid methods. Basically, AP provides a method to adapt operator selection probabilities by matching them to the reward distribution that is obtained after their application to a given problem. The term reward is used to state how much each operator contributed to the goal of a problem; in the context of optimization, it can be associated with the improvement of the fitness value.

Quality evaluations, probability calculations, and adaptation rules in AP are outlined in Algorithm 3. This algorithm runs in parallel with an optimization algorithm until the stopping condition is met. The probabilities of the operators are updated after each reward calculation as a result of fitness function evaluation. Notations used in AP are explained as follows. K denotes the number of operators. P_{min} is the lower selection probability limit of an operator. Using this parameter, the maximum selection probability, P_{max}, can also be determined by calculating what is left after distributing the P_{min} for $K-1$ operators. α and β are the other two critical parameters of AP that the former controls how quality values, or Q, are updated, and the latter controls how probabilities, or P, are adapted. It is also worth mentioning that AP differentiates between reward (R) and quality. A reward is a temporary variable that has always the latest performance of an operator, whereas quality takes previous rewards into account and provides a historical success of the operators. Therefore, quality provides a more robust way in the adaptation of probabilities instead of directly using the latest rewards obtained.

Algorithm 3: Probability Allocation with Adaptive Pursuit

$P_{max} \leftarrow 1-(K-1)P_{min}$;
for $i \leftarrow 1$ *to* K **do**
  $P_i \leftarrow 1/K$;
  $Q_i \leftarrow 1$;
end
while *Termination condition is not met* **do**
  $\alpha^s \leftarrow$ selectOperatorProbabilistically(P);
  $R_{a^s}(t) \leftarrow$ getReward(α^s);
  $Q_{a^s}(t+1) \leftarrow Q_{a^s}(t) + \alpha[R_{a^s}(t) - Q_{a^s}(t)]$;
  $\alpha^* \leftarrow ARGMAX_{\alpha=1\cdots K}(Q_a(t+1))$;
  $P_{a^*}(t+1) \leftarrow P_{a^*}(t) + \beta[P_{max} - P_{a^*}(t)]$;
  for $\alpha \leftarrow 1$ *to* K **do**
    if $\alpha \neq \alpha^*$ **then**
      $P_a(t+1) \leftarrow P_a(t) + \beta[P_{min} - P_a(t)]$;
    end
  end
end

The general working of the AP algorithm is summarized as follows. At the initialization phase, P_{max} value is calculated, and probability values are assigned equally as $1/K$ for each operator. In addition, starting quality values per operator are set as 1. Then the main loop begins and works until the termination criterion is satisfied. At each iteration of the loop, firstly, the selected operator is determined according to their probabilities. Secondly, the selected operator is used, and the reward is obtained. After that, the quality value of the selected operator is updated with regard to parameter $alpha \in [0, 1]$. As α gets close to 1 the effect of the latest reward increases. On the other hand, as it gets close to 0, the past quality value is preserved more. In the next steps, the selection probability of the highest quality operator is increased towards P_{max}, whereas probabilities of the rest are decreased towards P_{min}. Note that P_{max} and P_{min} are the upper and lower limits for any of the selection probabilities, respectively.

3 The Proposed Adaptive Pursuit Based IABC Algorithm

This section explains clearly how the proposed IABC algorithm with AP-based strategy selection mechanism, or IABC-AP, was developed. At first, we discuss how the strategies in IABC were constructed. Then, we give the details of the IABC-AP algorithm and explain how AP selection can be integrated into IABC to select these strategies adaptively during the optimization process.

The outline of the proposed algorithm is given in Algorithm 4. The first part of the initialization process is done for AP. In this step, P_{max} is calculated and the initial values for probability (P) and quality Q arrays are set (see Algorithm 3). The second part of the initialization is about the IABC algorithm, and the initial population is created randomly within specified search bounds. Also, the limit parameter is set in this step. Then, the main loop of the IABC-AP starts and continues until the stopping criterion.

As one of the critical decisions for this algorithm, we first need to determine what strategies are employed in the search process. To do this, we select the most critical parameters of IABC, namely p and M. Because M specifies how many dimensions are going to be updated, we make it dimension-independent by introducing the parameter $scaleM$. In this way, the calculation $M = \lfloor scaleM \times D \rfloor$ can be done to determine how many dimensions will be updated for a given problem dimension. Therefore, we constructed 9 pairs of parameter combinations, or strategies, as listed in Table 1. These strategies are determined by discretizing the parameter values and combining them to cover as many different search behaviors as possible while

Table 1 Strategies for the IABC-AP algorithm

Strategy	p	*scale M*
1	0.1	0.1
2	0.5	0.1
3	0.9	0.1
4	0.1	0.5
5	0.5	0.5
6	0.9	0.5
7	0.1	0.9
8	0.5	0.9
9	0.9	0.9

keeping the number of strategies acceptable. We prefer to use 3 discretized values per parameter and hence obtain 3×3 pairs. Although it is possible to increase the number of parameter combinations by constructing 4×4 or 5×5 pairs, this will make AP adapt harder for a given run budget and should be decided carefully. In the preliminary study, we observed that 9 strategies are enough for this particular work.

The general outline of the proposed algorithm's main loop is similar to that of ABC, but it additionally involves reward calculation and strategy selection steps. Before applying the neighbor search rule of IABC (see Algorithm 2), which is needed in employed and onlooker bees phases, it first decides which parameter values are going to be assigned. To achieve this, it probabilistically chooses one of the strategies with roulette wheel selection using the probabilities provided by the AP algorithm. Then, the values of M and p are set and the IABC neighbor search rule is applied. When the reward is calculated by subtracting the objective value of a neighbor solution from the current solution, AP processes this information to adapt the probabilities of the strategies. The neighbor solution is accepted if it satisfies the condition provided in (5). After the employed and onlooker bees phases are completed, the scout bees phase is applied and then the global best solution is updated before the next iteration of the main loop begins.

Algorithm 4: IABC-AP

```
AP ←InitializeAP();
initializeIABC();
while Termination condition is not met do
    for i ← 1 to NP do                                   ▷ Employed bees phase
        α^s ← AP.selectOperatorProbablistically();
        scaleM ← α^s.getScaleM();
        p ← α^s.getP();
        M ← ⌊scaleM × D⌋;
        Find neighbor solution V_i using Algorithm 2;
        reward ← X_i.f − V_i.f;
        AP.processReward(reward, α^s);
        Accept neighbor solution using (5);
    end
    for t ← 1 to NP do                                   ▷ Onlooker bees phase
        i ← selectSolutionWithFitnessProportionalProbability();
        Execute the commands from step 5 to 12 and return here;
    end
    Apply scout bees phase rules;
    Update global best solution;
end
```

We should also mention that the AP operator selection in this study needs a warm-up phase before it actually begins to work. Because the first selection will be completely random and the reward may be large, the first chosen strategy may gain an advantage over others. Although this issue can be eliminated through iterations via the adaptive mechanism of the AP, some of the search budgets will be used to compensate for it. Therefore, after preliminary observations, we decided to make $NP \times 2$ selection at random no matter which operator probabilities have been built so far. After this warm-up process is over, AP selection returns to its normal process and begins to provide probabilistic selection again.

Last but not least, because we are dealing with a bound-constrained optimization problem in this study, the boundaries for each solution should be fixed when necessary in the search process. Specifically, whenever a solution having one or more dimensions above the upper limit or below the lower limit is produced, corresponding dimension values are replaced with the nearest limit values.

4 Experimental Work

This section presents an experimental work that was performed to present the performance of the IABC-AP algorithm. First, the experimental environment and the benchmark are introduced. Second, the detailed numerical results obtained are presented. Finally, the performance of the developed algorithm is assessed by making comparisons with alternative approaches.

4.1 Experimental Setting

In this study, CEC 2017 [15] benchmark, which is one of the common test set for continuous optimization, has been used. This benchmark includes 29 single objective bound-constrained real-parameter optimization problems[1] in four categories and is publicly available[2] for interested researchers. The first category includes 2 unimodal functions from F1 to F2, the second category includes 7 simple multi-modal functions from F3 to F9, the third category includes 10 hybrid functions from F10 to F19, and the last category includes 10 composition functions from F20 to F29.

After preliminary experiments, we set NP to 20, set the limit parameter of IABC to $D \times NP$, set α and β parameters of AP to 0.5, and set p_{min} parameter of AP to 0.025. The rest of the experimental study follows the guideline in [15] and can be summarized as follows. The termination criterion was set as $D \times 10,000$ for each run of any algorithm in this study. Search range was taken as $[-100, 100]$ for each test function. Uniform random initialization was used for the populations within the search range. As a performance measure, the error value was calculated as $f(x) - f^*$, where $f(x)$ is the objective value of solution x, and f^* is the optimum objective value provided for the corresponding test function. Error-values less than 10^{-8} were considered as 0, and if this threshold was reached, the algorithms were terminated immediately. Because of the stochastic behavior of the optimization algorithms used in this study, they were run 51 times with different random seeds in order to get more accurate results.

4.2 Numerical Results

This section provides detailed numerical results obtained after running the proposed IABC with AP-based strategy selection. Tables 2, 3, 4, and 5 list the error values for the problem dimensions 10, 30, 50, and 100 respectively. The columns best, worst, mean, and standard deviation present statistical information for 51 independent runs of the algorithm. For dimension 10, the algorithm produces zero error values for 4 of the functions (F2, F5, F8, F19) at every run. In addition, it achieves zero error for 4 other functions (F4, F16, F21, F25) in terms of best values. As the problem dimension increases, it is expected that the functions become harder to solve. Indeed, in terms of best values among 51 runs, zero error values are only available for function F5 (dimensions 30, 50, and 100), and for function F8 (dimensions 30 and 50).

[1] Although the initial version of this benchmark contains 30 functions, F2 has been excluded later because of its unstable behavior. Then, function indices from F3 to F30 have been reduced by one and their optimum values have been reduced by 100 to accommodate this change. Therefore, CEC 2017 has now 29 test functions.

[2] https://github.com/P-N-Suganthan/CEC2017-BoundContrained.

Table 2 Numerical results of the IABC-AP algorithm for 10 dimensions

Func.	Best	Worst	Median	Mean	Std. Dev.
F1	5.14E+00	1.24E+04	1.15E+03	2.04E+03	2.65E+03
F2	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F3	1.36E-03	5.64E+00	3.87E+00	3.61E+00	1.44E+00
F4	**0.00E+00**	6.97E+00	3.00E+00	3.40E+00	1.21E+00
F5	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F6	1.07E+01	1.82E+01	1.34E+01	1.35E+01	1.48E+00
F7	4.66E-01	5.02E+00	3.03E+00	3.28E+00	1.09E+00
F8	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F9	3.57E+00	2.80E+02	5.38E+01	8.97E+01	8.49E+01
F10	2.47E-04	1.01E+01	1.12E+00	1.82E+00	1.78E+00
F11	5.05E+01	4.30E+04	7.24E+03	1.09E+04	1.15E+04
F12	3.21E+00	2.50E+04	1.59E+03	3.80E+03	5.79E+03
F13	2.88E-05	6.97E+00	2.01E+00	2.34E+00	1.58E+00
F14	4.68E-02	8.11E+01	1.70E+00	3.73E+00	1.13E+01
F15	2.18E-02	1.20E+02	6.59E-01	2.11E+01	4.29E+01
F16	**0.00E+00**	2.03E+01	3.32E-01	1.23E+00	3.62E+00
F17	1.30E+01	9.03E+03	3.21E+02	8.31E+02	1.54E+03
F18	1.05E-03	1.03E+02	1.02E+00	4.79E+00	1.59E+01
F19	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F20	1.00E+02	2.10E+02	1.02E+02	1.39E+02	5.04E+01
F21	**0.00E+00**	1.02E+02	1.00E+02	9.05E+01	3.01E+01
F22	3.00E+02	3.10E+02	3.05E+02	3.05E+02	2.32E+00
F23	1.00E+02	3.39E+02	3.34E+02	3.02E+02	8.15E+01
F24	3.98E+02	4.49E+02	4.44E+02	4.25E+02	2.35E+01
F25	**0.00E+00**	3.98E+02	3.00E+02	2.71E+02	8.77E+01
F26	3.83E+02	3.97E+02	3.87E+02	3.87E+02	2.93E+00
F27	3.00E+02	5.75E+02	5.19E+02	4.47E+02	1.17E+02
F28	2.33E+02	2.54E+02	2.41E+02	2.41E+02	5.94E+00
F29	5.90E+02	1.80E+04	9.09E+02	1.45E+03	2.45E+03

4.3 Comparison and Discussion

To evaluate the performance of the proposed method, we first compare the adaptive selection mechanism with the individual strategies from which it chooses. As it was given in Table 1 before, there are 9 different search strategies, or parameter combinations, for designing the IABC algorithm. In this section, we refer to each strategy with its number, i.e. strat1 corresponds to the strategy number 1. Tables 6, 7, 8, and 9 list mean error values for each strategy and the proposed IABC-AP algorithm

Table 3 Numerical results of the IABC-AP algorithm for 30 dimensions

Func.	Best	Worst	Median	Mean	Std. Dev.
F1	1.65E+00	1.99E+04	4.12E+03	7.11E+03	7.11E+03
F2	6.10E-01	1.54E+04	6.11E+03	6.71E+03	4.10E+03
F3	4.02E+00	1.13E+02	7.56E+01	7.42E+01	1.83E+01
F4	1.71E+01	5.77E+01	3.67E+01	3.61E+01	8.79E+00
F5	**0.00E+00**	1.37E-07	**0.00E+00**	2.68E-09	1.92E-08
F6	5.11E+01	1.03E+02	6.96E+01	7.09E+01	1.20E+01
F7	2.00E+01	6.28E+01	3.65E+01	3.67E+01	9.27E+00
F8	**0.00E+00**	1.18E+00	8.95E-02	2.02E-01	2.45E-01
F9	1.65E+03	4.41E+03	3.44E+03	3.39E+03	6.98E+02
F10	7.75E+00	9.39E+01	2.79E+01	3.96E+01	2.73E+01
F11	1.32E+03	1.05E+05	3.20E+04	3.70E+04	2.01E+04
F12	8.93E+01	2.55E+05	1.65E+04	2.90E+04	3.98E+04
F13	9.75E+01	3.54E+04	8.30E+03	1.09E+04	8.58E+03
F14	2.91E+01	4.13E+04	4.40E+03	9.08E+03	1.05E+04
F15	3.28E+01	6.56E+02	2.82E+02	3.15E+02	1.73E+02
F16	2.74E+01	1.84E+02	5.88E+01	8.06E+01	5.00E+01
F17	2.28E+03	4.68E+05	9.28E+04	1.13E+05	7.75E+04
F18	1.45E+01	5.16E+04	4.74E+03	1.06E+04	1.39E+04
F19	2.14E+01	1.72E+02	3.54E+01	7.16E+01	5.92E+01
F20	2.21E+02	2.83E+02	2.41E+02	2.42E+02	1.16E+01
F21	1.00E+02	1.05E+02	1.00E+02	1.00E+02	1.04E+00
F22	3.66E+02	4.26E+02	3.93E+02	3.94E+02	1.09E+01
F23	4.72E+02	5.49E+02	5.03E+02	5.02E+02	1.67E+01
F24	3.80E+02	4.06E+02	3.82E+02	3.84E+02	4.55E+00
F25	2.00E+02	1.59E+03	1.30E+03	1.13E+03	4.56E+02
F26	4.86E+02	5.33E+02	5.05E+02	5.05E+02	1.06E+01
F27	3.00E+02	4.68E+02	4.09E+02	4.04E+02	2.91E+01
F28	3.67E+02	7.09E+02	4.90E+02	5.06E+02	6.80E+01
F29	1.97E+03	1.60E+04	5.02E+03	6.74E+03	4.38E+03

for dimensions 30 and 50. To evaluate these results statistically, we applied the Mann–Whitney U test per function using the results of 51 independent runs. The signs $+$, $=$, $-$ located right of each value indicate that the proposed algorithm is better than (win), worse than (lose), and equal to (tie) the compared algorithm for a given function with respect to $p = 0.05$ significance level. The last row of the tables report summary for 29 functions. We can conclude that the performance of the strategies depends on the function to be solved, and the proposed adaptive selection algorithm outperforms per individual strategy in terms of total win, tie and lose counts.

Table 4 Numerical results of the IABC-AP algorithm for 50 dimensions

Func.	Best	Worst	Median	Mean	Std. Dev.
F1	3.29E+00	3.17E+04	4.09E+03	7.49E+03	9.38E+03
F2	3.36E+04	8.55E+04	5.86E+04	5.80E+04	1.09E+04
F3	8.06E+00	1.96E+02	1.09E+02	9.55E+01	5.05E+01
F4	5.09E+01	1.92E+02	9.21E+01	9.94E+01	3.09E+01
F5	**0.00E+00**	4.17E-02	**0.00E+00**	8.17E-04	5.83E-03
F6	1.10E+02	2.60E+02	1.52E+02	1.66E+02	4.50E+01
F7	5.78E+01	2.04E+02	8.76E+01	9.03E+01	2.30E+01
F8	**0.00E+00**	2.99E+00	6.33E-01	8.09E-01	6.73E-01
F9	6.01E+03	9.97E+03	9.06E+03	8.85E+03	8.66E+02
F10	4.15E+01	1.85E+02	9.06E+01	9.30E+01	3.26E+01
F11	4.53E+04	2.56E+06	5.07E+05	6.35E+05	5.36E+05
F12	3.05E+02	3.42E+04	8.05E+03	1.06E+04	1.01E+04
F13	5.86E+03	2.15E+05	5.26E+04	6.12E+04	4.61E+04
F14	1.40E+02	2.87E+04	3.59E+03	7.07E+03	6.94E+03
F15	3.53E+02	1.33E+03	8.95E+02	8.79E+02	2.37E+02
F16	1.25E+02	9.11E+02	4.92E+02	5.23E+02	1.60E+02
F17	7.22E+04	3.18E+06	6.69E+05	8.33E+05	7.20E+05
F18	1.31E+02	4.26E+04	1.52E+04	1.68E+04	1.27E+04
F19	7.68E+01	6.26E+02	3.51E+02	3.40E+02	1.48E+02
F20	2.70E+02	4.28E+02	3.10E+02	3.19E+02	3.66E+01
F21	1.00E+02	1.06E+04	9.16E+03	8.24E+03	2.82E+03
F22	4.96E+02	6.74E+0 2	5.42E+02	5.42E+02	3.08E+01
F23	6.17E+02	8.66E+02	7.01E+02	7.15E+02	6.13E+01
F24	4.59E+02	5.78E+02	5.29E+02	5.24E+02	3.34E+01
F25	1.66E+03	2.70E+03	2.02E+03	2.05E+03	1.81E+02
F26	5.07E+02	7.24E+02	5.52E+02	5.60E+02	3.60E+01
F27	4.57E+02	5.14E+02	4.97E+02	4.95E+02	1.69E+01
F28	3.64E+02	8.87E+02	5.38E+02	5.43E+02	1.09E+02
F29	4.20E+05	2.20E+06	7.45E+05	8.44E+05	3.67E+05

After showing that the proposed adaptive selection mechanism works successfully, we then make a second comparison with the original ABC and one of the up-to-date ABC versions, namely culture-based ABC (CB-ABC) [20]. The reasons why we chose that study for comparison are that both algorithms have been evaluated on the CEC 2017 benchmark and the detailed results are provided. In [20] the authors set the limit parameter of the original ABC was to $NP \times D$, and set SN (population size) to 25. In addition, they determined the termination condition as 10,000 maximum iterations. Considering that the number of fitness function evaluations at

Table 5 Numerical results of the IABC-AP algorithm for 100 dimensions

Func.	Best	Worst	Median	Mean	Std. Dev.
F1	1.14E+02	6.11E+04	1.15E+04	1.92E+04	1.99E+04
F2	2.36E+05	3.82E+05	3.05E+05	3.06E+05	3.22E+04
F3	1.36E+02	2.86E+02	2.06E+02	2.06E+02	3.07E+01
F4	1.61E+02	7.08E+02	3.03E+02	3.68E+02	1.55E+02
F5	**0.00E+00**	2.83E-02	1.06E-07	1.24E-03	4.81E-03
F6	2.78E+02	8.70E+02	4.95E+02	5.30E+02	1.73E+02
F7	1.75E+02	7.18E+02	2.85E+02	3.46E+02	1.51E+02
F8	2.53E+00	1.70E+02	2.36E+01	4.02E+01	4.55E+01
F9	2.30E+04	2.64E+04	2.53E+04	2.50E+04	9.35E+02
F10	3.04E+02	8.53E+02	4.94E+02	5.13E+02	1.11E+02
F11	7.38E+05	1.61E+07	4.52E+06	5.77E+06	4.13E+06
F12	4.54E+02	4.31E+04	5.19E+03	9.85E+03	1.00E+04
F13	3.89E+05	6.94E+06	2.21E+06	2.40E+06	1.33E+06
F14	4.01E+02	1.82E+04	2.80E+03	3.75E+03	3.93E+03
F15	2.07E+03	3.97E+03	3.05E+03	3.06E+03	4.51E+02
F16	1.46E+03	3.00E+03	2.03E+03	2.06E+03	3.41E+02
F17	1.31E+06	8.38E+06	3.36E+06	3.79E+06	1.86E+06
F18	1.89E+02	2.73E+04	2.11E+03	4.28E+03	5.71E+03
F19	1.12E+03	2.39E+03	1.98E+03	1.96E+03	2.68E+02
F20	4.25E+02	8.93E+02	5.69E+02	5.98E+02	1.15E+02
F21	2.24E+04	2.75E+04	2.58E+04	2.56E+04	1.02E+03
F22	6.64E+02	1.05E+03	7.75E+02	7.92E+02	7.25E+01
F23	1.09E+03	1.42E+03	1.22E+03	1.22E+03	6.79E+01
F24	5.94E+02	8.94E+02	7.34E+02	7.42E+02	6.28E+01
F25	5.02E+03	8.01E+03	6.10E+03	6.20E+03	6.51E+02
F26	6.05E+02	7.41E+02	6.70E+02	6.71E+02	3.22E+01
F27	4.94E+02	6.34E+02	5.52E+02	5.56E+02	2.79E+01
F28	1.30E+03	2.95E+03	2.28E+03	2.27E+03	2.97E+02
F29	2.63E+03	9.58E+04	1.32E+04	1.55E+04	1.70E+04

each iteration is $2 \times SN$, *i.e.* two separate loops for employed and onlooker bees phases, ABC and CB-ABC algorithms will perform $2 \times SN \times 10{,}000$ fitness function evaluations in total. Therefore, it seems fair enough to compare our results with $D \times 10{,}000$ maximum number of functions evaluations with the results provided for the original ABC and the CB-ABC.

Table 10 reports and compares the performance of IABC-AP with the original ABC and the CB-ABC in terms of mean error. The results are provided for 30 dimensions. We can conclude that proposed algorithm outperforms the original ABC

Table 6 Comparison of the proposed adaptive selection mechanism with each individual strategies for dimension 30 (Part-I)

Func.	strat1	T.	strat2	T.	strat3	T.	strat4	T.	strat5	T.	IABC-AP
F1	1.96E+03	−	8.71E+02	−	1.09E+03	−	4.24E+05	=	6.94E+03	=	7.11E+03
F2	1.50E+04	+	3.22E+04	+	5.20E+04	+	1.45E-05	−	1.04E+04	+	6.71E+03
F3	6.53E+01	−	7.02E+01	=	8.24E+01	+	1.22E+02	+	6.28E+01	−	7.42E+01
F4	3.88E+01	=	4.64E+01	+	8.64E+01	+	6.61E+01	+	4.33E+01	+	3.61E+01
F5	1.68E-07	+	8.05E-09	=	0.00E+00	=	8.24E-01	+	2.42E-02	+	2.68E-09
F6	6.94E+01	=	8.32E+01	+	1.21E+02	+	1.09E+02	+	6.78E+01	=	7.09E+01
F7	3.86E+01	=	5.01E+01	+	8.86E+01	+	6.38E+01	+	4.27E+01	+	3.67E+01
F8	5.13E-01	+	3.62E-01	=	2.40E-01	=	1.18E+02	+	2.78E+00	+	2.02E-01
F9	2.61E+03	−	3.82E+03	+	4.39E+03	+	2.52E+03	-	3.50E+03	=	3.39E+03
F10	5.27E+01	+	2.94E+01	−	3.37E+01	=	1.59E+02	+	7.63E+01	+	3.96E+01
F11	2.74E+05	+	3.46E+05	+	8.03E+05	+	6.97E+05	+	3.26E+04	=	3.70E+04
F12	1.46E+04	−	1.15E+04	−	1.11E+04	−	2.55E+04	=	1.83E+04	=	2.90E+04
F13	5.03E+04	+	6.19E+04	+	8.18E+04	+	6.62E+03	−	8.19E+03	=	1.09E+04
F14	6.40E+03	=	4.16E+03	−	3.29E+03	−	9.78E+03	=	1.10E+04	=	9.08E+03
F15	4.27E+02	+	4.14E+02	+	4.34E+02	+	6.10E+02	+	3.44E+02	=	3.15E+02

(continued)

Table 6 (continued)

Func.	strat1	T.	strat2	T.	strat3	T.	strat4	T.	strat5	T.	IABC-AP
F16	8.98E+01	=	8.27E+01	=	9.82E+01	+	2.19E+02	+	1.21E+02	+	8.06E+01
F17	1.21E+05	=	1.59E+05	+	1.78E+05	+	1.89E+05	=	2.26E+05	+	1.13E+05
F18	3.55E+03	−	3.02E+03	−	3.82E+03	=	9.40E+03	=	8.54E+03	=	1.06E+04
F19	1.09E+02	+	1.23E+02	+	1.32E+02	+	1.76E+02	+	1.06E+02	+	7.16E+01
F20	2.47E+02	+	2.53E+02	+	2.84E+02	+	2.67E+02	+	2.46E+02	=	2.42E+02
F21	1.01E+02	+	1.00E+02	+	1.00E+02	−	1.35E+03	+	6.74E+02	+	1.00E+02
F22	3.92E+02	=	3.96E+02	=	4.27E+02	+	4.51E+02	+	4.05E+02	+	3.94E+02
F23	5.00E+02	=	5.08E+02	=	5.42E+02	+	5.07E+02	=	4.75E+02	−	5.02E+02
F24	3.82E+02	−	3.82E+02	−	3.83E+02	=	4.10E+02	+	3.89E+02	+	3.84E+02
F25	1.33E+03	+	1.13E+03	=	1.23E+03	+	1.88E+03	+	1.41E+03	+	1.13E+03
F26	5.00E+02	−	5.03E+02	=	5.09E+02	+	5.36E+02	+	5.20E+02	+	5.05E+02
F27	3.66E+02	−	3.77E+02	−	3.89E+02	−	4.99E+02	+	3.70E+02	=	4.04E+02
F28	5.07E+02	=	4.89E+02	=	5.35E+02	+	7.87E+02	+	6.59E+02	+	5.06E+02
F29	5.65E+03	=	6.61E+03	=	1.01E+04	+	9.27E+03	+	6.32E+03	=	6.74E+03
+/ = /−	11/10/8		12/10/7		19/5/5		20/6/3		15/12/2		

Table 7 Comparison of the proposed adaptive selection mechanism with each individual strategies for dimension 30 (Part-II)

Func.	strat6	T.	strat7	T.	strat8	T.	strat9	T.	IABC-AP
F1	7.94E+03	=	1.05E+10	+	2.70E+07	=	1.05E+04	+	7.11E+03
F2	5.95E+04	+	5.81E+02	−	0.00E+00	−	3.93E+04	+	6.71E+03
F3	5.58E+01	−	1.32E+03	+	8.66E+01	+	4.20E+01	−	7.42E+01
F4	1.72E+02	+	1.87E+02	+	8.14E+01	+	5.69E+01	+	3.61E+01
F5	3.73E-07	=	3.13E+01	+	4.09E+00	+	6.91E-02	+	2.68E-09
F6	2.07E+02	+	4.23E+02	+	1.28E+02	+	1.35E+02	+	7.09E+01
F7	1.73E+02	+	1.66E+02	+	8.55E+01	+	5.13E+01	+	3.67E+01
F8	1.59E-02	−	2.67E+03	+	2.29E+02	+	6.50E+00	+	2.02E-01
F9	6.61E+03	+	3.88E+03	+	3.37E+03	=	6.33E+03	+	3.39E+03
F10	7.92E+01	+	9.68E+02	+	1.84E+02	+	7.27E+01	+	3.96E+01
F11	5.90E+04	+	3.11E+08	+	9.57E+06	+	3.88E+04	=	3.70E+04
F12	1.20E+04	−	6.57E+07	+	2.15E+04	=	1.92E+04	=	2.90E+04
F13	1.20E+04	=	6.30E+05	+	4.93E+04	+	4.83E+03	−	1.09E+04
F14	7.46E+03	=	4.45E+04	+	1.31E+04	=	1.25E+04	=	9.08E+03
F15	5.65E+02	+	1.27E+03	+	6.74E+02	+	3.85E+02	=	3.15E+02
F16	8.96E+01	+	6.04E+02	+	3.61E+02	+	1.56E+02	+	8.06E+01
F17	1.07E+06	+	1.46E+06	+	2.44E+05	+	1.61E+05	=	1.13E+05
F18	5.22E+03	−	2.48E+05	+	1.02E+04	=	9.12E+03	=	1.06E+04
F19	4.93E+01	=	5.79E+02	+	3.70E+02	+	1.75E+02	+	7.16E+01
F20	3.68E+02	+	3.67E+02	+	2.90E+02	+	2.57E+02	=	2.42E+02
F21	7.41E+02	−	3.29E+03	+	1.56E+03	+	1.46E+03	=	1.00E+02
F22	5.10E+02	+	6.58E+02	+	5.00E+02	+	4.12E+02	+	3.94E+02
F23	5.99E+02	+	7.23E+02	+	5.36E+02	+	4.83E+02	−	5.02E+02
F24	3.87E+02	+	9.06E+02	+	4.06E+02	+	3.87E+02	+	3.84E+02
F25	1.94E+03	+	4.18E+03	+	2.27E+03	+	1.41E+03	+	1.13E+03
F26	5.07E+02	=	6.56E+02	+	5.57E+02	+	5.24E+02	+	5.05E+02
F27	3.50E+02	−	1.53E+03	+	4.64E+02	+	3.61E+02	−	4.04E+02
F28	5.92E+02	+	1.38E+03	+	1.04E+03	+	6.42E+02	+	5.06E+02
F29	5.65E+03	=	1.16E+07	+	3.07E+04	+	6.68E+03	=	6.74E+03
+/ = /−	16/7/6		28/0/1		23/5/1		16/9/4		

for all the functions. In addition, IABC-AP produces generally lower mean error values than the CB-ABC algorithm. Together, we can conclude that the IABC-AP is among the successful ABC variants for solving numerical optimization problems.

Table 8 Comparison of the proposed adaptive selection mechanism with each individual strategies for dimension 50 (Part-I)

Func.	strat1	T.	strat2	T.	strat3	T.	strat4	T.	strat5	T.	IABC-AP
F1	5.91E+03	=	5.07E+03	=	3.84E+03	=	3.14E+08	+	1.13E+04	=	7.49E+03
F2	7.27E+04	+	1.08E+05	+	1.45E+05	+	1.49E+03	−	5.00E+04	−	5.80E+04
F3	1.02E+02	=	9.50E+01	=	1.06E+02	=	3.08E+02	+	8.26E+01	=	9.55E+01
F4	1.02E+02	=	1.52E+02	+	2.50E+02	+	1.60E+02	+	9.30E+01	=	9.94E+01
F5	1.18E-03	+	3.59E-08	=	0.00E+00	=	4.62E+00	+	6.69E-02	+	8.17E-04
F6	1.49E+02	=	2.25E+02	+	3.04E+02	+	2.69E+02	+	1.33E+02	−	1.66E+02
F7	1.04E+02	+	1.50E+02	+	2.46E+02	+	1.54E+02	+	9.71E+01	=	9.03E+01
F8	2.26E+00	+	4.69E-01	−	4.45E-01	−	1.17E+03	+	2.67E+01	+	8.09E-01
F9	7.91E+03	−	8.99E+03	=	9.78E+03	+	5.15E+03	−	7.97E+03	=	8.85E+03
F10	1.28E+02	+	6.08E+01	−	8.79E+01	=	2.85E+02	+	1.69E+02	+	9.30E+01
F11	1.80E+06	+	2.26E+06	+	3.21E+06	+	6.55E+07	+	2.00E+05	−	6.35E+05
F12	8.52E+03	=	5.60E+03	−	6.51E+03	=	3.44E+05	+	7.50E+03	=	1.06E+04
F13	3.57E+05	+	5.13E+05	+	5.86E+05	+	3.89E+04	−	3.56E+04	−	6.12E+04
F14	5.34E+03	=	4.81E+03	=	4.53E+03	=	6.81E+03	=	5.85E+03	=	7.07E+03
F15	1.03E+03	+	9.98E+02	+	1.02E+03	+	1.31E+03	+	1.03E+03	+	8.79E+02
F16	6.37E+02	+	5.84E+02	=	6.80E+02	+	1.07E+03	+	7.31E+02	+	5.23E+02
F17	6.26E+05	=	6.70E+05	=	1.14E+06	+	6.05E+05	−	1.04E+06	=	8.33E+05

(continued)

Table 8 (continued)

Func.	strat1	T.	strat2	T.	strat3	T.	strat4	T.	strat5	T.	IABC-AP
F18	6.97E+03	−	6.52E+03	−	7.90E+03	−	1.95E+04	=	1.53E+04	=	1.68E+04
F19	4.72E+02	+	4.71E+02	+	5.39E+02	+	7.04E+02	+	5.61E+02	+	3.40E+02
F20	3.17E+02	=	3.61E+02	+	4.52E+02	+	3.54E+02	+	2.95E+02	−	3.19E+02
F21	6.78E+03	−	7.88E+03	=	8.66E+03	+	5.31E+03	−	7.54E+03	=	8.24E+03
F22	5.42E+02	=	5.67E+02	+	6.66E+02	+	6.66E+02	+	5.59E+02	+	5.42E+02
F23	6.99E+02	=	7.21E+02	=	8.27E+02	+	7.23E+02	=	6.20E+02	−	7.15E+02
F24	5.26E+02	=	5.18E+02	=	5.06E+02	−	6.51E+02	+	5.36E+02	+	5.24E+02
F25	2.12E+03	+	2.35E+03	+	3.33E+03	+	3.52E+03	+	2.43E+03	+	2.05E+03
F26	5.49E+02	=	5.41E+02	−	5.50E+02	=	8.20E+02	+	6.94E+02	+	5.60E+02
F27	5.00E+02	=	4.93E+02	=	4.83E+02	−	9.40E+02	+	5.12E+02	=	4.95E+02
F28	5.63E+02	=	5.41E+02	=	6.35E+02	+	1.24E+03	+	7.77E+02	+	5.43E+02
F29	6.57E+05	−	6.48E+05	−	7.16E+05	=	1.71E+06	+	1.01E+06	+	8.44E+05
+/ = /−	11/14/4		11/12/6		17/8/4		21/3/5		12/11/6		

Table 9 Comparison of the proposed adaptive selection mechanism with each individual strategies for dimension 50 (Part-II)

Func.	strat6	T.	strat7	T.	strat8	T.	strat9	T.	IABC-AP
F1	1.00E+04	=	5.07E+10	+	4.14E+08	+	1.29E+04	+	7.49E+03
F2	1.56E+0 5	+	1.82E+03	−	5.72E+02	−	1.22E+05	+	5.80E+04
F3	9.13E+01	=	6.21E+03	+	1.81E+02	+	7.85E+01	=	9.55E+01
F4	3.74E+02	+	4.25E+02	+	2.02E+02	+	1.10E+02	=	9.94E+01
F5	9.80E-04	=	4.81E+01	+	1.36E+01	+	2.20E-01	+	8.17E-04
F6	4.24E+02	+	1.29E+03	+	3.51E+02	+	2.29E+02	=	1.66E+02
F7	3.72E+02	+	4.16E+02	+	1.96E+02	+	1.03E+02	+	9.03E+01
F8	1.12E-01	−	1.01E+04	+	1.30E+03	+	8.42E+01	+	8.09E-01
F9	1.29E+04	+	7.60E+03	−	6.26E+03	−	1.29E+04	+	8.85E+03
F10	1.73E+02	+	5.29E+03	+	3.41E+02	+	1.41E+02	+	9.30E+01
F11	6.80E+05	=	6.91E+09	+	2.03E+07	+	2.45E+05	−	6.35E+05
F12	4.82E+03	−	1.30E+09	+	2.10E+04	+	1.09E+04	=	1.06E+04
F13	1.42E+05	+	3.77E+06	+	1.36E+05	+	3.02E+04	−	6.12E+04
F14	4.51E+03	=	4.46E+07	+	7.11E+03	=	4.73E+03	=	7.07E+03
F15	2.16E+03	+	2.32E+03	+	1.51E+03	+	1.09E+03	+	8.79E+02
F16	1.30E+03	+	1.93E+0 3	+	1.50E+03	+	9.30E+02	+	5.23E+02
F17	4.73E+06	+	1.28E+07	+	4.22E+05	−	7.01E+05	=	8.33E+05
F18	1.82E+04	=	7.40E+06	+	1.71E+04	=	2.12E+04	=	1.68E+04

(continued)

Table 9 (continued)

Func.	strat6	T.	strat7	T.	strat8	T.	strat9	T.	IABC-AP
F19	1.39E+03	+	1.23E+03	+	9.62E+02	+	8.12E+02	+	3.40E+02
F20	5.68E+02	+	6.20E+02	+	3.93E+02	+	3.11E+02	−	3.19E+02
F21	1.30E+04	+	7.92E+03	−	6.50E+03	−	1.27E+04	+	8.24E+03
F22	7.98E+02	+	1.18E+03	+	7.75E+02	+	5.86E+02	+	5.42E+02
F23	8.72E+02	+	1.18E+03	+	7.91E+02	+	6.46E+02	−	7.15E+02
F24	4.99E+02	−	4.29E+03	+	5.75E+02	+	5.27E+02	=	5.24E+02
F25	4.59E+03	+	8.86E+03	+	4.67E+03	+	2.59E+03	+	2.05E+03
F26	5.57E+02	=	1.43E+03	+	9.30E+02	+	6.42E+02	+	5.60E+02
F27	4.87E+02	−	5.33E+03	+	7.79E+02	+	5.04E+02	=	4.95E+02
F28	5.20E+02	−	2.99E+03	+	1.72E+03	+	9.31E+02	+	5.43E+02
F29	8.97E+05	+	5.46E+07	+	1.53E+06	+	1.04E+06	+	8.44E+05
+/ = /−	17/7/5		26/0/3		23/2/4		16/9/4		

Table 10 Comparison of the IABC-AP with other ABC algorithms for dimension 30

Func.	Orig. ABC	CB-ABC	IABC-AP
F1	7.33E+03	2.95E+03	7.11E+03
F2	1.39E+06	1.22E+05	6.71E+03
F3	4.22E+02	4.39E+02	7.42E+01
F4	7.21E+02	5.90E+02	3.61E+01
F5	6.00E+02	6.00E+02	2.68E-09
F6	9.54E+02	8.04E+02	7.09E+01
F7	1.03E+03	9.00E+02	3.67E+01
F8	1.50E+05	5.79E+03	2.02E-01
F9	1.08E+04	3.36E+03	3.39E+03
F10	1.25E+07	3.67E+04	3.96E+01
F11	7.89E+10	7.64E+07	3.70E+04
F12	4.84E+05	1.82E+04	2.90E+04
F13	7.23E+07	1.10E+07	1.09E+04
F14	3.58E+05	1.22E+06	9.08E+03
F15	3.65E+03	3.10E+03	3.15E+02
F16	2.64E+03	2.41E+03	8.06E+01
F17	4.38E+09	3.05E+07	1.13E+05
F18	1.23E+06	7.32E+03	1.06E+04
F19	3.08E+03	2.67E+03	7.16E+01
F20	2.53E+03	2.43E+03	2.42E+02
F21	1.22E+04	5.97E+03	1.00E+02
F22	2.87E+03	2.77E+03	3.94E+02
F23	3.06E+03	3.12E+03	5.02E+02
F24	6.64E+05	2.91E+03	3.84E+02
F25	5.57E+03	4.72E+03	1.13E+03
F26	3.20E+03	3.19E+03	5.05E+02
F27	3.30E+03	3.18E+03	4.04E+02
F28	3.30E+12	1.12E+04	5.06E+02
F29	2.37E+12	1.39E+06	6.74E+03

5 Conclusion

This study presents an adaptive ABC algorithm called IABC-AP that automatically selects among 9 search strategies while the optimization process is running. The adaptive selection mechanism is based on a probabilistic operator selection technique that is called AP. The AP provides a way to update the selection probabilities of alternative operators based on their latest optimization performance.

To evaluate the performance of the presented algorithm, firstly, we compared the IABC-AP with the individual strategies from which it chooses. Adaptive selection mechanism has been shown to work successfully and outperform each strategy. We then make a second comparison with the original ABC and one of the up-to-date ABC versions that provide benchmark results on CEC 2017, namely culture-based ABC (CB-ABC). It was shown that the proposed algorithm clearly outperforms the original ABC. In addition, IABC-AP can generally produce lower mean error values than the CB-ABC algorithm produces.

Considering the experimental results, it can be concluded that the adaptive strategy selection method can contribute to the performance of ABC algorithms. Moreover, it increases the usability of algorithms by reducing the need for manual parameter settings.

Further research might concentrate on the adaptive selection of other ABC search strategies and the decision of their combinations to increase the optimization performance. Also, other low-level parameters such as the limit in ABC can be included in the adaptive process to fully automate the algorithm.

References

1. Aydin, D., Özyön, S., Yasar, C., Liao, T.: Artificial bee colony algorithm with dynamic population size to combined economic and emission dispatch problem. Int. J. Electr. Power Energy Syst. **54**, 144–153 (2014)
2. Karaboga, D., Okdem, S., Ozturk, C.: Cluster based wireless sensor network routing using artificial bee colony algorithm. Wirel. Netw. **18**(7), 847–860 (2012)
3. Akay, B., Karaboga, D.: Artificial bee colony algorithm for large-scale problems and engineering design optimization. J. Intell. Manuf. **23**(4), 1001–1014 (2012)
4. Aydogdu, I., Akın, A., Saka, M.P.: Design optimization of real world steel space frames using artificial bee colony algorithm with Levy flight distribution. Adv. Eng. Softw. **92**, 1–14 (2016)
5. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
6. Zhu, G., Kwong, S.: Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl. Math. Comput. **217**(7), 3166–3173 (2010)
7. Gao, W., Liu, S.: Improved artificial bee colony algorithm for global optimization. Inf. Process. Lett. **111**(17), 871–882 (2011)
8. Alatas, B.: Chaotic bee colony algorithms for global numerical optimization. Expert Syst. Appl. **37**(8), 5682–5687 (2010)
9. Aydın, D., Liao, T., De Oca, M.A.M., Stützle, T.: Improving performance via population growth and local search: the case of the artificial bee colony algorithm. In: International Conference on Artificial Evolution (Evolution Artificielle), pp. 85–96 (2011)
10. Eiben, Á.E., Hinterding, R.: Michalewicz, Z: Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. **3**(2), 124–141 (1999)
11. DaCosta, L., Fialho, A., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, pp. 913–920 (2008)
12. Gökalp, O., Ugur, A.: A high-level and adaptive metaheuristic selection algorithm for solving high dimensional bound-constrained continuous optimization problems. Turk. J. Elec. Eng. Comp. Sci. **28**(3), 1549–1566 (2020)

13. Fialho, Á., Ros, R., Schoenauer, M., Sebag, M.: Comparison-based adaptive strategy selection with bandits in differential evolution. In: International Conference on Parallel Problem Solving from Nature, pp. 194–203 (2010)
14. Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, pp. 1539–1546 (2005)
15. Awad, N.H., Ali, M.Z., Suganthan, P.N. et al.: Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization. Technical report. Nanyang Technological University, Singapore and Jordan University of Science and Technology, Jordan and Zhengzhou University, Zhengzhou China (2016)
16. Karaboga, D., Gorkemli, B., Ozturk, C., et al.: A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif. Intell. Rev. **42**, 21–57 (2014)
17. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
18. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
19. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge (1992)
20. Saad, E., Elhosseini, M.A., Haikal, A.Y.: Culture-based artificial bee colony with heritage mechanism for optimization of wireless sensors network. Appl. Soft Comput. **79**, 59–73 (2019)

Beetle Antennae Search Algorithm for the Motion Planning of Industrial Manipulator

Junwen Cui and Zhan Li

Abstract Beetle antennae search (BAS) algorithm as a computational intelligence algorithm has excellent nonlinear optimization ability applied in scientific and engineering applications. In this chapter, BAS is applied to the redundancy resolution of an industrial manipulator with dynamic joint velocity constraints by searching in high-dimensional space. The addressed application does not need to construct inverse kinematics equations in joint velocity level but directly uses forward kinematics to construct antennae fitness function. The experiment is practiced in the CoppeliaSim simulator for the industrial IIWA Kuka industrial manipulator to verify the performance.

Keywords Beetle Antennae Search · Motion planning · Redundancy resolution

1 Introduction

The Beetle Antennae Search (BAS) algorithm is inspired by longhorn beetles searching for food by tentacles [1] via two long antennae. In the beginning, the beetle does not know where the object is but can sense the intensity of the actual smell. Then, the long tentacles on both sides evaluate which side of the food smell is more vital. If the feeling of the antennae left is stronger than the right, the beetle will choose to fly to the left at a distance and then make another decision and movement. Based on this simple repetitive search behavior, the longhorn beetle can survive tenaciously in the natural environment.

The BAS has been developed in many fields due to its excellent performance and minimal configuration requirements, such as unmanned aerial vehicle path planning [2], pattern classification [3], route planning [4], geomechanics analysis [5], servo

J. Cui · Z. Li (✉)
University of Electronic Science and Technology of China, Chengdu, China
e-mail: zhan.li@uestc.edu.cn

J. Cui
e-mail: junwen_cui@126.com

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_4

controller parameter tuning [6], scheduling of power system [7], ship predictive collision avoidance [8], neural network training [3], bridge sensor optimal placement [9], and investment portfolio issue [10]. And, the BAS algorithm also has applications in the field of robotic manipulators. The termed trajectory-planning beetle swarm optimization (TPBSO) is proposed to resolve the redundant manipulator's point-to-point planning problem [11]. The beetle antennae olfactory recurrent neural network (BAORNN) with the concept of combining BAS and neural network realizes motion planning of redundant manipulator in task space [12].

The concept of this nature-inspired method is simplistic enough. In theory, it does not take up a lot of computing resources or occupy a great quantity of memory space. Moreover, it may not be required for specific application scenarios to construct a complex mathematical model. Associated with other methods such as the neural network [13], BAS does not require any training data sets, and the terminal numerical accuracy can be guaranteed due to its iterative search rules. As for the genetic algorithm (GA) [14], BAS reflects a more direct convergence direction without being transformed into another form to express. Moreover, for the particle swarm optimization (PSO) based method [15], the BAS does not require lots of particles to obtain a precise trajectory.

Several variants of BAS methods have been proposed to improve search efficiency, such as BAS without parameter tuning (BAS-WPT) [16] and beetle swarm antennae search (BSAS) [17]. The BAS-WPT is an improved modification of BAS that makes less dependency on parameter tuning, and it provides an approach to adding new or fused constraints-cost functions. The BSAS is chiefly through various beetles to make up for the shortcomings of a single iterative exploration. It requires all the beetles at the same time initially in a different direction, and then only one beetle is selected to move the actual step. Besides, there are other improvements, such as BAS with PSO [18] and BAS with going backward [19].

The redundant robots have the issue of redundancy resolution that leads to the highly-coupled nonlinearity of inverse kinematics. And the BAS seems to be an effective algorithm for managing such nonlinearity difficulties that have arisen in [20, 21]. In this chapter, a concise BAS algorithm is presented to accomplish accurate motion planning of redundant manipulators [22]. All the resolution is in the joint space considering the position of the end-effector to track a schemed curve, and the orientation is ignored. When the manipulator is going on a task, to ensure safety requirements in industrial environments, the joints' statuses are required to have dynamic limits that are angle and time-varying velocity boundaries. With these dynamic constraints, the verification experiments are taken applying a model of IIWA Kuka industrial manipulator in the CoppeliaSim/V-REP simulator [23].

2 Problem Formulation

To illustrate the non-linear optimization ability of the BAS algorithm, it will discuss in conjunction with the redundancy resolution of the Kuka industrial manipulator. The joint resolution is set constraints with variable joint velocity and angle that can be tricky and challenging. Specifically, the redundancy resolution problem is formulated, and the optimization framework with the BAS algorithm is addressed.

2.1 Kinematics Model

The forward kinematics of redundant manipulators is the mapping from joints to end-effector position in task space. And the corresponding simplistic expression [24] as

$$\mathbf{x}(t) = g(\boldsymbol{\theta}(t))) \tag{1}$$

where $\boldsymbol{\theta}(t) \in \mathbb{R}^n$ denotes the joint angle vector and $\mathbf{x}(t) \in \mathbb{R}^3$ denotes the end-effector position vector in the task space. $g(\cdot)$ means the forward kinematics that maps joint space to Cartesian space [24]. The manipulators may track the pre-designed trajectory that requires inverse kinematics to map the task space to the joint space. The inverse solution can be formulated as

$$\boldsymbol{\theta}(t) = g^{-1}(\mathbf{x}_d(t)) \tag{2}$$

where $\mathbf{x}_d(t)$ denotes the expected point of the pre-designed trajectory in the moment of t. $g^{-1}(\cdot)$ is the inverse kinematics function which is highly-coupled nonlinearity that can be hard to get the generalized analytic solution. And, the conventional way can be the Damped Least Square (DLS) method expressed as

$$\min_{\dot{\boldsymbol{\theta}}} \left\| J\dot{\boldsymbol{\theta}} - \dot{\mathbf{x}} \right\|_2^2 + \lambda^2 \left\| \dot{\boldsymbol{\theta}} \right\|_2^2 \tag{3}$$

where the $J = \partial g / \partial \boldsymbol{\theta} \in \mathbb{R}^{3\times n}$ is the Jacobian matrix and λ^2 denotes the damped term. However, it can be hard to solve with extra constraints such as the time-varying velocity limit. Hence, the new minimize formula with the concept of reducing the error between the actual endpoint and the desired is presented as

$$\min d(\mathbf{x}_d(t), \boldsymbol{\theta}(t)) = \|\mathbf{x}_d(t) - g(\boldsymbol{\theta}(t))\|_2^2 . \tag{4}$$

The formulation (4) is a minimization problem of nonlinear since the nonlinear $g(\cdot)$ formulation got by the chained homogeneous matrices make. The formulation can further be regarded abstractly as the antennae fitness function in the BAS algorithm.

2.2 Joint Constraints

The end-effector of the manipulators would be limited to a specific range of positions since the actual working area may be limited for some obstacles. Here it is simple to express the end-effector in the form of linear constraints in the task space as

$$\mathbf{x}_{\min} < \mathbf{x}(t) < \mathbf{x}_{\max} \tag{5}$$

where $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ denote the constraint boundaries of lower and upper. Also, the joint angle should be limited in some specific boundaries $\boldsymbol{\theta}_{\min}$ and $\boldsymbol{\theta}_{\max}$ as

$$\boldsymbol{\theta}_{\min} < \boldsymbol{\theta}(t) < \boldsymbol{\theta}_{\max}. \tag{6}$$

The movement of the manipulator may not be precise due to many factors such as the unknown working environment and uncertainty of itself. The joint velocity factor can not be ignored if to pursue the tracking accuracy. Hence, the joint velocity limits necessitate being considered as time-varying or time-dependent during the actual execution. The boundaries of the joint velocity are time-varying as

$$\dot{\boldsymbol{\theta}}_{\min} := \dot{\boldsymbol{\theta}}_{\min}(t), \quad \dot{\boldsymbol{\theta}}_{\max} := \dot{\boldsymbol{\theta}}_{\max}(t). \tag{7}$$

To increase the complexity of time-varying constraints, the boundaries are set like to the possible oscillation form presented as

$$\boldsymbol{\varphi}(t) = A + B\sin(wt) \tag{8}$$

where $\boldsymbol{\varphi}(t) = \dot{\boldsymbol{\theta}}_{\min} = -\dot{\boldsymbol{\theta}}_{\max}$ denotes the joint velocity boundaries. A and B are the offset and amplitude of the oscillation form. w denotes the oscillation frequency. In an advance declaration, the joint velocity boundaries are only utilized as a test method, and it is not a necessary form.

3 Beetle Antennae Search Application

The exploration space of the BAS algorithm is in the joint space of the Kuka industrial manipulator with configuration. The two antennas represent the searching directions which is a random configuration of normal joint angle vector. The intensity of the smell felt by antennas is set according to the tracking error of the end-effector. The search process is carried out iteratively based on the previous searching results. And, the specific design of the presented BAS method in the application of redundancy resolution of redundant manipulators will be expanded in the following subsections.

3.1 Searching Design

The searching strategy of the BAS algorithm is the main part that confirms the direction and step length. The searching will be executed in each instant time T_{inr}, and the beetle exploring space is in the joints angle space in which the dimension is n. The BAS algorithm requires the two antennas to decide which direction is to go, which can be formulated as

$$\begin{aligned} \boldsymbol{\theta}_{left}(t) &= F(\boldsymbol{\theta}(t) + \lambda(t)b) \\ \boldsymbol{\theta}_{right}(t) &= F(\boldsymbol{\theta}(t) - \lambda(t)b) \end{aligned} \tag{9}$$

where $\boldsymbol{\theta}_{left}(t) \in \mathbb{R}^n$ and $\boldsymbol{\theta}_{right}(t) \in \mathbb{R}^n$ denote the left antennas and the right for seeking the best joint angle. $F(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ denotes the searching mandatory limit function. $\lambda(t)\mathbb{R}$ denotes the antenna length, and $b \in \mathbb{R}^n$ denotes the random normal vector. The function $F(\cdot)$ is $f(\cdot) : \mathbb{R} \to \mathbb{R}$ with limit function as follows

$$f(\theta_i(t)) = \begin{cases} \theta_i(t), & \theta_{i,\min}(t) < \theta_i(t) < \theta_{i,\max}(t) \\ \theta_{i,\max}(t), & \theta_i(t) > \theta_{i,\max}(t) \\ \theta_{i,\min}(t), & \theta_i(t) < \theta_{i,\min}(t) \end{cases} \tag{10}$$

where $\theta_i(t) \in \mathbb{R}$ represents the ith element of $\boldsymbol{\theta}(t)$. The movement restriction here is preliminary, to avoid the possibility of a smaller fitness value through (4) when the joint angle $\boldsymbol{\theta}(t)$ exceeds the actual restrictions.

3.2 State Update

The BAS algorithm needs to update the state of the searching beetle in each iteration, which means the beetle will be in a new position after the movement. The updated state can be represented as

$$\boldsymbol{\theta}'(t + \Delta t) = F(\boldsymbol{\theta}(t) - \boldsymbol{\sigma}') \tag{11}$$

where

$$\boldsymbol{\sigma}' = \gamma(t) sgn[d(\mathbf{x}_d(t), \boldsymbol{\theta}_{left}(t)) - d(\mathbf{x}_d(t), \boldsymbol{\theta}_{right}(t))]b'. \tag{12}$$

$\boldsymbol{\theta}'(t + \Delta t) \in \mathbb{R}^n$ is the temporary joints state corresponding to the random direction vector b'. $\gamma(t) \in \mathbb{R}$ denotes the relative movement distance of the beetle. sgn represents the saturation sign function as follows

$$sgn(s) = \begin{cases} 1, & s > 0 \\ 0, & s = 0 \\ -1, & s < 0 \end{cases} \tag{13}$$

Next, the location of the beetle can be refreshed, and its fitness can be calculated according to the cost function (4). To ensure the convergence of tracking errors, it requires assessing that the chosen $\boldsymbol{\theta}'(t + \Delta t)$ can verify the current cost function value can be better than the previous step or the cost can be controlled in a scope. In the n steps, each updated $\boldsymbol{\theta}'(t + \Delta t)$ is recorded in a set with relevant the fitness, and the best option can be $\boldsymbol{\theta}(t + \Delta t)$ which is the actual resolved joint angle to implement.

3.3 *Exploration Range and Step Size*

When the exploration is at the start, the end-effector position error may be larger than the following stages. Hence, the step length and antenna exploration range parameter might be not optimal that requires these parameters to update with the going of exploration. The step length $\gamma(t)$ and the detection distance $\lambda(t)$ can be determined according to the cost described in (4). Take BAS-WPT [16] and BAORNN [12] as a reference, the updating of $\lambda(t)$ and $\gamma(t)$ can be

$$\lambda(t) = k_\lambda \sqrt{d(\mathbf{x}_d(t-1), \boldsymbol{\theta}'(t))} \tag{14}$$

$$\gamma(t) = k_\gamma \lambda(t) \tag{15}$$

where k_λ and k_γ are the tuning parameters for the convergence of the BAS algorithm. Moreover, the tuning of k_λ and k_γ should be carefully considered according to the sampling time T_{inr}.

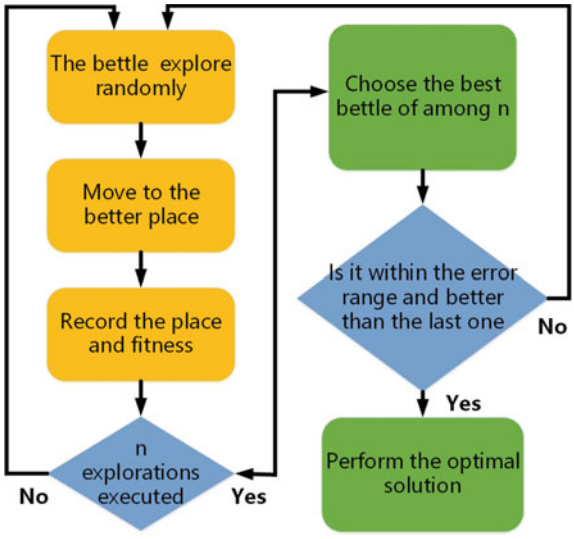


Fig. 1 Algorithm structure diagram of BAS with the application of motion planning

Algorithm 1 The BAS Algorithm for Motion Planning With Variable Joint Status Limit

Input:
The target point of the end-effector, $\mathbf{x}_d(t)$;
the forward kinematics of the redundant manipulator, $g(\boldsymbol{\theta}(t))$;
the fitness function, $d(\mathbf{x}_d(t), \boldsymbol{\theta}(t))$;
the convergence rate parameters, k_λ and k_γ;
the iteration times, n
Output:
Optimal joint angle $\boldsymbol{\theta}(t)$

$T \leftarrow$ Total execution time
$T_{inr} \leftarrow$ Time interval of planning
$\boldsymbol{\theta}(0) \leftarrow$ Joint angle at the start
$t \leftarrow 0$;
while $t < T$ **do**
  Get $\lambda(t)$ and $\gamma(t)$ according to (14) and (15)
  for $i \leq n$ **do**
    Initialize one beetle and randomly take the normalized direction vector, $b \in \mathbb{R}^n$
    Compute $\boldsymbol{\theta}_{left}$ and $\boldsymbol{\theta}_{right}$ with input of b according to (9), and constrain them within the boundary described in (10)
    Get the position of ith beetle in joint space by (11) and limit the value within the bounds with (10)
    Calculate moving speed of ith beetle and constrain it through (8)
    Record the fitness calculated by (4) and position of ith beetle
    Renew the position of ith beetle
    $i \leftarrow i + 1$
  end for
  Choose the best beetle of n according to the recorded fitness
  if *the best beetle is within the allowable error range and better than the last one* **then**
    Running the actual joint angle $\boldsymbol{\theta}(t)$ represented by the best beetle
    $t \leftarrow t + T_{inr}$
  end if
end while

The process diagram of the presented BAS algorithm for motion planning of redundant manipulator with the time-varying joint status limit is shown in Fig. 1. Theorem 1 in [25] has proved the convergence of the primary BAS. In Theorem 1, the proper step length can make sure of the asymptotic convergence of the BAS. On the other hand, the redundant resolution of the industrial manipulator in this work can also extend the evidence of its convergence in [25].

4 Experiment Results

With the given trajectory, the redundant manipulator tracks the next desired point in 0.2 s time interval T_{inr} in the experiment, and the entire track exercise time is 40 s.

The presented algorithm is built and run in MATLAB (R2017a) on a laptop with the CPU is Intel i5-8400. And, the experiment platform is chosen in CoppeliaSim/V-REP with the robot model of the IIWA Kuka manipulator. The simulation is performed synchronously between MATLAB and CoppeliaSim by communication.

4.1 Implementation on Kuka Industrial Manipulator

The IIWA Kuka manipulator is tested by tracking some regular trajectories, which can be a circle and a figure-eight curve to verify and appraise the performance of the presented BAS algorithm in the application of motion planning of redundant manipulator. The trajectories and time equation of the circle path and figure-eight curve are

$$\mathbf{x}_{d,c}(t) = \mathbf{x}_0 + \cos(\frac{2\pi t}{T})A_x + \sin(\frac{2\pi t}{T})A_y \tag{16}$$

$$\mathbf{x}_{d,f}(t) = \mathbf{x}_0 + \cos(\frac{2\pi t}{T})A_x + \sin(\frac{4\pi t}{T})A_y \tag{17}$$

where $\mathbf{x}_0 \in \mathbb{R}^3$ denotes the center of the circle and the figure-eight curve. $A_x \in \mathbb{R}^3$ and $A_y \in \mathbb{R}^3$ denote the parameter matrices projecting the shape in different directions with different length.

The center of the circle and the figure-eight curve is set as $\mathbf{x}_0 = [0.3 \;\; 0.3 \;\; 0.9]^T$ in the base frame. The parameter matrices are set as $A_x = [0.2 \;\; 0 \;\; 0]^T$ and $A_y = [0 \;\; 0.2 \;\; 0]^T$. The actual joint angle limit is $-\frac{\pi}{2} < \theta_i(t) < \frac{\pi}{2}$. The joint time-varying velocity boundaries are set as $\dot{\boldsymbol{\theta}}_{min} = -1 - 0.1\sin(t)$ rad/s and $\dot{\boldsymbol{\theta}}_{max} = 1 + 0.1\sin(t)$ rad/s. And, the beetle exploration parameters are set as $n = 25$, $k_\lambda = 2$ and $k_\gamma = 20$. The Kuka manipulator is set with the initial joint position as $\boldsymbol{\theta}(0) = [\frac{\pi}{8} \;\; \frac{\pi}{6} \;\; 0 \;\; -\frac{\pi}{3} \;\; 0 \;\; -\frac{\pi}{8} \;\; -\frac{\pi}{6}]^T$, then to employ motion planning with the BAS iteration.

4.1.1 Circle Tracking

In the case of tracking a schemed circle, the comprehensive simulation performance is presented in Fig. 2. It can be seen that the end-effector trajectory is approximately coincident compared with the schemed circle, and the transition between limb trajectories is even. The joints' angles are all within boundaries and appear to be mostly smooth. The position error of the end-effector shrinks quickly, although the initial position is not on the start point. Due to the large gap between the initial point and the desired point, the joint velocity shows oscillating between the limit bounds. And, the strategy drives the manipulator with joint velocity approaching the limit only in the

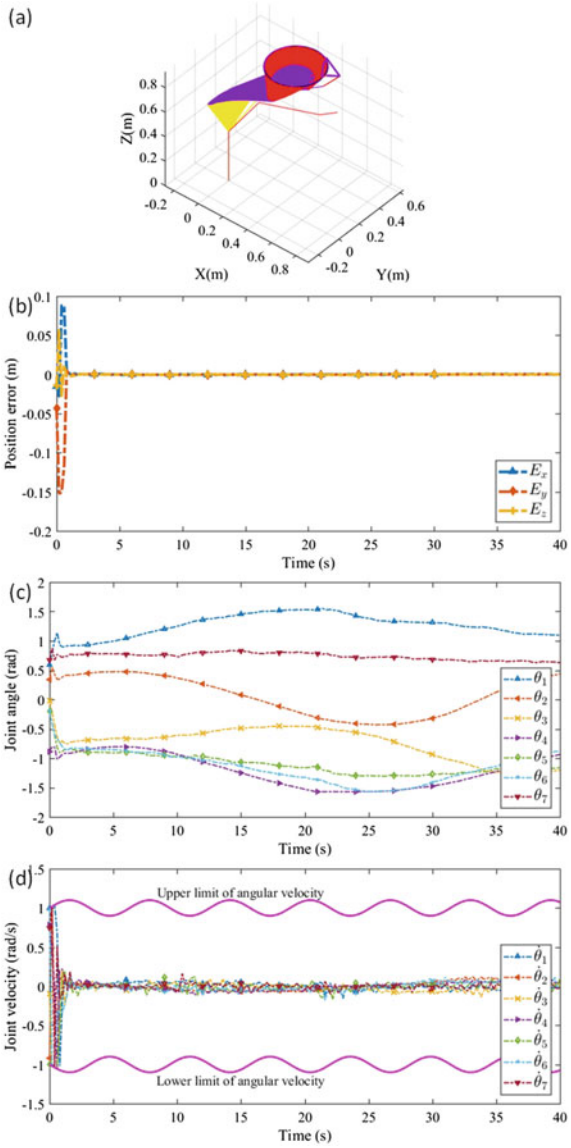


Fig. 2 The synthesized results of tracking a circle of the Kuka manipulator Under the constraints of angle and velocity by the BAS algorithm. **a** The trajectory of limbs and end-effector. **b** Tracking error of the end-effector. **c** Resolved joint angle. **d** Resolved joint velocity

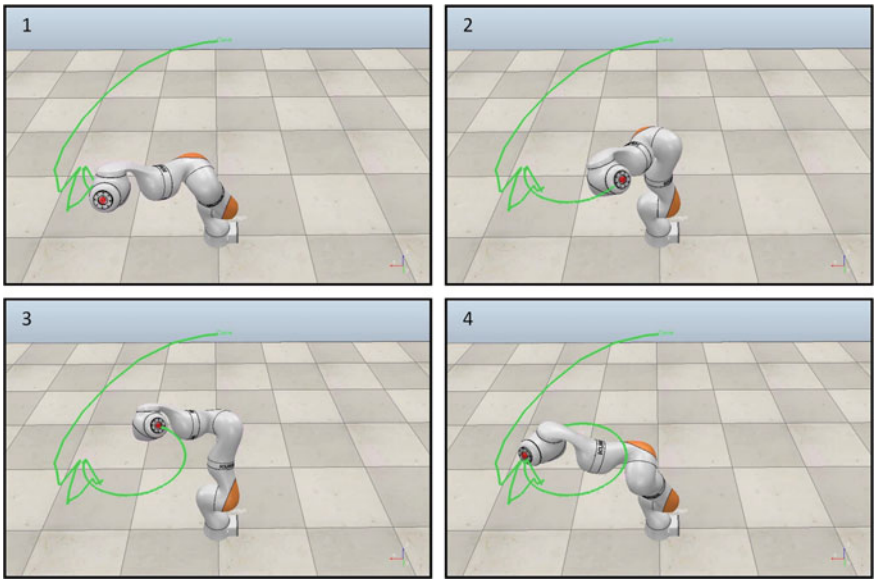


Fig. 3 The simulating snapshots of the IIWA Kuka manipulator for drawing a circle through the presented method

beginning, then it turns to mitigate since the tracking is ensured. The performance presented by the presented method shows the good motion planning ability of the redundant manipulator.

Moreover, the test is also be taken in the CoppeliaSim/V-REP platform for verification in a simulated physical environment. The snapshots of the operating process of the Kuka manipulator for tracking a circle are shown in Fig. 3. The irregular trajectory in the scene frame is because the circular trajectory is directly tracked at the beginning, and there is a gap between the actual start point and the circle. From the nearly complete arc trajectory, it confirmed that the presented BAS algorithm works expectantly.

4.1.2 Figure-Eight Curve Tracking

The experiment is also taken to track a figure-eight curve with the same beetle exploration parameters to illustrate that the circle curve tracking is not a special redundancy resolution case. The corresponding results are shown in Fig. 4.

The performance is approximate with the same features, such as the position error is fast convergent, joint angle and joint velocity are all in bonds, and joint velocity is stable after the initial short duration. The snapshots of this experiment

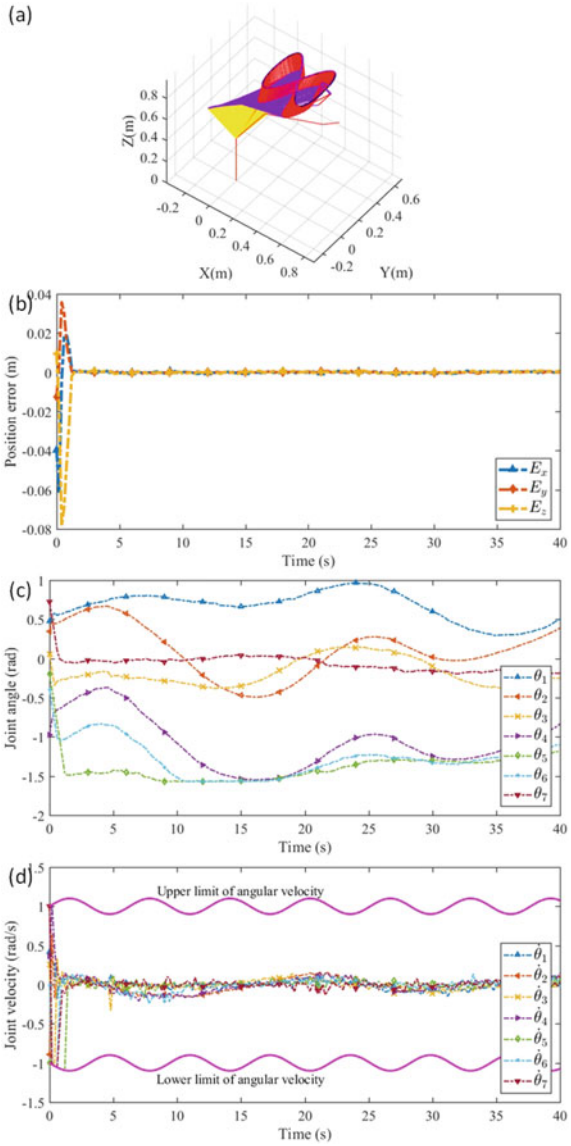


Fig. 4 The synthesized results of tracking a figure-eight curve of the Kuka manipulator Under the constraints of angle and velocity by the BAS algorithm. **a** Trajectory of limbs and end-effector. **b** Tracking error of the end-effector. **c** Resolved joint angle. **d** Resolved joint velocity

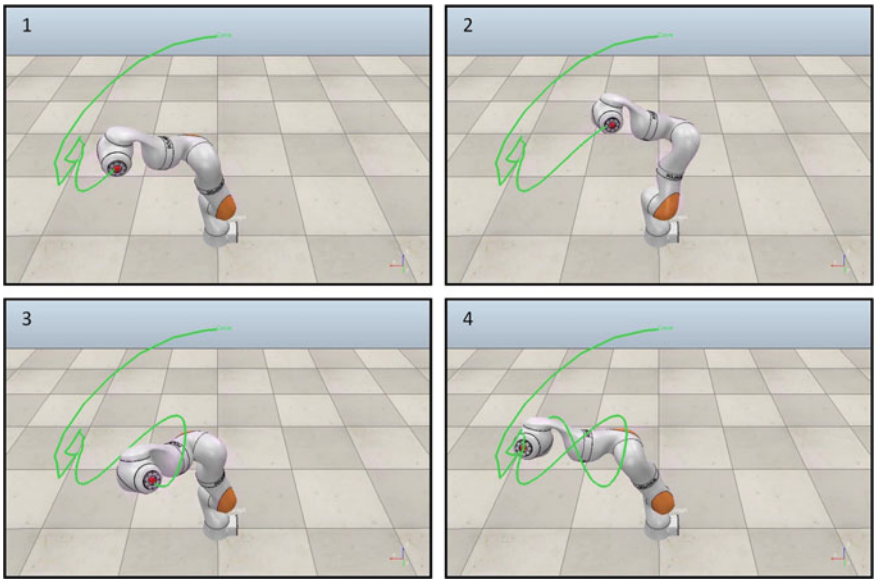


Fig. 5 The simulating snapshots of the IIWA Kuka manipulator for drawing a figure-eight curve through the BAS algorithm

in the CoppeliaSim platform are presented in Fig. 5. It also can be seen that the presented method reforms the trajectory of the end-effector to be similar sufficient to expectations.

4.2 Evaluation of Different Exploration Parameters

The BAS algorithm has several regulator parameters that may affect the exploration of the beetle. In this application, the main factors are the iteration index n and convergence step length k_λ and k_γ. The influence characteristics of each parameter will be considered and discussed separately.

4.2.1 Exploration Times *n*

The exploration times describe how long the journey is that the beetle will search in the joint space. The longer journey means more computation costs that will cause the execution time of the resolution loop longer. Of course, for the presented BAS application, the one search journey can be finished when the conditions are met, which is that the renewed fitness function value is less than the last search and within the error scope. The remaining parameters are set to be feasible and constant for

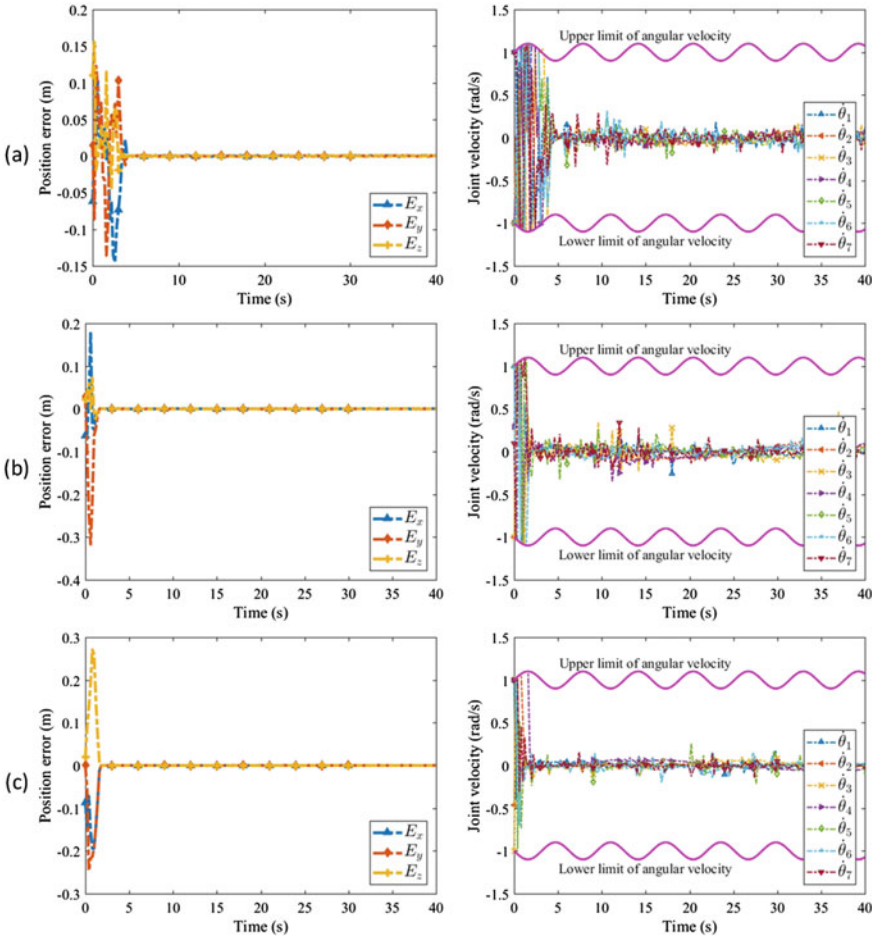


Fig. 6 The convergence results of tracking a circle with different explorations parameter n compared in position error of end-effector and joint velocity. **a** The performance with $n = 1$. **b** The performance with $n = 50$. **c** The performance with $n = 200$

the comparison of different exploration times. Considering the ability to converge and time-varying velocity constraints, the outcomes are shown in Fig. 6 with $n = 1$, $n = 50$, and $n = 200$. The whole duration of motion planning of different parameters are recorded as presented in Table 1 to evaluate the time costs of probable journeys.

It can be seen that the convergence of position error becomes faster comparing the case of $n = 1$ and $n = 50$, and is not obvious for $n = 50$ and $n = 200$. It is easy to imagine that the resolution will be more precise with the increase of search journey, but probably not be suitable for a particularly long journey. The beetle may hover around an excellent point, and the motion range is dependent on k_λ and k_γ. On the other hand, the duration of large oscillations of the joint velocity is reduced with the

Table 1 The BAS algorithm running time of different exploration parameters n with conditions of $k_\lambda = 2$ and $k_\gamma = 20$

n	Bout 1 (s)	Bout 2 (s)	Bout 3 (s)	Bout 4 (s)	Bout 5 (s)	Average (s)
1	0.63	0.77	0.92	1.11	0.73	0.83
50	1.16	1.16	1.05	1.21	1.05	1.13
200	3.18	3.20	3.24	3.07	3.12	3.16

increased n. According to Table 1, the entire time spent will increase as the journey increases, which will take up more computing resources. Finally, the configuration of n should be considered the accuracy condition and cost of time and calculation.

4.2.2 Tuning Parameters k_λ

The coefficient parameter k_λ of the exploration distance expounds the strength of exploration. The effects of position error and joint velocity of different k_λ are simulated as shown in Fig. 7. And, the corresponding time cost of the whole duration of different k_λ is evaluated multiple times, and results are given in Table 2.

In the case of the selected other parameters unchanged, the total duration time is reduced obviously with a proper increase of k_λ. When the k_λ is too large, the whole time cost seems to be higher. Since the search step size is too long, fast convergence may not be guaranteed. However, the joint velocity shows more periodic oscillation that may make the manipulator tremble. And, the convergence of position error seems to be longer. Integrate the experimental results, the deployment of k_λ requires the thoughtful consideration of the actual situation to make sure the fast convergence.

4.2.3 Tuning Parameters k_γ

The coefficient parameter k_γ directly affects the distance of beetle moving in one exploration step. Similar to k_λ, the evaluation is likewise taken with there cases of k_γ are shown in Fig. 8 and Table 3.

It is straightforward to obtain that the larger k_γ will result in faster operation of the entire duration. And the joint velocity shows more extreme values of shock in the beginning and holds a long time compared to the smaller k_γ. Of course, the configuration of k_γ also needs to be in the appropriate range, and it depends on the specific problem and the configuration of other parameters.

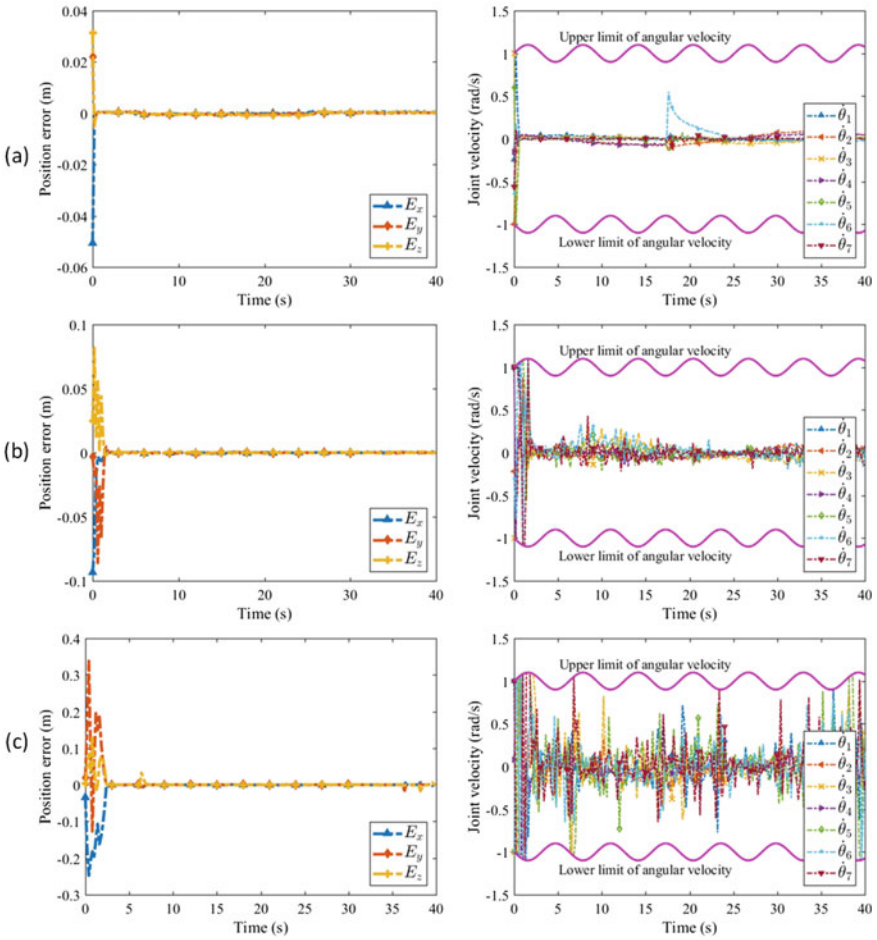


Fig. 7 The convergence results of tracking a circle with different explorations parameter k_λ compared in position error of end-effector and joint velocity. **a** The performance with $k_\lambda = 0.5$. **b** The performance with $k_\lambda = 8$. **c** The performance with $k_\lambda = 16$

Table 2 The BAS algorithm running time of different exploration parameters k_λ with conditions of $n = 50$ and $k_\gamma = 5$

k_λ	Bout 1 (s)	Bout 2 (s)	Bout 3 (s)	Bout 4 (s)	Bout 5 (s)	Average (s)
0.5	5.39	4.42	4.57	6.60	4.24	5.04
8	1.12	0.97	1.03	0.93	1.04	1.02
16	2.32	2.70	2.18	2.40	2.29	2.38

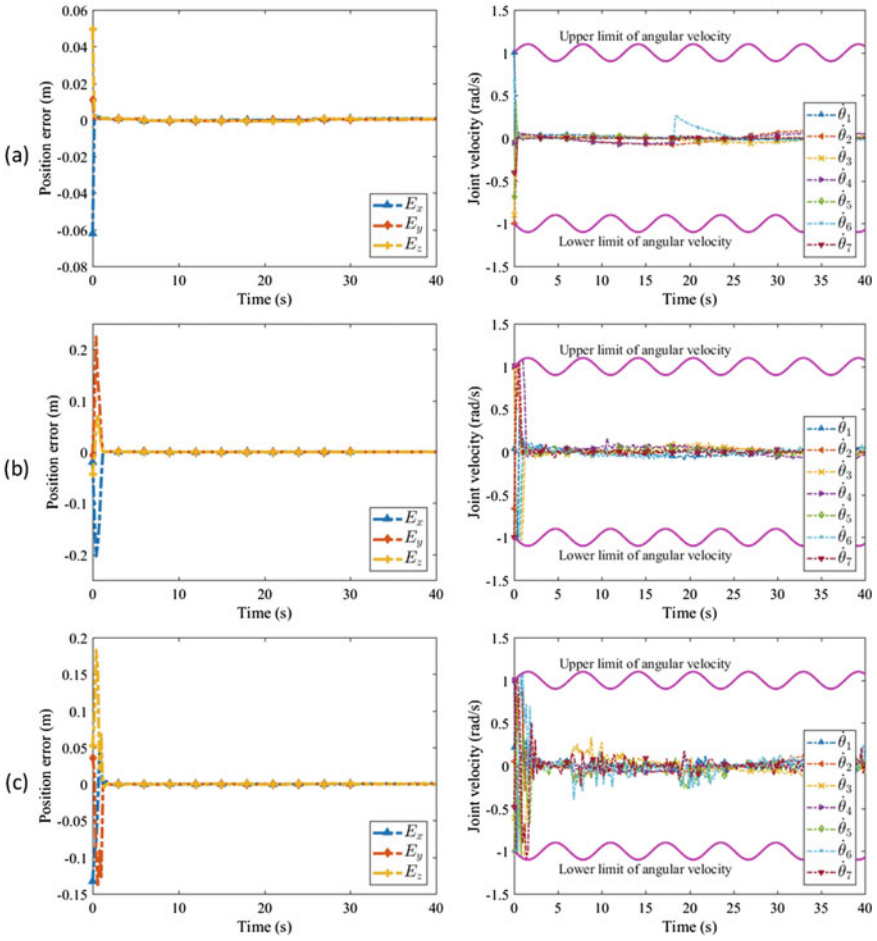


Fig. 8 The convergence results of tracking a circle with different explorations parameter k_γ compared in position error of end-effector and joint velocity. **a** The performance with $k_\gamma = 0.5$. **b** The performance with $k_\gamma = 8$. **c** The performance with $k_\gamma = 16$

5 Conclusion

The BAS algorithm is inspired by the biological strategy of beetle that can be employed in the optimized problem, which can be very tricky for some general methods. In the case of motion planning of redundant manipulator, the BAS algorithm shows the excellent performance of redundant resolution with time-varying joint velocity constraints. And, the presented method barely requires the forward kinematics to build a fitness function that decides the moving direction. The simulations on Kuka industrial manipulator for tracking two regular geometric trajectories

Table 3 The BAS algorithm running time of different exploration parameters k_γ with conditions of $n = 50$ and $k_\lambda = 2$

k_γ	Bout 1 (s)	Bout 2 (s)	Bout 3 (s)	Bout 4 (s)	Bout 5 (s)	Average (s)
0.5	8.21	8.75	7.55	7.16	8.47	8.03
8	1.66	1.49	1.69	1.73	1.64	1.64
16	1.39	1.32	1.14	1.24	1.17	1.25

have demonstrated the efficiency of the presented method for motion planning of redundant manipulator, and reveal the reliable performance of the presented method.

References

1. Jiang, X., Li, S.: BAS: Beetle Antennae Search algorithm for optimization problems (2017). arXiv:1710.10724
2. Wu, Q., Shen, X., Jin, Y., et al.: Intelligent Beetle Antennae Search for UAV sensing and avoidance of obstacles. Sensors **19**, 1758 (2019)
3. Wu, Q., Ma, Z., Xu, G., et al.: A novel neural network classifier using beetle antennae search algorithm for pattern classification. IEEE Access **7**, 64686–64696 (2019)
4. Mu, Y., Li, B., An, D., Wei, Y.: Three-dimensional route planning based on the beetle swarm optimization algorithm. IEEE Access **7**, 117804–117813 (2019)
5. Sun, Y., Zhang, J., Li, G., et al.: Optimized neural network using beetle antennae search for predicting the unconfined compressive strength of jet grouting coalcretes. Int. J. Numer. Anal. Meth. Geomech. **43**, 801–813 (2019)
6. Fan, Y., Shao, J., Sun, G.: Optimized PID controller based on Beetle Antennae Search algorithm for electro-hydraulic position servo control system. Sensors **19**, 2727 (2019)
7. Li, Q., Wang, Z., Wei, A.: Research on optimal scheduling of wind-PV-hydro-storage power complementary system based on BAS algorithm. IOP Conf. Ser.: Mater. Sci. Eng. **490**, 072059 (2019)
8. Xie, S., Chu, X., Zheng, M., Liu, C.: Ship predictive collision avoidance method based on an improved beetle antennae search algorithm. Ocean Eng. **192**, 106542 (2019)
9. Yang, J., Peng, Z.: Beetle-swarm evolution competitive algorithm for bridge sensor optimal placement in SHM. IEEE Sens. J. **20**, 8244–8255 (2020)
10. Chen, T., Zhu, Y., Teng, J.: Beetle swarm optimisation for solving investment portfolio problems. J. Eng. **2018**, 1600–1605 (2018)
11. Wang, L., Wu, Q., Lin, F., et al.: A new trajectory-planning beetle swarm optimization algorithm for trajectory planning of robot manipulators. IEEE Access **7**, 154331–154345 (2019)
12. Khan, A., Li, S., Luo, X.: Obstacle avoidance and tracking control of redundant robotic manipulator: An RNN-based metaheuristic approach. IEEE Trans. Ind. Inf. **16**, 4670–4680 (2020)
13. Braspenning, P.: Artificial Neural Networks. Springer, Berlin (1995)
14. Sivanandam, S., Deepa, S.: Introduction to Genetic Algorithms. Springer, Berlin (2008)
15. Parsopoulos, K., Vrahatis, M.: Particle Swarm Optimization and Intelligence. Information Science Reference, Hershey, PA (2010)
16. Jiang, X., Li, S.: Beetle Antennae Search without Parameter Tuning (BAS-WPT) for multi-objective optimization. arXiv:1711.02395
17. Wang, J., Chen, H.: BSAS: Beetle Swarm Antennae Search algorithm for optimization problems (2018). arXiv:1807.10470

18. Lin, M., LI, Q.: A hybrid optimization method of beetle antennae search algorithm and particle swarm optimization. DEStech Trans. Eng. Technol. Res. (2018)
19. Wu, Q., Lin, H., Jin, Y., et al.: A new fallback Beetle Antennae Search Algorithm for path planning of mobile robots with collision-free capability. Soft. Comput. **24**, 2369–2380 (2019)
20. Zou, D., Chen, P., Liu, K.: Indoor location algorithm based on the search optimization of the beetle. J. Hubei Univ. Technol. **34**, 427–431 (2018)
21. Chen, X., Liu, H., Liu, Y., et al.: Research on cable water resistance based on BAS-BP model. J. Hubei Univ. Technol. **34**, 35–39 (2019)
22. Cheng, Y., Li, C., Li, S., Li, Z.: Motion planning of redundant manipulator with variable joint velocity limit based on Beetle Antennae Search Algorithm. IEEE Access **8**, 138788–138799 (2020)
23. Robot simulator: CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics (2020). https://www.coppeliarobotics.com/
24. Li, Z., Li, C., Li, S., Cao, X.: A fault-tolerant method for motion planning of industrial redundant manipulator. IEEE Trans. Ind. Inf. **16**, 7469–7478 (2020)
25. Zhang, Y., Li, S., Xu, B.: Convergence analysis of Beetle Antennae Search Algorithm and its applications (2019). arXiv:1904.02397

Solving Optimal Power Flow with Considering Placement of TCSC and FACTS Cost Using Cuckoo Search Algorithm

Benyekhlef Larouci, Houari Boudjella, Ahmed Nour El Islam Ayad, and Abdelkader Si Tayeb

Abstract The main objective of an optimal power flow techniques is to obtain a steady-state operating point that minimizes the total cost of production, losses, and maintains the power system in an acceptable performance in terms of physical limits of electrical network equipment, such as generators, transmission lines, transformers and compensator shunt. The contribution of this paper is to present an efficient and reliable nature-inspired Cuckoo Search (CS) algorithm for nonlinear constrained optimization to solve the optimal power flow (OPF) problems with the installation of the FACTS devices, considering TCSC cost. Installation of FACTS devices in power systems, increases the power transfer capability, guarantees transient stability, reduces transmission losses, and minimizes the fuel cost of generation. Our method employs a cuckoo search algorithm to find the optimal set of control variables. To prove the effectiveness and the robustness of the proposed approach, CSA is applied on the standard IEEE 9-bus test system for minimizing fuel cot including TCSC cost, with consideration of optimal location and sizing of TCSC devices.

Keywords OPF · Cuckoo search (CSA) · TCSC · FACTS cost

B. Larouci · H. Boudjella (✉) · A. N. E. I. Ayad
Department of Electrical Engineering, Faculty of Applied Sciences, University Kasdi Merbah, Ghardaia Road, BP 511, 30000 Ouargla, Algeria
e-mail: boudjella.houari@univ-ouargla.dz

B. Larouci
e-mail: larouci.benyekhlef@univ-ouargla.dz

A. N. E. I. Ayad
e-mail: ayad.ahmed@univ-ouargla.dz

A. Si Tayeb
Applied Research Unit On Renewable Energies "URAER Ghardaia", 47133 Ghardaïa, Algeria

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_5

1 Introduction

The FACTS technology represents a new potential to control power and improve the usable capacity in the present, as well as new and upgraded lines. The FACTS devices can increase power flow in lines closer to their thermal rating [1]. Within the basic system security guidelines, the FACTS devices enable the transmission lines to obtain one or more benefits [2].

Optimal power flow (OPF), with the integration of the FACTS devices, is a nonlinear complex optimization problem, where the steady-state parameters of an electrical network need to be determined for its economical and efficient operation. The complexity of the problem increases with the pervasive presence of the constraints. Solving OPF remains a popular but difficult task among electrical systems researchers [3].

Different sorts of constraints or multiobjective are regrouped in OPF problems [4]. OPF solutions include both traditional methodologies and artificial intelligence-based methods [5]. Traditional methods are known as deterministic or classical optimization methods, which are based on mathematical programming approaches and are used to solve different sizes of OPF problems. To meet the requirements of different objective functions and types of constraints, various popular deterministic techniques are used in the literature, such as Linear Programming, Gradient methods, Quadratic Programming [6], Newton Raphson, Nonlinear Programming, and Interior Point [7].

Several attempts to address the constraints of the mathematical programming methodologies, such as intelligent or meta-heuristic optimization methods [4], have recently been examined. The intelligent methods mainly include Genetic algorithm [8], Particle Swarm Optimization [9], Search Group Algorithm [10], Ray Optimization [11], Big Bang-Big Crunch Algorithm[12], Mine Blast [13], Imperialist Competitive Algorithm [12], Firefly algorithm [14], Dolphin Echolocation [15], Bat-Inspired [16], Teaching–Learning-Based Optimization [17].

In this work, a new meta-heuristic search algorithm inspired by nature, named Cuckoo Search (CS), is proposed to solve specifically the optimal power flow and the security constraints optimal power flow problem with the incorporation of FACTS devices.

2 Problem Formulation

2.1 Economic Load Dispatch

The cost of a thermal generator unit is generally a quadratic function in a traditional economic dispatch, and the dispatch center aimed to minimize the total costs of all generation units [18].

$$Min\ C(P_{gi}) = \sum_{i=1}^{N_g} \left(a_i P_{gi}^2 + b_i P_{gi} + c_i\right) \tag{1}$$

where, $C(P_{gi})$ is the total costs ($/h) of ith generator with output P_{gi}, N_g is the number of generating units. a_i, b_i and c_i are the coefficients cost of ith generator.

Equation (1) subject to:

Inequality constraints: Generation power should be within the minimum and the maximum output.

$$P_{gi}^{min} \leq P_{gi} \leq P_{gi}^{max} \tag{2}$$

Equality constraints: The total generation should be sufficient to fulfill the total demand P_D and power losses P_L. The transmission losses are assumed zero or constant (independent of the unit outputs) [19]:

$$\sum_{i=1}^{N_g} P_{gi} = P_D + P_L \tag{3}$$

2.2 Optimal Power Flow Dispatch

The OPF problem can be expressed mathematically as follow:

$$Min\ f(x,u) \tag{4}$$

$$S.t : \begin{cases} g(x,u) = 0 \\ h(x,u) \leq 0 \end{cases} \tag{5}$$

where:

$f(x,u)$ is the objective function.

$g(x,u)$ consists of the equality constraint of the system and is always active during the optimization process of the power flow equations, as shown below [20, 21]:

$$P_{gi} - P_{dl} - \sum_{j=1}^{N} V_i V_j Y_{ij} \cos(\delta_i - \delta_j - \delta_i - \theta_{ij}) = 0 \tag{6}$$

$$Q_{gi} - Q_{dl} - \sum_{j=1}^{N} V_i V_j Y_{ij} \sin(\delta_i - \delta_j - \delta_i - \theta_{ij}) = 0 \tag{7}$$

$h(x, u)$ consists of the inequality constraints, such as the thermal limit of the lines, active and reactive power generation [22, 23].

x is the vector of state variables presented by the dependent quantities of the control variables.

$$x = [\delta \ V_L]^T \tag{8}$$

u is the vector of control variables presented by the independent quantities of the control variables [24]. The set of control variables, also called decision variables, which can control the power flow in the power system, is represented by the following vector:

$$u = \left[P_{gi} \ V_{gi} \ P_l \ Q_{FACTS} \ \ldots\right]^T \tag{9}$$

For optimal active power dispatch, the objective function is total generation cost as expressed as follows:

$$Minimize \ \sum_{i=1}^{Ng} \left(a_i + b_i \ P_{gi} \ + \ c_i \ P_{gi}^2\right) \tag{10}$$

2.3 Optimal Power Flow with FACTS Devices Cost (TCSC)

Optimal Power Flow (OPF) is a very important tool in planning and controlling modern power systems operations. In an OPF, the values of some or all of the control variables must be determined in order to optimize (minimize or maximize) a preset objective [25]. Flexible Alternative Current Transmission System controllers (FACTS), may modify network parameters quickly and efficiently to improve system performance [26]. These devices are utilized to enhance the dynamic performance of power systems stability, while also increasing power transfer capability and steady-state voltage profile [27].

The mathematical models of the FACTS devices are developed mainly to perform steady-state research. Therefore, the Thyristor Controlled Series Compensator (TCSC) is modeled to modify the transmission's reactance directly [28].

The TCSC device consists of three main components: Capacitor bank, bypass inductor, and bidirectional Thyristor SCR1 and SCR2 as shown in Fig. 1 [29].

The function of the TCSC is to alter the value of the transmission line reactance by adding either a capacitive or inductive component to the main transmission line reactance as shown in Fig. 2 [30].

The reactance of the line where TCSC is placed is given by:

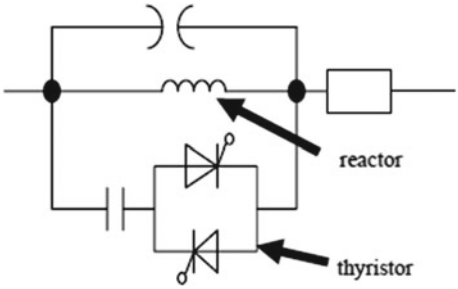


Fig. 1 Circuit diagram of TCSC

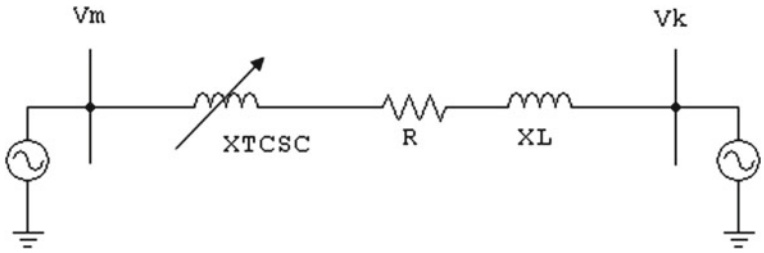


Fig. 2 Effect of TCSC on transmission line reactance

$$X_{mk} = X_L + X_{TCSC} \tag{11}$$

where: X_L is the reactance of the transmission line and X_{TCSC} is the reactance of TCSC. To evade overcompensation, the operating limits of TCSC (X_{TCSC}) are selected between *–0.7.* X_L and *0.2.* X_L [31].

After installing TCSC on the line between bus m and bus k of a general power system, the new system admittance matrix $\overrightarrow{Y}'_{bus}$ can be computed as [32]:

$$\vec{Y}'_{bus} = \vec{y}_{mk} + \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \Delta\vec{y}_{mk} & 0 & \dots & 0 & -\Delta\vec{y}_{mk} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -\Delta\vec{y}_{mk} & 0 & \dots & 0 & \Delta\vec{y}_{mk} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix} \tag{12}$$

$\overrightarrow{y}_{mk}$: Admittance matrix without presence of TCSC.

For the reason that Y_{bus} has to be update for each of the different locations and the amount of compensation of TCSC, Eq. (12) is applied at each iteration (Fig. 3).

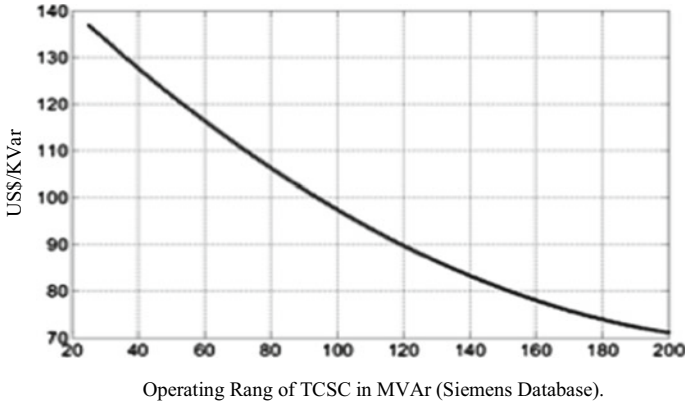


Fig. 3 Variation of the investment costs for TCSC in terms of operating rang

The cost function C_{TCSC} in ($/KVar) displays the installation cost of the TCSC device in the network, which is calculated by the following equation:

$$C_{TCSC} = 153.75 - 0.7130\, S_{TCSC} + 0.0015\, S_{TCSC}^2 \tag{13}$$

where: S_{TCSC} is the operating range of the TCSC device in M_{var}.

The total cost of generation including the cost of FACTS controllers, can be formulated as follows [33]:

$$Min\ C_{Total} = C_1\left(P_{gi}\right) + C_2(f) \tag{14}$$

$$S.t \begin{cases} E_1(f,g) = 0 \\ B_1(f) \rangle 0\ , \quad B_2(g) \rangle 0 \end{cases} \tag{15}$$

where:

C_1: Total generation costs given in Eq. (8).

C_2: Average investment costs of FACTS devices given in Eq. (13).

C_{Total}: Overall cost of objective function.

E_1: Equality constraints with respect to active and reactive power flow.

B_1, B_2: Inequality constrains for FACTS devices and power flow.

f, P_{gi}: are the variables of FACTS devices and real power generated.

The generation cost and the investment costs of FACTS devices are given in (US$/h) and (US$), respectively. They must be unified into US$/h [4, 34]. In this paper, three years are applied to evaluate the cost function. Therefore, the average value of the investment costs is calculated using the following equation:

$$C_1(f) = \frac{C(f)}{8760 \times 3} \ (\$/h) \tag{16}$$

C(f): is the total investment costs of FACTS devices.

3 Cuckoo-Inspired Metaheuristics

The Cuckoo Search Algorithm (CSA) is one of the most popular metaheuristic algorithms inspired by nature, developed in 2009 by Yang and Deb [35, 36]. The research of Cuckoo is based on the parasitism of the nests of certain species of Cuckoo. Furthermore, the proposed algorithm is reinforced by so-called Levy flights, rather than by simple isotropic random steps.

In 2011, Rajabioun proposed a Cuckoo metaheuristic optimization method based on a population of solutions (CSA). The pioneer of the Cuckoo optimization technique drew on the behavior of Cuckoos in their life, reproduction, and development to propose a new evolutionary metaheuristic [37].

The original version of the CSA is based on the Lévy flight mechanism. Indeed, it operates in continuous research spaces and allows real-type solutions to be found. However, binary optimization problems require binary-type solutions. To extend the CSA to binary spaces, in 2012, Gherboudj, proposed a binary version, which she named BCS "Binary Cuckoo Search" [37].

In 2014, Ouaarab, proposed a new version of CS, named "Improved Cuckoo Search" or improved CS, the Cuckoo as the first level of intensification and diversification control, and since this Cuckoo is an individual of a population, then, this population is qualified to be the second level of control. The idea of the enhancement is to reorganize the population by incorporating a new type of Cuckoo that is smarter and more efficient in its search, compared to other Cuckoos. [38].

The process of adapting metaheuristics initially developed to solve continuous optimization problems to combinatorial optimization problems is generally difficult. A. Ouaarab, tried to adapt CSA to a combinatorial optimization problem. The result is a discrete version of CS called "Discrete Cuckoo Search (DCS)" [38].

Figure 4 represents a summary of the different methods inspired by Cuckoo that are:

3.1 Cuckoo Search Algorithm (CSA)

The CSA is a famous and recent, nature-inspired approach for solving nonlinear optimization problems. Previous research published in the literature, shows that the cuckoo algorithm, is very promising and may overtake existing algorithms such as genetic algorithms, PSO and neural networks.

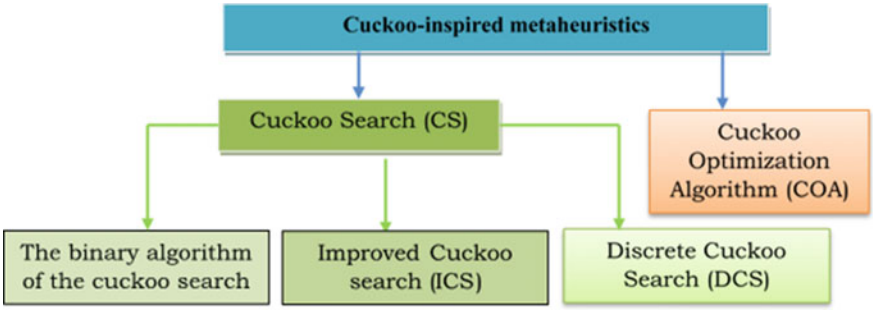


Fig. 4 Cuckoo-inspired metaheuristics methods

The proposed algorithm is based on the aggressive reproduction strategy of certain species of Cuckoo birds in combination with the random Lévy flight process, which is inspired, by the flight system of fruit flies [35].

Lévy flight

The French mathematician Paul Pierre Lévy, one of the founders of modern probability theory, proposed the Lévy flight. Since its creation, the flight of Lévy has given theoretical interpretations to several physical, chemical, biological and natural phenomena. Lévy's flight makes it possible to model random walks made up of a large number of steps where the transitions are based on probabilities.

In mathematical terminology, Lévy's flight (Fig. 5) is a random walk in which the distance between the steps has a probability distribution with a heavy tail: whose tails are not bounded exponentially [36].

Where:

Random walk: a mathematical formalization of a trajectory composed of a set of random steps.

Probability distribution: a function that represents the probability of a random number to take a given value.

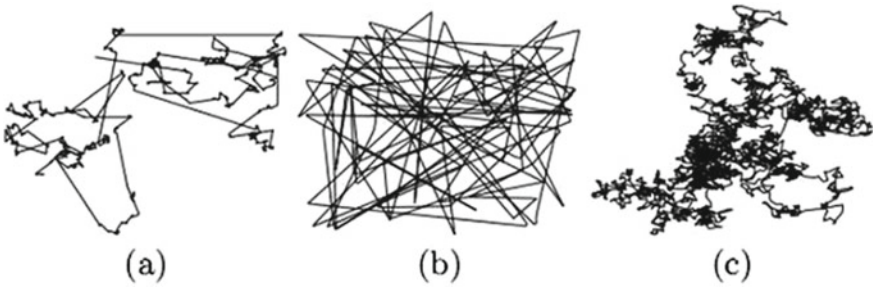


Fig. 5 Levy flight (**a**), simple random walk (**b**) and Brownian motion (**c**)

The jumps of the random steps of Levy Flight are distributed according to the Levy distribution, which consists of a power law with infinite variance and means of the type:

$$Levy\,(\beta)\; y = x^{-\beta} \tag{17}$$

In the case of CSA, the use of Levy flight improve and optimize the search: new solutions are generated by a random Levy walk, around the best solution obtained so far, which speeds up the overall search [37].

When generating new solutions *x (t + 1)* for a Cuckoo (solution) *i*, a Lévy flight (Eq. 17) or a random walk (Eq. 18) is performed [38, 39]:

$$X_i(t+1) = x_i(t) + \alpha \oplus Levy\,(\beta) \tag{18}$$

The product $\oplus$ represents the matrix product.

α: is the maximum length of the step, which should be related to the scale of the search space of the problem.

3.2 Cuckoo Search (CS) Strategy

The Cuckoo search algorithm is summarized around the following ideal rules [37]:

- Each Cuckoo egg in a nest represents a solution.
- Each Cuckoo bird will lay only one egg at a time and will choose its nest randomly. Therefore, each individual of the Cuckoo population has the right to randomly generate a single new solution.
- The best quality egg nests will lead us to the new generations. Here, we have implicitly introduced the notion of intensification or research around the best solutions.
- Some new solutions must be generated by the Levy flights around the best solution obtained so far. This will speed up local searches.
- The number of host nest*s* is fixed, and the *egg* laid by the bird is discovered by the host with a probability $Pa \in [0, 1]$. In this case, the host bird chooses to get rid of the egg, or abandon the nest and rebuild another nest somewhere. For simplicity, this last assumption will be approximated by the fraction *pa* of the nests *n*, which are replaced by new ones (new random solutions).
- A significant fraction of the new solutions must be generated by randomization to distant regions and whose locations must be far enough from the best current solution, which will ensure that the system will not be trapped in a local optimum.
- Each nest can contain several eggs meaning a set of solutions.

The steps of the CSA are as follows [40, 41]:

1. Select values for CSA parameters, which are the number of nests (eggs) (n), the step size parameter (β), discovering probability (pa), and the maximum number of iterations for termination of the cycles.
2. Generate initial population of n host nests $\{x_i\}$, $(i = 1, 2\ldots n)$ randomly each of which represents a candidate solution to the optimization problem with objective function of $f(x)$ and decision variables $\{x_i\} = \{x_1, x_2 \ldots x_m\}^T$.
3. Get a cuckoo randomly by Levy flights using $x_i^{v+1} = x_i^v + \beta \times \lambda$ and evaluate its fitness F_i. Here λ is a random walk based on Levy flights, which is calculated from Eqs. (18) to (19).
4. Choose randomly a nest among n (say j) and evaluate its fitness Fj.
   If $F_j < F_i$, replace j by the new solution.
5. Abandon a fraction of the worst nests and build new ones. This is carried out, depending on pa probability parameter. First, find out whether each nest keeps its current position (Eq. 20). R matrix stores 0 and 1 values such that any one of them is assigned to each component of the ith nest, in which 0 means that the current position is kept and 1 implies that the current position is to be updated:

$$R_i \leftarrow \begin{cases} 1 \;\; if \;\; rand \;\; < \;\; Pa \\ 0 \;\; if \;\; rand \;\; \geq \;\; Pa \end{cases} \tag{19}$$

   New nests are conducted by means of Eq. (20):

$$x_i^{t+1} = x_i^t \; + \; r \times \; R_i \; \times \; (perm \; 1_i \; - \; perm \, 2_i) \tag{20}$$

   where: r is a random number between 0 and 1. $Perm_1$ and $Perm_2$ are two row permutations of the corresponding nest. R defines the probability matrix.
6. Rank solutions and find the current best one.
7. Repeat steps 3–6 until termination criterion is satisfied which is usually taken as the maximum number of iterations.

Based on the rules cited above, the basic steps of the Cuckoo search algorithm can be summarized in the flowchart [42, 43].

4 Simulation Results and Discussion

The economic power dispatch problem with and without TCSC device is applied to the standard IEEE 9-bus 3-generators (Fig. 7) [44] using CSA. This approach has been executed in MATLAB 2017 under windows 8.1 on Intel Core(TM) i5-3110 CPU 2.40 GHz, with 4 GB RAM. To demonstrate the effectiveness of the CSA in solving OPF problem, two cases, are discussed (Fig. 6):

Case I: Optimal Power Flow without TCSC.

Case II: Optimal Power Flow with TCSC.

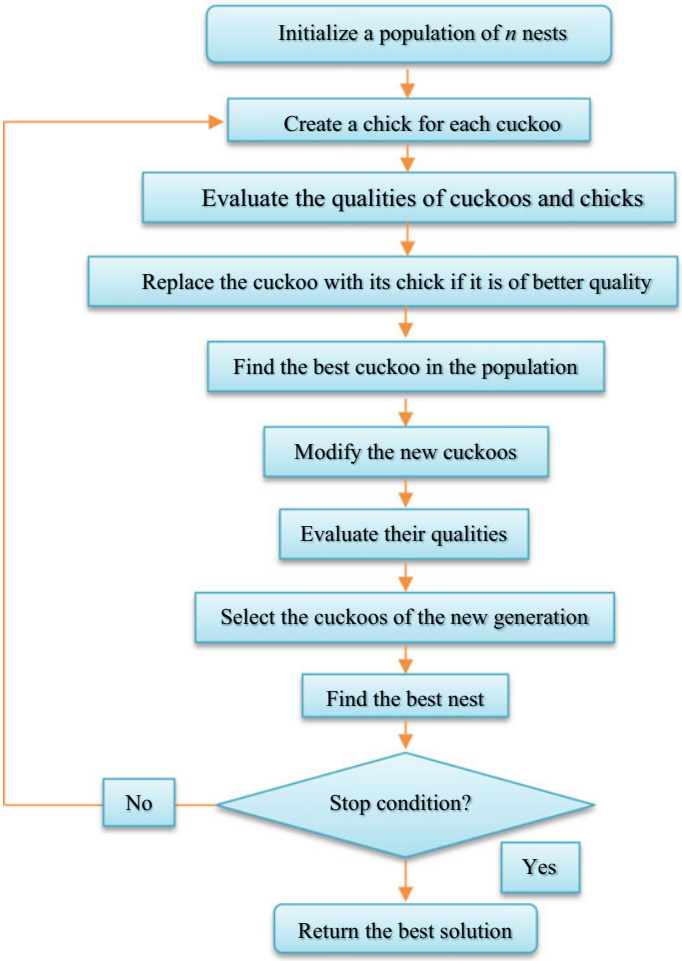


Fig. 6 Cuckoo search optimization algorithm flowchart

4.1 Case I: OPF Without the Installation of TCSC

In this case, the parameters of CS-OPF are: *nest* set to 30, the maximum number of iteration is 100, and discovery rate of alien eggs or fraction probability, *Pa* equals 0.25.

The optimal power flow program (with variable losses) is based on the Newton–Raphson method, to determine the voltages at the different buses, the powers generated, and the transmission losses. The load demand of the standard IEEE 9-bus test system is 315 MW. The obtained results using CSA are given in Tables 1 and 2. These results are compared with those obtained by GA in [45] and MATPOWER software in [46].

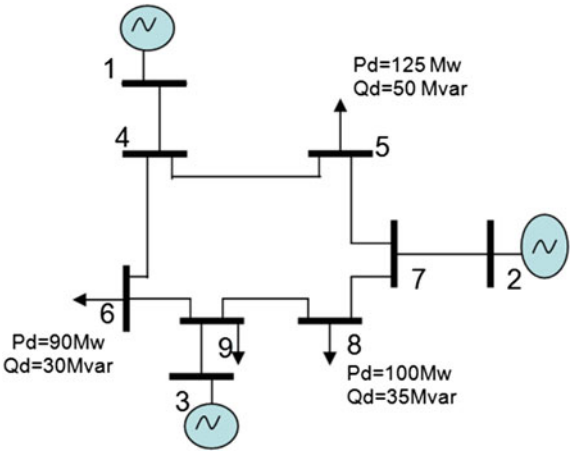


Fig. 7 IEEE 9-bus test power system

Table 1 Optimal power flow with variable transmission losses

Bus	V	Angle	Injection		Generation		load	
No	p.u	Deg	MW	MVAr	MW	MVAr	MW	MVAr
1	1.04	0.00	86.57	39.64	86.60	39.64	0.00	0.00
2	1	6.85	138.64	−0.25	138.64	−0.25	0.00	0.00
3	1	4.41	93.72	−14.78	93.72	−3.88	0.00	10.9
4	1.0192	−2.70	0.00	0.00	0.00	0.00	0.00	0.00
5	0.9841	−5.05	−125	−50	0.00	0.00	125	50
6	1.0001	−4.33	−90	−30	0.00	0.00	90	30
7	1.0039	1.89	0.00	0.00	0.00	0.00	0	0.00
8	0.9933	−0.71	−100	−35	0.00	0.00	100	35
9	1.0102	1.29	0.00	0.00	0.00	0.00	0.00	0.00
Total	–	–	4.18	−90.40	318.96	35.50	315	125.9

Table 2 The optimal values

Control variable	CS-OPF	MATPOWER [45]	GA-OPF [46]
Pg_1 (MW)	86.60	89.91	74.87
Pg_2 (MW)	138.64	134.36	161.50
Pg_3 (MW)	93.72	94.25	83.79
Total PG (MW)	318.96	318.51	320.16
Power losses (MW)	4.18	**3.515**	5.19
Cost ($/h)	**5101.59**	5301.77	5430.58

The optimal values of the powers generated, the power losses, and the fuel cost are depicted in Table 2. The comparison of our results with those obtained using other methods, proves that the proposed algorithm give a better fuel cost, is reduced by 3.77% and 6.06% than others given in Matpower [45] and GA-OPF [46], respectively. While the minimum power losses were obtained by using Matpower (3.515 MW).

Figures 8 and 9 illustrate the variations of the cost of production and the power output of all generators, respectively.

Figure 8 shows the variation of the total cost obtained by the CS- OPF. From this figure, it is clear that CSA converges to the global solution of 5194.92 ($/hr) in 65 iterations. It is obvious from Table 2 and Fig. 9; the optimum generation powers are within the security limits of each generator.

The values of the power flows of IEEE 9-bus system without installing TCSC device are depicted in Table 3.

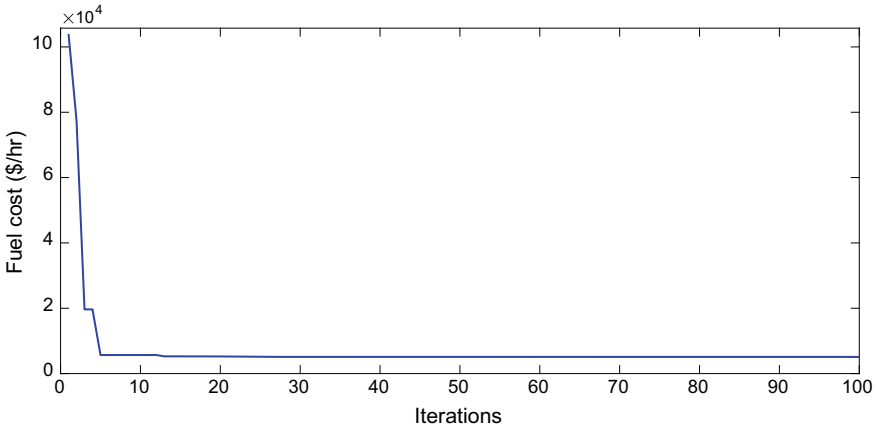


Fig. 8 Fuel cost convergence behavior of case I

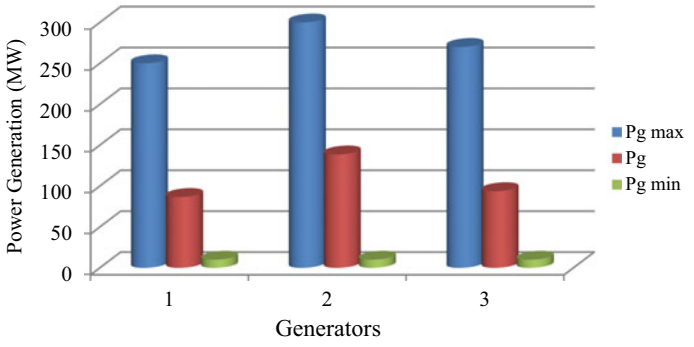


Fig. 9 Best generation schedule of IEEE 9-bus system using CSA (case I)

Table 3 Power flow through transmission lines without TCSC (case I)

From bus	To bus	P (MW)	Q (MVAr)	Line capacity		
				S_{line}	S_{max}	Overload %
4	5	90.815	69.741	114.5040	250	45.8016
4	6	17.054	14.782	22.5687	300	7.5229
5	7	−110.5	−0.436	110.4999	150	73.6666
6	9	−73.032	9.222	73.6119	250	29.4448
7	8	58.361	5.059	58.5799	250	23.4319
8	9	−41.936	−15.178	44.5982	250	17.8393

According to Table 3, to define the most critical line power flow capability, we can substantiate that the optimal placement for installing the TCSC is the line with important overload. The line 5–7 is therefore the best placement to install the TCSC.

4.2 Case II: OPF with the Presence of TCSC

In this application, we are interested in the resolve the optimal power flow with the integration of a TCSC device in the network. In order, to improve the feasibility of the Cuckoo search method, it is applied to solve OPF-CS with TCSC of the same test system used in case I. We are increasing the power demand from 315 to 390 MW. The input parameters of CSA are: *nest* set to 30, fraction probability Pa equals 0.25, and the maximum number of iteration is 500. The parameters of the TCSC are grouped in Table 4.

An optimal power flow program based on the Newton Raphson method allowed us to determine the voltages at the different buses, the powers generation, and the transmission losses.

The OPF is solved by considering the TCSC cost to determine the size of the TCSC devices and its effect on the fuel cost, the voltage profile, the transmission loss and the transfer power of the lines. In this case, the TCSC investment cost is added to the multi-objective function of the problem. The optimal location of the TCSC devices is performed by running the CS-OPF determining the priority list for locating TCSC devices. Therefore, four locations for TCSC are studied; the installation of the TCSC on lines 5–7, 6–9, 4–6, 4–5 and 7–8 and without TCSC. The control variables of the OPF problem are illustrated in Table 5.

Table 4 TCSC parameters (limits values & coefficients cost)

X_{Lmin}	X_{Lmax}	c ($)	b ($/kVar)	a ($/kVar^2)
$+0.2 \times X_L$	$-0.7 \times X_L$	127.38	−0.7130	0.0015

Table 5 Solutions of optimal control variables obtained for different placement of TCSC

Variables control	TCSC Lines 6–9	TCSC Lines 7–8	**TCSC Lines 5–7**	TCSC Lines 8–9	TCSC Lines 4–6	TCSC Lines 4–5	**Without TCSC**
Pg_1 (MW)	98.74	97.41	109.26	115.85	100.29	98.35	107.68
Pg_2 (MW)	185.89	177.74	169.06	162.18	152.59	181.55	173.34
Pg_3 (MW)	114.29	122.69	120.08	120.24	145.42	118.06	117.40
Total PG (MW)	398.92	397.85	398.41	398.27	398.29	397.95	398.43
Losses (MW)	8.86	8.49	**8.41**	8.55	8.65	8.69	8.62
V_1	1.0400	1.0400	1.0400	1.0400	1.0400	1.0400	1.0400
V_2	1.0100	1.0100	1.0000	1.0000	1.0000	1.0000	1.0000
V_3	1.0100	1.0100	1.0000	1.0000	1.0000	1.0000	1.0000
V_4	1.0055	1.0057	1.006	1.0105	1.0078	1.0081	0.9990
V_5	0.9500	0.9500	0.957	0.9634	0.9616	0.9605	0.9330
V_6	0.9898	0.9903	0.986	0.9915	0.9988	0.9987	0.9830
V_7	1.0026	1.0025	0.995	0.9998	0.9979	0.9972	0.9900
V_8	0.9952	0.9962	0.986	0.9937	0.9891	0.9887	0.9830
V_9	1.0166	1.01494	1.006	1.0061	1.0089	1.0088	1.0040
Fuel cost ($/h)	7320.44	7285.75	**7284.59**	7294.48	7421.49	7285.84	7382.33
TCSC cost ($/KVAr)	406.56	391.04	**363.13**	370.89	393.16	394.00	–

For the TCSC installed between the lines 5–7: the fuel cost, the transmission loss, the optimum values of powers generation and the locations of the TCSC are illustrated by Figs. 10, 11, 12 and 13, respectively.

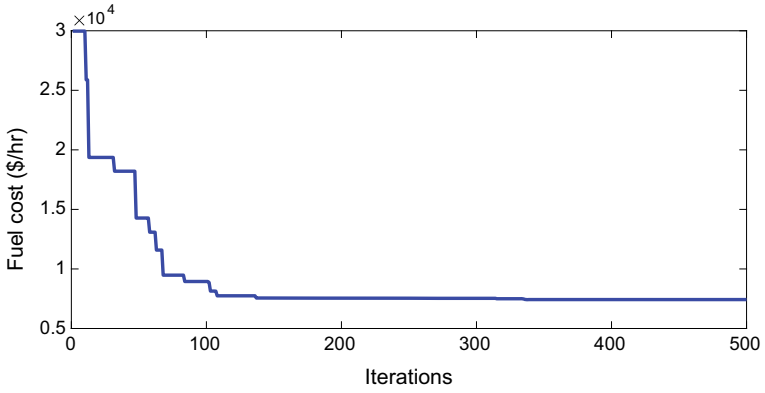


Fig. 10 Fuel cost convergence behavior using CSA (case II)

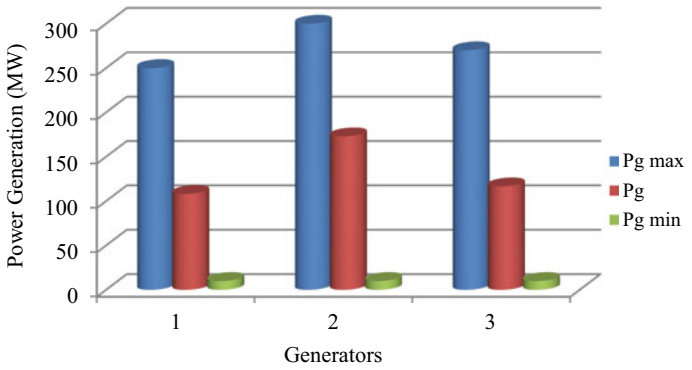


Fig. 11 Best generation schedule of IEEE 9-bus system using CSA (case II)

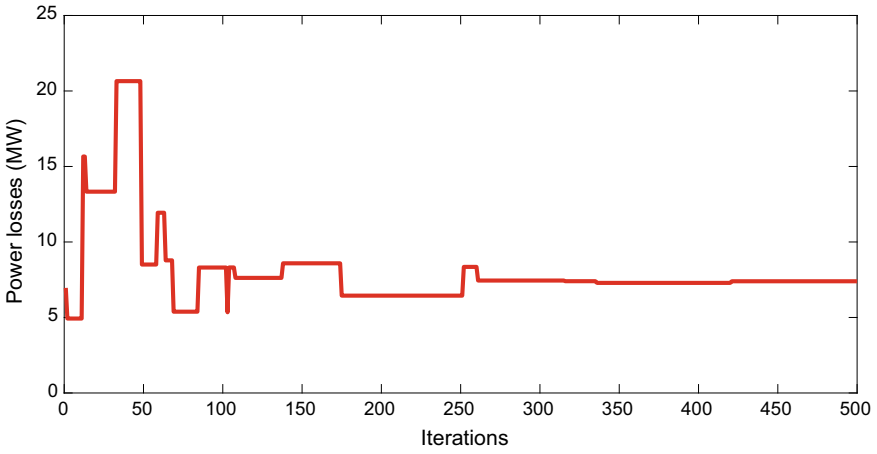


Fig. 12 Convergence property of power losses for IEEE 9-bus system using CSA

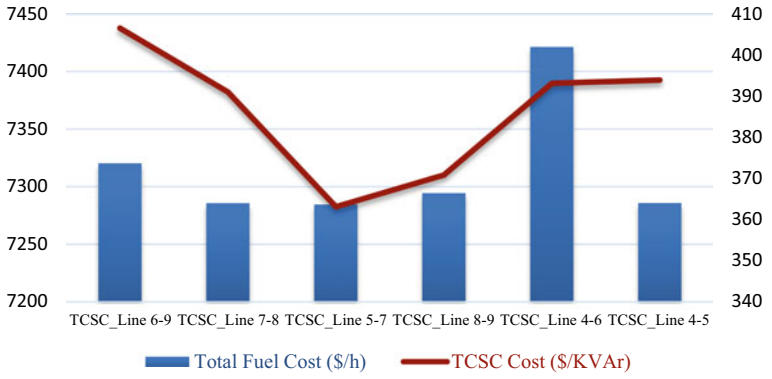


Fig. 13 Effect of location of the TCSC on the optimal cost of production

Table 6 Power flow through transmission lines with TCSC (case II)

From bus	To bus	P (MW)	Q (MVAr)	Line capacity		
				S_{line}	S_{max}	Overload %
4	5	107.675	50.654	118.9947	250	47.5979
4	6	1.589	21.677	21.7352	300	7.2451
5	7	−93.723	10.833	94.3470	150	62.8980
6	9	−88.491	16.346	89.9880	250	35.9952
7	8	72.231	5.749	72.4594	250	28.9838
8	9	−28.22	−15.667	32.2773	250	12.9109

From the results given in Table 5, we determine that the optimal location of TCSC is between the line 5–7 reduce the fuel cost, the installation cost of TCSC devices, the transmission active losses and improve the bus voltage compared with the location of the TCSC in the lines 6–9, 7–8, 8–9, 4–6, 4–5 and without the TCSC.

The total cost found by our algorithm when the TCSC is installed in the line 5–7 is equal to 7284.60 $/h. It is smaller compared to the TCSC are installed in the lines 6–9, 7–8, 8–9, 4–6 and 4–5. They are estimated at 7320.44 $/h, 7285.75 $/h, 7294.48 $/hr, 7421.49 $/hr and 7285.84 $/hr respectively. The cost of TCSC installed in line 5–7 which equals 363.13 $/ KVAr is less than in the lines 6–9, 7–8, 8–9, 4–6 and 4–5.

The investissement costs are estimated at 406.56 $/KVar and 391.04$/KVar, 370.89 $/KVar, 393.16 $/KVar and 394.00 $/KVar respectively.

The active power loss reduces to the 8.41 MW compared to the installation of TCSC in the lines 6–9, 7–8, 8–9, 4–6, and line 4–5 (8.86 MW, 8.49 MW, 8.55 MW, 8.65 MW and 8.69 MW), respectively. They are minimized by 0.45 MW, 0.08 MW, 0.14 MW, 0.24 MW and 0.28 MW respectively.

The power flow through transmission lines with installation of TCSC in line 5–7, are summarized in Table 6.

The results confirm that the series compensator TCSC is an efficient transfer power controller. We note that the overload of powers in line 5–7 are reduced and in the majority of the lines of the network has been made by comparing it to the transfer power capacity without TCSC.

Figure 13 shows the effect of optimal location of the TCSC device on the total optimal cost.

5 Conclusion

In this work, a Cuckoo search algorithm (CSA) was presented to solve the problems of optimal power flow in power systems with the investment costs of TCSC devices (Thyristor Controlled Series Capacitor). One of the main challenges of optimal power flow is congestion or overloading in the power system; TCSC devices can control the

active power transit in the networks. In general, the cost function of production units is non-convex. The cost function of the TCSC device presents the cost of installing the TCSCs modeled and imposed in the form of multi-objective problems. CSA is an effective tool for solving complex optimization problems. The effectiveness of the CS algorithm has been verified by computer experiments. Two typical OPF problems with TCSC effects were considered and the performances of different locations were compared. The numerical results from the superiority and the feasibility of the CSA compared to the other methods. The cuckoo search algorithm has been applied to determine the optimal location of TCSC devices, which can reduce power loss, improve voltage of power systems, improve power transfer and solve the overload of the line. We also conclude that with the complexity of the problems related to electrical networks by modifying their topologies by the insertion of TCSC devices, the Cuckoo search algorithm presents a better solution for optimal power flow.

Appendix 1: List of Abbreviations and Acronyms

CSA	Cuckoo search algorithm
FACTS	Flexible alternative current transmission system
GA	Genetic algorithm
OPF	Optimal power flow
TCSC	Thyristor controlled series compensator
Total PG	Total power generation

Appendix 2: List of Mathematical Symbols

Parameters:	
a_i, b_i, c_i	Cost coefficients of *i*th generator
C_1	Total generation costs
C_2	Average investment costs of FACTS devices
Ng	Number of generators in the network
N	Total number of buses in power system
P_{gi}^{min}, P_{gi}^{max}	Minimum/maximum limits of *i*th generator power, respectively
Pg_i	Generator power output
P_D	Power demand of the system
P_L	Transmission power losses

(continued)

(continued)

$Perm_1$ and $Perm_2$	Two row permutations of the corresponding nest
r	Random number varies between 0 and 1
R	The probability matrix
S_{TCSC}	Operating range of the TCSC device in MVar
X_L	Reactance of the line
X_{TCSC}	Reactance of the TCSC
$\overrightarrow{y}_{mk}$	Admittance matrix without presence of TCSC
$\overrightarrow{Y}'_{bus}$	New system admittance matrix after installing TCSC
δ	Angle voltage
Variables	
B_1, B_2	Inequality constrains for FACTS devices and power flow
$C(P_{gi})$	Total fuel cost ($/h)
$C(f)$	Total investment costs of FACTS devices
E_1	Equality constraints with respect to active and reactive power flow
$f(x, u)$	Objective function
$g(x, u), h(x, u)$	Equality and Inequality constraints, respectively
u	Control or decision variables of power system
x	State variables of power system

References

1. Murali, D., Rajaram, M.: Active and reactive power flow control using FACTS devices. Int. J. Comput. Appl. **9**, 45–50 (2010)
2. Jawad, R.S., Hussein, I.I., Mahariq, I.: FACTS technology: current challenges and future trends. In: ICEEP IV: Proceedings of the Fourth International Conference on Energy & Environmental Protection in Sustainable Development, vol. 6, pp. 45–50
3. Biswas, P.P., Suganthan, P.N., Mallipeddi, R., Amaratunga, G.A.: Optimal power flow solutions using differential evolution algorithm integrated with effective constraint handling techniques. Eng. Appl. Artif. Intell. **68**, 81–100 (2018)
4. Larouci, B., Benasla, L., Belmadani, A., Rahli, M.: Cuckoo search algorithm for solving economic power dispatch problem with consideration of FACTS devices. Sci. Bull. Ser. C-Electr. Eng. Comput. Sci. **79**(1), 43–54 (2017)
5. Shilaja, C., Ravi, K.: Optimal power flow using hybrid DA-APSO algorithm in renewable energy resources. Energy Procedia **117**, 1085–1092 (2017)
6. Khamees, A.K., Badra, N.M., Abdelaziz, A.: Optimal power flow methods: a comprehensive survey. Int. Electr. Eng. J. (IEEJ) **7**(4), 2228–2239 (2016)
7. Amarnath, R.V., Ramana, N.V.: State of art in optimal power flow solution, methodologies. J. Theor. & Appl. Inf. Technol. **30**(2) (2011)
8. Fraga, E.S., Salhi, A., Talbi, E.G.: On the impact of representation and algorithm selection for optimisation in process design: motivating a meta-heuristic framework. In: Recent Developments in Metaheuristics, pp. 141–149 (2018)

9. Gomes, W.J., Beck, A.T., Lopez, R.H., Miguel, L.F.: A probabilistic metric for comparing metaheuristic optimization algorithms. Struct. Saf. **70**, 59–70 (2018)
10. Gonçalves, M.S., Lopez, R.H., Miguel, L.F.F.: Search group algorithm: a new metaheuristic method for the optimization of truss structures. Comput. Struct. **153**, 165–184 (2015)
11. Kaveh, A., Khayatazad, M.: A new meta-heuristic method: ray optimization. Comput. Struct. **112**, 283–294 (2012)
12. Kaveh, A., Talatahari, S.: Optimum design of skeletal structures using imperialist competitive algorithm. Comput. Struct. **88**, 1220–1229 (2010)
13. Sadollah, A., Bahreininejad, A., Eskandar, H., Hamdi, M.: Mine blast algorithm for optimization of truss structures with discrete variables. Comput. Struct. **102**, 49–63 (2012)
14. Miguel, L.F.F., Miguel, L.F.F., Lopez, R.H.: A firefly algorithm for the design of force and placement of friction dampers for control of man-induced vibrations in footbridges. Optim. Eng. **16**(3), 633–661 (2015)
15. Kaveh, A., Farhoudi, N.: A new optimization method: dolphin echolocation. Adv. Eng. Softw. **59**, 53–70 (2013)
16. Hasançebi, O., Teke, T., Pekcan, O.: A bat-inspired algorithm for structural optimization. Comput. Struct. **128**, 77–90 (2013)
17. Degertekin, S.O., Hayalioglu, M.S.: Sizing truss structures using teaching-learning-based optimization. Comput. Struct. **119**, 177–188 (2013)
18. Boudjella, H., Laouer, M., Bouzeboudja, H., et al.: Solution of economic load dispatch problems using novel improved harmony search algorithm. Int. J. Electr. Eng. Inform. **13**(1), 218–241 (2021)
19. Si Tayeb, A., Larouci, B., Rezzak, D., et al.: Application of a new hybridization to solve economic dispatch problem on an Algerian power system without or with connection to a renewable energy. Diagnostyka **22**(3), 101–112 (2021)
20. Roshni, S., Pradhan, C.R., Mohapatra, B.: A Comparative study of economic load dispatch problems using classical method and artificial intelligence method. Int. J. Adv. Res. Electr. Electron. Instrum. Eng. **4**(3), 1564–1569 (2015)
21. Sedighizadeh, M., Faramarzi, H., Mahmoodi, M.M.: Hybrid approach to FACTS devices allocation using multi-objective function with NSPSO and NSGA-II algorithms in Fuzzy framework. Int. J. Electr. Power Energy Syst. **62**, 586–598 (2014)
22. Sreejith, S., Psimon, S., Selvan, M.P.: Optimal location of interline power flow controller in a power system network using ABC algorithm. Arch. Electr. Eng. **62**, 91–110 (2013)
23. Rahiminejad, A., Alimardani, A., Vahidi, B., Hosseinian, S.H.: Shuffled frog leaping algorithm optimization for AC–DC optimal power flow dispatch. Turk. J. Electr. Eng. Comput. Sci. **22**(4), 874–892 (2014)
24. Larouci, B., Benasla, L., Tahri, A., et al.: Amélioration de l'influence des variations paramétriques sur les performances de l'UPFC. Acta Electrotehnica **53**(3), 187–191 (2012)
25. Mahdad, B., Srairi, K., Bouktir, T.: Optimal power flow for large-scale power system with shunt FACTS using efficient parallel GA. In: IECON: Proceedings of the 34th Annual Conference of IEEE, pp. 867–872 (2008)
26. Capitanescu, F.: Challenges ahead risk based AC optimal power flow under uncertainty for smart sustainable power systems. In: Dynamic Vulnerability Assessment and Intelligent Control: For Sustainable Power Systems. Wiley-IEEE Press, pp. 149–176 (2018)
27. Khatoon, N., Shaik, S.: A survey on different types of flexible AC transmission systems (FACTS) controllers. Int. J. Eng. Dev. Res. **4**(5), 796–814 (2017)
28. Boudjella, H., Gherbi, F.Z., Lakdja, F., Sehnoune, H.: Performance analysis of Static Var Compensators (SVC), on congestion management and voltage profile in power systems with PSAT toolbox. Istanb. Univ.-J. Electr. Electron. Eng. **14**(1), 1697–1707 (2014)
29. Lakdja, F., Gherbi, F.Z., Berber, R., Boudjella, H.: Optimal TCSC placement for optimal power flow. J. Electr. Eng. **63**(5), 316–321 (2012)
30. Nguyen, T.T., Mohammadi, F.: Optimal placement of TCSC for congestion management and power loss reduction using multi-objective genetic algorithm. Sustainability **12**(7), 2813 (2020)

31. Sridevi, J., Amarnath, J., Rao, G.G., et al.: Influence of FACTS devices on congestion management in deregulated power system. Autom. Control Syst. Eng. **11**(2), 17–24 (2011)
32. Reddy, D.B.G., Kalavathi, M.S.: Congestion management using optimal choice and allocation of FACTS controllers. Int. J. Recent Trends Eng. Technol. **8**(2), 66–72 (2013)
33. Bhooma, S.R., Regatti, S.: Enhancement of ATC by optimal allocation of TCSC and SVC by using genetic algorithm. J. Electr. Electron. Eng. **7**(3), 24–31 (2013)
34. Cai, L.J., Erlich, I., Stamtsis, G.: Optimal choice and allocation of FACTS devices in deregulated electricity market using genetic algorithms. In: IEEE PES Power Systems Conference and Exposition, pp. 201–207 (2004)
35. Saravanan, M., Slochanal, S.M.R., Venkatesh, P.: Application of particle swarm optimization technique for optimal location of FACTS devices considering cost of installation and system loadability. Electr. Power Syst. Res. **77**(3), 276–283 (2007)
36. Larouci, B., Sitayeb, A., Boudjella, H., Ayad, A.N.E.I.: Cuckoo search algorithm to solve the problem of economic emission dispatch with the incorporation of FACTS devices under the valve-point loading effect. Facta Univ. Ser.: Electron. Energ. **34**(4), 569–588 (2021)
37. Yang, X.-S.: Nature-Inspired Metaheuristic Algorithms, Second edn., pp. 10–12. Luniver press, United Kingdom (2010)
38. Gherboudj, A.: Methods of solving difficult academic problems. Doctoral thesis. University of Biskra (2013)
39. Ouaarab, A.: Solving combinatorial optimization problems by metaheuristics inspired by nature: cuckoo search via Lévy flights. Doctoral thesis, Mohammed V University Faculty of Sciences, Rabat, Morocco (2015)
40. Feyel, P.: Optimization of correctors by metaheuristics. Application to the inertial stabilization of line of sight. Doctoral thesis (2015)
41. Gandomi, A.H., Yang, X.S., Talatahari, S., Alavi, A.H.: Metaheuristic algorithms in modeling and optimization. In: Metaheuristic Applications in Structures and Infrastructures, pp. 1–24 (2013)
42. Chandrasekaran, K., Simon, S.P.: Multi-objective unit commitment problem using cuckoo search Lagrangian method. Int. J. Eng. Sci. Technol. **4**(2), 98–105 (2012)
43. Yang, X.S., & Deb, S.: Cuckoo search via Lévy flights. In: Proceedings of the World Congress on Nature & Biologically Inspired Computing NaBIC, pp. 210–214 (2009)
44. Pan, J.S., Song, P.C., Chu, S.C., Peng, Y.J.: Improved compact cuckoo search algorithm applied to location of drone logistics hub. Mathematics **8**(3), 333–362 (2020)
45. Tahir, M.J., Latiff, I.A., Alam, M.: Transient stability analysis: evaluation of IEEE 9 bus system under line fault conditions. J. Eng. Technol. **5**, 54–58 (2017)
46. Bouktir, T., Slimani, L., Belkacemi, M.: A genetic algorithm for solving the optimal power flow problem. Leonardo J. Sci. **4**, 44–58 (2004)
47. Labed, K., Fizazi, H.M.H.: Use of the Cuckoo search algorithm for remote sensing data grouping. In: Third International Conference on Artificial Vision, CVA, Mouloud Mammeri University, Tizi-Ouzou, 12–14 April 2015 (2015)

Parameter Estimation of Per-Unit Photovoltaic Models Using Optimization Algorithms: Comparative Study

H. G. G. Nunes, J. P. A. Portugal, J. A. N. Pombo, S. J. P. S. Mariano, and M. R. A. Calado

Abstract Environmentally friendly energy production is one of this century's main challenges to achieve energy sustainability. Increasing energy production from renewable sources is essential to meet climate goals without slowing down economic growth and reducing well-being. Particularly, cost reduction in photovoltaic production technologies allows increases in installed capacity, contributing to a less expensive energy transition. Therefore, using tools to continuously monitor and control photovoltaic systems is a crucial task of increasing importance. This chapter evaluates the performance of classical and per-unit mathematical models to estimate the current–voltage characteristic curve describing the behaviour of photovoltaic cells or modules. Concretely, we compare photovoltaic parameter estimation using the classic models commonly used in the literature (single-, double- and three-diode models) with their per-unit variants. To properly assess the accuracy and reliability of the different classical and per-unit models, we developed optimization algorithms that use metaheuristic methods combined with the Newton–Raphson method. We used data from two case studies, referring to a photovoltaic cell and a photovoltaic module, both widely used in the literature. The results revealed that the accuracy, reliability and computational cost with both modelling approaches were very similar. Thus,

H. G. G. Nunes (✉) · J. P. A. Portugal · J. A. N. Pombo · S. J. P. S. Mariano · M. R. A. Calado
University of Beira Interior, Covilhã, Portugal
e-mail: hugo.nunes@ubi.pt

J. P. A. Portugal
e-mail: j.alves.portugal@ubi.pt

J. A. N. Pombo
e-mail: jose.pombo@ubi.pt

S. J. P. S. Mariano
e-mail: sm@ubi.pt

M. R. A. Calado
e-mail: rc@ubi.pt

H. G. G. Nunes · J. A. N. Pombo · S. J. P. S. Mariano · M. R. A. Calado
IT-Instituto de Telecomunicações, Covilhã, Portugal

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_6

the use of per-unit models to solve the photovoltaic parameter estimation problem through metaheuristic methods can be a valid alternative to classical models.

Keywords Photovoltaic parameter estimation · Classical models · Per-unit models · Metaheuristic methods · Newton–Raphson method

Nomenclature

a	Agent number
c_1	Cognitive acceleration coefficient
c_2	Social acceleration coefficient
C_d	Random number with a uniform distribution ∈ [0,2]
CR	Constant crossover factor ∈ [0,1]
d	Optimization problem dimension
$d_{juniorphase}$	Dimension of the junior gaining and sharing phase
$d_{seniorphase}$	Dimension of the senior gaining and sharing phase
F	Constant mutation factor ∈ [0,2]
i	Output current
i_0, i_{01}, i_{02} and i_{03}	Diode reverse saturation currents
i_{ph}	Photoelectric current
$\hat{I}$	Estimated current [A]
I	Output current [A]
I_0, I_{01}, I_{02} and I_{03}	Diode reverse saturation currents [μA]
I_H	Highest current value [A]
I_{ph}	Photoelectric current [A]
$iter_{max}$	Maximum number of iterations
k	Boltzmann constant [J/K]
k_f	Knowledge factor
k_r	Knowledge ratio
k_{rt}	Knowledge rate
L	Random number with a Lévy distribution
m_a	Mutated agent
n, n_1, n_2 and n_3	Diode ideality factors
N	Number of experimental I–V data pairs
N_s	Number of cells connected in series
p	Control parameter ∈ [0,1]
p_s	Switch probability ∈ [0,1]
P_T	Total number of agents in the population
q	Electron charge [C]
r	Resistance
r_1, r_2, r_w and ε	Random numbers with a uniform distribution ∈ [0,1]
r_D	Random index ∈ [1,d]
r_p	Parallel resistance

r_s	Series resistance
R	Resistance [Ω]
R_p	Parallel resistance [Ω]
R_s	Series resistance [Ω]
S	Step amplitude
t	Current iteration
T	Temperature [K]
u_a	Cross agent
v	Output voltage
V	Output voltage [V]
V_H	Highest voltage value [V]
V_t	Thermal voltage [V]
x	Agent position
x_{a1}, x_{a2} and x_{a3}	Mutually exclusive agents randomly selected
x_{best}	Random best agent
x_{gbest}	Global best solution/position
x_m	Random middle agent
$x_{pbest\ a}$	Best position found by the agent a so far
x_r	Randomly selected agent
x_{worst}	Random worst agent
x_α	Alpha agent
x_β	Beta agent
x_δ	Delta agent
τ	Photovoltaic model's unknown parameters
$\Gamma(\lambda)$	Gamma function with a valid distribution for $S > 0$
λ	Scaling factor
φ	Velocity of agent a
χ	Inertia weight coefficient
RMSE	Root mean square error
STD	Standard deviation

1 Introduction

A sustainable and efficient energy transition will undoubtedly rely on a resilient, decarbonised, decentralized, digitalized electrical power system and, above all, greater participation of society. To achieve this transition, it is necessary to change from a vertical electrical power system structure, supported by fossil fuels, to a circular, carbon-neutral electrical power system based on renewable energies. Furthermore, a profound transformation of today's society is needed, bringing challenges and new opportunities for companies and citizens.

A strategic factor to accelerate this sustainable transition is the promotion and dissemination of decentralized energy production from renewable and endogenous

sources. Among all renewable energy sources, photovoltaic (PV) energy has the greatest growth margin [1] and is expected to become the main renewable energy source by 2050. However, due to the intrinsic characteristics of PV energy, i.e., unpredictability and variability, suitable mathematical models are necessary to accurately characterize/simulate a PV cell or module. These mathematical models are essential to estimate/predict PV energy production and provide reliable and accurate dimensioning and technical or economic feasibility studies.

There are several mathematical models in the specialized literature to characterize the behaviour of a PV cell or module, which differ in complexity, computational cost, accuracy and popularity. The mathematical models most used in the literature are: the single-diode model (SDM) [2], double-diode model (DDM) [3] and three-diode model (TDM) [4]. These models are characterized by five, seven and nine unknown parameters, respectively. Recently, other mathematical models have emerged to more accurately characterise the physical effects that occur at the PN junction, as well as the behaviour of new PV technologies. Particularly, the multidiode model (MM) [5], formed by several diodes connected in parallel, and the multidimension diode model (MDM) [6], formed by a configurable diode network, including diodes connected in parallel and in series.

However, PV parameter estimation is a complex problem, due to the implicit and nonlinear nature of the mathematical equations that characterize these mathematical models (multimodal nonconvex problem) [7]. The available information determines the approach adopted to estimate the PV parameters associated with each mathematical model. The respective parameters can be estimated using the information available in the datasheets provided by the manufacturers or using the current–voltage (I–V) characteristic curve measured experimentally [8]. In the conventional approach, i.e., using the datasheet information, the PV parameters are estimated with analytical methods, usually with elementary equations applied at certain key points of the I–V characteristic curve or through simplifications and approximations making the equations explicit, such as the Lambert *W* function [9]. Generally, these methods are simple to implement and have a low computational cost. However, their accuracy depends significantly on the selected key points, as well as on the need to perform some simplifications or approximations. Moreover, the key points of the I–V characteristic curve, available in the datasheets, refer only to standard test conditions (STC), requiring extrapolation of values for other operating conditions, which limits the accuracy of the PV parameters.

In the second approach, PV parameter estimation is considered an optimization problem aiming to minimize the error between the experimentally measured and simulated I–V characteristic curve [10]. For this approach, there is a wide variety of methods in the specialized literature that can be categorized roughly as deterministic and metaheuristic. Deterministic methods are very efficient in convex optimization problems (local search), and, generally have a low computational cost. These include, for example, the Newton–Raphson method (NRM) [11], Levenberg–Marquardt (LM) method [12] and trust-region-reflective (TRR) method [13]. However, these methods can converge prematurely to local minimums; require conditions of continuity, convexity and differentiability; and their efficiency is extremely

dependent on initial positioning [14]. To overcome these limitations, several authors use techniques to reduce the search space dimensions. These reduction techniques, typically applied to SDM, divide the unknown PV parameters into dependent and independent. The independent PV parameters are determined by solving a nonlinear system of equations (obtained based on key points of the I–V characteristic curve) and the dependent PV parameters are determined through approximate equations or manipulated algebraically as a function of the independent PV parameters [15]. Generally, the application of techniques to reduce search space dimensions requires datasheet information and a refinement procedure to achieve high quality solutions [14]. As an alternative to dimension reduction, a new SDM modelling approach was recently introduced in the literature that uses a deterministic method based on the Lambert W function and follows the per-unit (PU) computation concept, achieving high quality solutions without reducing the search space dimensions [16].

On the other hand, metaheuristic methods have been successful in solving the PV parameter estimation problem as they simply overcome the implicit and nonlinear nature of the mathematical equations of PV models, and because they are effective in dealing with multimodal nonconvex optimization problems. These methods can be classified according to the number of evaluations performed in each iteration and whether they are based on a trajectory (a single individual or agent) or on a population of agents that cooperate with each other when exploring the search space to find high quality solutions. In fact, in recent years, there has been a clear trend towards the use of population-based metaheuristic methods, and there are numerous methods in the literature, such as: flower pollination algorithm (FPA) [17]; particle swarm optimization (PSO) with guaranteed convergence (GCPSO) [7]; double exponential function-based dynamic inertia weight PSO (DEDIWPSO) [18]; either-or teaching learning based algorithm (EOTLBO) [19]; tree growth algorithm (TGA) [20]; farmland fertility optimization (FFO) [21]; comprehensive learning Jaya algorithm (CLJAYA) [22]; multi-strategy success-history based adaptive differential evolution (DE) with linear population size reduction (MLSHADE) [23]; coyote optimization algorithm (COA) [24]; enhanced Lévy flight bat algorithm (ELBA) [25]; backtracking search algorithm with reusing differential vectors (BSARDVs) [26]; improved gaining-sharing knowledge (GSK) with automatic knowledge rate adjustment (IGSK) [27]; orthogonal moth flame optimization (MFO) with a local search (NMSOLMFO) [28]; artificial ecosystem-based optimization (AEO) [29]; grey wolf optimizer (GWO) [29]; and generalized normal distribution optimization (GNDO) [30]. Many of these metaheuristic methods were applied to the PV parameter estimation problem to improve exploration of the search space (diversification and intensification mechanisms) and, consequently, reduce computational cost. The intensification mechanism (local search) builds new solutions in regions close to the best solution found so far. In contrast, the diversification mechanism favours a global search, forcing the construction of new solutions in unexplored regions, i.e., far from the best global solution found so far. The balance and harmony between the diversification and intensification mechanisms establish the effectiveness of metaheuristic methods.

Although there is a wide variety of approaches and methods to estimate PV parameters, the development of new approaches to more efficiently find accurate and reliable solutions remains fundamental to generate techniques and tools that allow the continuous monitoring and control of PV systems. This chapter evaluates the performance of PU formulation (per-unit computation concept) in estimating the PV parameters of the single-, double-, and three-diode models through metaheuristic methods. The PU formulation was applied to SDM according to a new optimization perspective and extended to mathematical models that include two or three diodes in the equivalent circuit (DDM and TDM), models also used widely in the specialized literature. Concretely, five optimization algorithms were implemented that used five metaheuristic methods (GSK, DE, FPA, PSO and GWO) with distinct search mechanisms, i.e., different diversification and intensification mechanisms. The several optimization algorithms combined the NRM with the respective metaheuristic method to overcome the intrinsic limitations of the mathematical equations (implicit and nonlinear nature) that characterize the PV models. To properly evaluate the performance of the PU formulation in the PV parameter estimation, a comparative study was carried out that considered two standard case studies in the literature: the RTC France PV cell and the Photowatt-PWP201 PV module. The results with both modelling approaches (classical and PU models) were very competitive in terms of accuracy, reliability and computational cost, implying that PU models to estimate PV parameters through metaheuristic methods can be a valid alternative to classic models.

The chapter is organized as follows: Sect. 2 presents the per-unit mathematical models; Sect. 3 formulates the PV parameter estimation problem and introduces the metaheuristic methods; Sect. 4 presents and discusses the results of the comparative study between classical and PU models; Sect. 5 comparatively analyses the performance of the classical and PU models; Sect. 6 concludes the chapter.

2 Per-Unit Mathematical Models

Several mathematical models model the behaviour of a PV cell or module under different operating conditions, for example: the SDM, DDM, TDM, MM, and MDM. In the specialized literature, the most used mathematical models are the SDM, DDM and TDM. However, due to the implicit and nonlinear nature of the mathematical equations that characterize these models, determination of PV parameters is a complex problem (multimodal nonconvex problem). Recently, a deterministic method based on the Lambert *W* function and inspired by the PU computation concept was applied to the SDM and achieved high quality solutions without reducing the search space dimensions [16]. The "per-unit" formulation, or simply PU, expresses the physical quantities (amperes [A] for currents; volts [V] for voltages; and ohms [Ω] for resistances) in a normalized form as a function of predefined base values. To extend the application of the PU formulation to the DDM and TDM, metaheuristic optimization methods combined with NRM were used to overcome the implicit and

nonlinear nature inherent to the mathematical equations that characterize the respective PV model. This section presents and describes the models: per-unit single-diode model (PUSDM); per-unit double-diode model (PUDDM); and per-unit three-diode model (PUTDM).

2.1 Per-Unit Single-Diode Model

The SDM, shown in Fig. 1, is formed by a current source representing the photoelectric current (I_{ph}) generated by the photovoltaic effect and that depends on the operating conditions, mainly, irradiance and temperature. In parallel with the current source, there is a diode to describe the physical effects that occur at the PN junction and a resistance (R_p) that represents the leakage current resulting from parasite currents caused by imperfections in the PV cell material. Moreover, this model includes a series resistance (R_s) to account for the ohmic losses in the semiconductors as well as in the metallic contacts, i.e., losses due to the Joule effect.

Applying Kirchhoff's current law to the circuit in Fig. 1, the output current (I) is given by Eq. (1).

$$I = I_{ph} - I_0\left(e^{\frac{V+IR_s}{nV_t}} - 1\right) - \frac{V + IR_s}{R_p} \tag{1}$$

where I_0 represents the reverse saturation current of the diode, V represents the voltage, n represents the diode ideality factor, and V_t represents the thermal voltage obtained by Eq. (2).

$$V_t = \frac{N_s \times k \times T}{q} \tag{2}$$

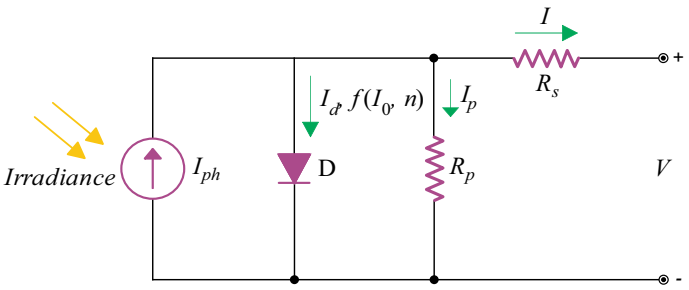


Fig. 1 Equivalent circuit of the single-diode model (SDM)

where N_s is the number of cells connected in series, k is the Boltzmann constant (1.3806503E −23 J/K), T is the temperature in Kelvin, and q is the electron charge (1.60217646E − 19 C).

Considering an I–V characteristic curve with N I–V data pairs, the following key points were selected as base values for the physical quantities of voltage, current and resistance: the highest voltage value (V_H), the highest current value (I_H) and, through Ohm's law, the resistance ($R = V_H/I_H$) [16]. Thus, applying the base values corresponding to the SDM equation, we obtain:

$$\frac{I}{I_H} = \frac{I_{ph}}{I_H} - \frac{I_0}{I_H}\left(e^{\frac{\frac{V}{V_H} + \frac{I}{I_H}\frac{R_s I_H}{V_H}}{n\frac{V_t}{V_H}}} - 1 \right) - \frac{\frac{V}{V_H} + \frac{I}{I_H}\frac{R_s I_H}{V_H}}{\frac{R_p I_H}{V_H}} \tag{3}$$

Simplifying and presenting all the terms from the previous equation in the PU formulation (lowercase characters), i.e., $I/I_H = i$, $I_{ph}/I_H = i_{ph}$, $I_0/I_H = i_0$, $V/V_H = v$, $V_t/V_H = 1/b$, $R_s/R = r_s$, $R_p/R = r_p$ with $(r = 1 + r_s/r_p)$ it is possible to represent the equation that characterizes the PUSDM as follows:

$$i = \frac{i_{ph}}{r} - \frac{i_0}{r}\left(e^{\frac{b}{n}(v + i r_s)} - 1 \right) - \frac{v}{r r_p} \tag{4}$$

2.2 Per-Unit Double-Diode Model

The DDM, shown in Fig. 2, differs from the model described above because it has two diodes in parallel with the current source. The addition of the second diode allows a more accurate representation of the physical effects that occur at the PN junction. Due to the constructive (physical) characteristics of PV cells, mathematical models with only one diode cannot reproduce the recombination process of the charge

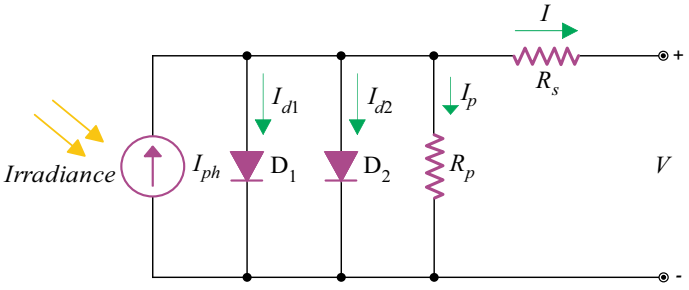


Fig. 2 Equivalent circuit of the double-diode model (DDM)

carriers at the PN junction. Thus, the DDM is commonly considered more accurate when compared to the SDM, particularly for certain PV technologies and under low irradiance levels.

Applying Kirchhoff's current law to the circuit in Fig. 2, the output current (I) is given by Eq. (5).

$$I = I_{ph} - I_{01}\left(e^{\frac{V+IR_s}{n_1 V_t}} - 1\right) - I_{02}\left(e^{\frac{V+IR_s}{n_2 V_t}} - 1\right) - \frac{V + IR_s}{R_p} \tag{5}$$

where I_{01} and n_1 represent the reverse saturation current and the ideality factor of the first diode, and I_{02} and n_2 represent the reverse saturation current and the ideality factor of the second diode.

By generalizing the conversion procedure applied to the SDM, it is possible to rewrite the DDM equation in the PU formulation as follows:

$$\frac{I}{I_H} = \frac{I_{ph}}{I_H} - \frac{I_{01}}{I_H}\left(e^{\frac{\frac{V}{V_H} + \frac{I}{I_H}\frac{R_s I_H}{V_H}}{n_1 \frac{V_t}{V_H}}} - 1\right) - \frac{I_{02}}{I_H}\left(e^{\frac{\frac{V}{V_H} + \frac{I}{I_H}\frac{R_s I_H}{V_H}}{n_2 \frac{V_t}{V_H}}} - 1\right) - \frac{\frac{V}{V_H} + \frac{I}{I_H}\frac{R_s I_H}{V_H}}{\frac{R_p I_H}{V_H}} \tag{6}$$

Taking into account all the terms of the previous equation in the PU formulation, i.e., $I/I_H = i$, $I_{ph}/I_H = i_{ph}$, $I_{01}/I_H = i_{01}$, $I_{02}/I_H = i_{02}$, $V/V_H = v$, $V_t/V_H = 1/b$, $R_s/R = r_s$, $R_p/R = r_p$ and considering ($r = 1 + r_s/r_p$), we obtain Eq. (7) that characterizes the PUDDM.

$$i = \frac{i_{ph}}{r} - \frac{i_{01}}{r}\left(e^{\frac{b}{n_1}(v+ir_s)} - 1\right) - \frac{i_{02}}{r}\left(e^{\frac{b}{n_2}(v+ir_s)} - 1\right) - \frac{v}{rr_p} \tag{7}$$

2.3 Per-Unit Three-Diode Model

The TDM is formed by three diodes in parallel with the current source, as shown in Fig. 3. In certain PV technologies, the physical effect of recombination of the carriers at the PN junction has a greater preponderance. To contemplate the physical recombination process in the defect regions and grain boundaries, the addition of a third diode in the equivalent circuit model is suggested. Although TDM, theoretically, can account for a greater number of physical effects at the PN junction, it requires more complex mathematical procedures (due to the greater number of PV parameters) and a higher computational cost.

Applying Kirchhoff's current law to the circuit in Fig. 3, the output current (I) is given by Eq. (8).

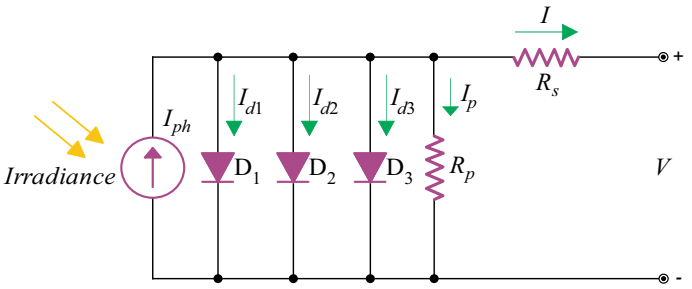


Fig. 3 Equivalent circuit of the three-diode model (TDM)

$$I = I_{ph} - I_{01}\left(e^{\frac{V+IR_s}{n_1 V_t}} - 1\right) - I_{02}\left(e^{\frac{V+IR_s}{n_2 V_t}} - 1\right) - I_{03}\left(e^{\frac{V+IR_s}{n_3 V_t}} - 1\right) - \frac{V + IR_s}{R_p} \tag{8}$$

where I_{01} and n_1 represent the reverse saturation current and the ideality factor of the first diode, I_{02} and n_2 represent the reverse saturation current and the ideality factor of the second diode, and I_{03} and n_3 represent the reverse saturation current and the ideality factor of the third diode.

Applying the base values corresponding to the TDM equation, obtained through an I–V characteristic curve with N I–V data pairs, the TDM equation in the PU formulation can be rewritten as Eq. (9).

$$\begin{aligned}\frac{I}{I_H} = \frac{I_{ph}}{I_H} &- \frac{I_{01}}{I_H}\left(e^{\frac{\frac{V}{V_H}+\frac{I}{I_H}\frac{R_s I_H}{V_H}}{n_1 \frac{V_L}{V_H}}} - 1\right) - \frac{I_{02}}{I_H}\left(e^{\frac{\frac{V}{V_H}+\frac{I}{I_H}\frac{R_s I_H}{V_H}}{n_2 \frac{V_L}{V_H}}} - 1\right) \\ &- \frac{I_{03}}{I_H}\left(e^{\frac{\frac{V}{V_H}+\frac{I}{I_H}\frac{R_s I_H}{V_H}}{n_3 \frac{V_L}{V_H}}} - 1\right) - \frac{\frac{V}{V_H} + \frac{I}{I_H}\frac{R_s I_H}{V_H}}{\frac{R_p I_H}{V_H}}\end{aligned} \tag{9}$$

Applying the same procedures and considering, respectively, the terms of the reverse saturation current of the first, second and third diodes as $I_{01}/I_H = i_{01}$, $I_{02}/I_H = i_{02}$ and $I_{03}/I_H = i_{03}$, we obtain Eq. (10) that characterizes PUTDM.

$$\begin{aligned} i = \frac{i_{ph}}{r} &- \frac{i_{01}}{r}\left(e^{\frac{b}{n_1}(v+ir_s)} - 1\right) - \frac{i_{02}}{r}\left(e^{\frac{b}{n_2}(v+ir_s)} - 1\right) \\ &- \frac{i_{03}}{r}\left(e^{\frac{b}{n_3}(v+ir_s)} - 1\right) - \frac{v}{rr_p}\end{aligned} \tag{10}$$

3 Problem Formulation and Metaheuristic Methods

PV parameter estimation is an optimization problem that consists in minimizing the error between experimental and estimated data. In this chapter, the parameters of several PV models are estimated, namely: SDM, PUSDM, DDM, PUDDM, TDM and PUTDM. Thus, the complexity and number of unknown parameters in the respective optimization problem depends directly on the number of diodes associated with each PV model. In particular, SDM and PUSDM are characterized by five unknown parameters $\tau = [I_{ph}, I_0, n, R_s, R_p]$; DDM and PUDDM are characterized by seven unknown parameters $\tau = [I_{ph}, I_{01}, I_{02}, n_1, n_2, R_s, R_p]$; and TDM and PUTDM are characterized by nine unknown parameters $\tau = [I_{ph}, I_{01}, I_{02}, I_{03}, n_1, n_2, n_3, R_s, R_p]$.

Once the PV parameter estimation is formulated as an optimization problem, the use of optimization algorithms in its resolution becomes indispensable. For this, metaheuristic methods are frequently used due to their high search capability without any problem-specific knowledge and their excellent performance on multimodal problems (multiple local minima). Their search mechanisms are usually very distinct as they are inspired by different ideas/behaviours from the real world (nature-inspired) and, therefore, can be more or less effective in finding high quality solutions. To differently solve the respective optimization problem, ensuring the adequate performance evaluation of the PU formulation, five competitive metaheuristic methods were selected, inspired by completely different ideas, namely: gaining-sharing knowledge, differential evolution, flower pollination algorithm, particle swarm optimization and grey wolf optimizer.

3.1 Objective Function

To minimize error and assess the degree of correspondence between the measured and estimated data, several performance indexes can be used, such as: the absolute error (AE); mean absolute error (MAE); sum squared error (SSE); and root mean square error (RMSE). In this chapter, the performance index considered is the RMSE, with the objective function (OF) given by Eq. (11).

$$Min\ OF(\tau) = Min\ RMSE(\tau) = Min \sqrt{\frac{1}{N}\sum_{z=1}^{N}(I_z - (V_z, \tau))^2} \tag{11}$$

where N represents the set of points (I_z, V_z) experimentally measured with $z \epsilon N$, and $\hat{I}(V_z, \tau)$ the estimated value of the current as a function of the unknown parameters τ that characterize the PV models.

Since the respective optimization problem uses implicit equations, the NRM was used to estimate the current values and, thus, overcome this limitation. For a fair comparison between the several metaheuristic methods, the unknown parameters

Table 1 Parameter ranges for the classical and per-unit models (SDM, DDM, TDM, PUSDM, PUDDM and PUTDM)

PV parameters	Case study 1				Case study 2			
	Classical models		Per-unit models [−]		Classical models		Per-unit models [−]	
	Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound
I_{ph} [A]	0	1	0	$1/I_H$	0	1.2	0	$1.2/I_H$
$I_0, I_{01}, I_{02}, I_{03}$ [A]	1E-12	1E-5	1E-12/I_H	1E-5/I_H	1E-12	1E-5	1E-12/I_H	1E-5/I_H
n, n_1, n_2, n_3	0.5	2.5	0.5	2.5	0.5	2.5	0.5	2.5
R_s [Ω]	0.001	0.5	0.001	0.5/R	0.001	2	0.001	2/R
R_p [Ω]	0.001	100	0.001	100/R	0.001	5000	0.001	5000/R

were limited according to the literature as well as according to the PV model, for each case study. Table 1 presents the bounds for the considered PV models (classical and PU models).

3.2 Gaining-Sharing Knowledge

The gaining-sharing knowledge (GSK) is a population-based metaheuristic method inspired on the acquisition and sharing of knowledge during a human being's lifetime. In a modern society, all individuals (agents), during their life course, share experiences, cooperate, acquire knowledge, and interact with each other [31]. To mimic this process, GSK has two main phases: the junior gaining and sharing phase; and the senior gaining and sharing phase. Initially, all agents in the population are considered juniors because they do not have knowledge or experience. Due to the process of acquiring and sharing knowledge, agents gradually become seniors. The knowledge branches (optimization problem dimensions) are updated at each iteration/generation through Eqs. (12) and (13).

$$d_{juniorphase}(t) = d\left(1 - \frac{t}{iter_{\max}}\right)^{k_{rt}} \tag{12}$$

$$d_{seniorphase}(t) = d - d_{juniorphase}(t) \tag{13}$$

where d represents the optimization problem dimension, $d_{juniorphase}$ represents the dimension of the junior gaining and sharing phase, t is the current iteration/generation, $iter_{max}$ is the maximum number of iterations/generations, k_{rt} is the knowledge rate (which is a real number > 0), and $d_{seniorphase}$ represents the dimension of the senior gaining and sharing phase.

In the junior gaining-sharing phase, agents gain experience and knowledge and try to share the knowledge acquired with other agents who may or may not belong to their group. This knowledge acquisition and sharing process follows the following procedure: (1) agents are sorted in ascending order based on the value of the objective function: x_{best}, …, x_{a-1}, x_a, x_{a+1}, …, x_{worst}; (2) For each agent (x_a), the two closest agents are selected as sources of acquired knowledge: the best (x_{a-1}) and the worst (x_{a+1}). Furthermore, a third agent is randomly selected (x_r) to be the source of shared knowledge. The procedure for updating agents in the junior gaining-sharing phase is obtained by

$$x_{a,d}(t+1) = \begin{cases} x_{a,d}(t) + k_f\left(\begin{matrix}\left(x_{a-1,d}(t) - x_{a+1,d}(t)\right)\\ + \left(x_{r,d}(t) - x_{a,d}(t)\right)\end{matrix}\right), & if\ f(x_a) > f(x_r)\\ & and\ r_1 \le k_r \\ x_{a,d}(t) + k_f\left(\begin{matrix}\left(x_{a-1,d}(t) - x_{a+1,d}(t)\right)\\ + \left(x_{a,d}(t) - x_{r,d}(t)\right)\end{matrix}\right), & if\ f(x_a) \le f(x_r)\\ & and\ r_1 \le k_r \\ x_{a,d}(t),\ if\ r_1 > k_r \end{cases} \quad (14)$$

where k_f is the knowledge factor (which is a real number > 0), k_r is the knowledge ratio ∈ [0, 1], and r_1 is a random number ∈ [0,1].

In the senior gaining-sharing phase, agents are influenced by social factors in the learning and knowledge acquisition process. This phase establishes the impact of the agent's interaction with other agents that have a different social condition ("social status"). This socialization process follows the following procedure: (1) agents, based on the value of the objective function, are classified into three groups, i.e., *best*, *middle* and *worst*; (2) For agent x_a to acquire knowledge, an agent from the *best* group (corresponding to $100p\%$ of the population) and another agent from the *worst* group (also related to $100p\%$ of the population) is randomly selected. Furthermore, for knowledge sharing, a third agent from the *middle* group is randomly selected (in this case $P_T - (2 \times 100p\%)$ of the population). For the senior gaining-sharing phase, the agent update procedure is obtained by

$$x_{a,d}(t+1) = \begin{cases} x_{a,d}(t) + k_f\left(\begin{matrix}\left(x_{best,d}(t) - x_{worst,d}(t)\right)\\ + \left(x_{m,d}(t) - x_{a,d}(t)\right)\end{matrix}\right), & if\ f(x_a) > f(x_m)\\ & and\ r_1 \le k_r \\ x_{a,d}(t) + k_f\left(\begin{matrix}\left(x_{best,d}(t) - x_{worst,d}(t)\right)\\ + \left(x_{a,d}(t) - x_{m,a}(t)\right)\end{matrix}\right), & if\ f(x_a) \le f(x_m)\\ & and\ r_1 \le k_r \\ x_{a,d}(t),\ if\ r_1 > k_r \end{cases} \quad (15)$$

where P_T is the total number of agents in the population, p is a control parameter ∈ [0,1], and x_{best}, x_m and x_{worst} are agents randomly selected from the *best*, *middle* and *worst* group, respectively.

3.3 Differential Evolution

Differential evolution (DE) is a metaheuristic method inspired by the selection and natural evolution of species [32]. The evolution of species takes place through successive modifications, where individuals (agents) with greater fitness are more likely to reproduce and transmit their genetic characteristics to their descendants. For that, DE uses selection procedures, based on the agent's fitness, and modifications that occur by mutation and crossover processes. The mutation process uses the vector difference between randomly selected agents. In the DE/rand/1/bin variant, a randomly selected agent (a_1) suffers a mutation that results from a vectorial difference between agents (a_2 and a_3), weighted by the mutation factor (F). Thus, the position of the new agent created by the mutation process (m_a) is given by Eq. (16).

$$m_{a,d}(t+1) = x_{a_1,d}(t) + F(x_{a_2,d}(t) - x_{a_3,d}(t)) \tag{16}$$

where x_{a1}, x_{a2} and x_{a3} are randomly selected mutually exclusive agents, F represents a constant mutation factor $\in$ [0,2], and t represents the current iteration.

The crossover process, which typically can be binomial (bin) or exponential (exp), increases diversification in the construction of new solutions. The binomial crossover process guarantees that the position of the new agent (u_a), created by the crossover process, receives at least one component from the agent generated by the mutation process. The binomial crossover process is the most used in the literature and is given by

$$u_{a,d}(t+1) = \begin{cases} m_{a,d}(t+1) & if\ r_1 \leq CR\ or\ j = r_D \\ x_{a,d}(t) & if\ r_1 > CR\ or\ j \neq r_D \end{cases} \tag{17}$$

where CR is a constant crossover factor $\in$ [0,1], r_1 is a random number with a uniform distribution between 0 and 1, and r_D is a random index $\in$ [1,d].

To decide whether a new agent u_a will be a member of the next generation, an elitist process called selection process is used, which evaluates agent u_a through the objective function. Thus, only agents with greater fitness move on to the following generations/iterations, where the selection process is given by

$$\begin{cases} if\ f(u_a(t+1)) < f(x_a(t)) \Rightarrow x_a(t+1) = u_a(t+1) \\ if\ f(u_a(t+1)) \geq f(x_a(t)) \Rightarrow x_a(t+1) = x_a(t) \end{cases} \tag{18}$$

3.4 Flower Pollination Algorithm

The flower pollination algorithm (FPA) is a metaheuristic method that simulates the pollination behaviour of vegetal species such as plants or flowers [33]. Pollination is a natural process that can be carried out by self-pollination or by cross-pollination. Cross-pollination occurs from the transfer of pollen between different vegetal species. On the other hand, self-pollination occurs in the vegetal species itself and typically occurs when there is no available pollinating agent. Pollinating agents are biotic or abiotic factors that guarantee the transport of pollen, ensuring the propagation of vegetal species. In biotic pollination, pollen is transferred by insects or other animals, while in abiotic pollination, pollen transfer occurs through physical factors, such as wind or water diffusion. All these pollination characteristics were considered in algorithm development, establishing the diversification and intensification mechanisms. The balance and harmony between the diversification and intensification mechanisms are controlled through a switch probability $p_s \in [0,1]$.

Equation (19) favours the diversification mechanism with the introduction of a random number that follow a Lévy distribution. The Lévy distribution is a stable distribution with infinite variance (heavy-tailed) that forces the construction of new solutions in regions far from the global best solution, favouring greater exploration of the search space and, thus, avoiding premature convergence.

$$x_{a,d}(t+1) = x_{a,d}(t) + L(S,\lambda)(x_{a,d}(t) - x_{gbest,d}(t)) \tag{19}$$

where $x_a(t)$ represents the position of agent a at iteration t, $x_{gbest}(t)$ represents the global best solution at iteration t, and $L(S,\lambda)$ represents a random number with a Lévy distribution calculated through Eq. (20).

$$L(S,\lambda)\ \frac{\lambda\Gamma\ (\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\pi}\frac{1}{S^{1+\lambda}} \quad (S >> S_0 > 0) \tag{20}$$

where $\Gamma(\lambda)$ is the gamma function with a valid distribution for $S > 0$, and λ represents a scaling factor that controls the step amplitude S.

To favour the intensification mechanism and ensure the construction of new solutions in regions close to the best global solution, Eq. (21) is given.

$$x_{a,d}(t+1) = x_{a,d}(t) + \varepsilon(x_{a1,d}(t) - x_{a2,d}(t)) \tag{21}$$

where ε is a random number with a uniform distribution between 0 and 1, and $x_{a1}(t)$ and $x_{a2}(t)$ are positions of two mutually exclusive agents randomly selected at iteration t.

3.5 Particle Swarm Optimization

The particle swarm optimization (PSO) is a population-based metaheuristic method inspired by principles of cooperation and social behaviour of biological swarms such as flocks of birds or swarms of fish [34]. Each agent in the population, which represents a possible solution, has a position and velocity adjusted by the agent's experience or that of other agents in the population, to search for the global best solution. Agents are randomly initialized within the multidimensional search space and, during the optimization process, evaluated through an objective function. The velocity adjustment is made through Eq. (22), which considers three factors: the cognitive factor that represents the agent's tendency to follow his own personal experience (x_{pbest}); the social factor that represents the agent's tendency to follow collective experience (x_{gbest}); and the tendency of the agent's own movement weighted by the inertia factor (χ). These three factors establish the balance between the intensification and diversification mechanisms, forcing agents to move to regions closer or further away from the global best solution, respectively. Thus, for a multidimensional search space, the movement of each agent is expressed by Eqs. (22) and (23).

$$\varphi_{a,d}(t+1) = \chi\varphi_{a,d}(t) + c_1 r_1 (x_{pbest_{a,d}}(t) - x_{a,d}(t)) + c_2 r_2 (x_{gbest,d}(t) - x_{a,d}(t)) \tag{22}$$

$$x_{a,d}(t+1) = x_{a,d}(t) + v_{a,d}(t+1) \tag{23}$$

where $x_a(t)$ represents the position of agent a at iteration t, $\varphi_a(t)$ represents the velocity of agent a at iteration t, d represents the optimization problem dimension, χ represents the inertia weight coefficient, c_1 and c_2 are acceleration coefficients (cognitive and social), r_1 and r_2 are random numbers with a uniform distribution $\in$ [0,1], $x_{pbest_a}(t)$ is the best position found by the agent a so far, and $x_{gbest}(t)$ is the global best position so far (considering all agents in the population).

3.6 Grey Wolf Optimizer

The grey wolf optimizer (GWO) is a metaheuristic method inspired by the social hierarchy and hunting behaviour of grey wolf packs (Canis lupus) [35]. By nature, wolves (agents) live in small populations with a well-defined social hierarchy. Typically, each population is organized into four distinct hierarchical levels: alpha (α), beta (β), delta (δ) and omega (ω). Considering the hierarchical levels, the position of the global solution is referred to as alpha (x_α), the second best as beta (x_β) and the third best as delta (x_δ). The remaining agents are considered omega, ω, and are the agents lower in the social hierarchy, which during the hunting social movement follow the first three. GWO's social hunting behaviour is divided into four stages: encircling prey, hunting, attacking prey, and searching for prey. The dynamic and

harmony of these four stages establishes the balance between the diversification and intensification mechanisms that are coordinated by two control parameters, described by Eqs. (24) and (25).

$$G_w = 2 - 2\frac{t}{iter_{\max}} \tag{24}$$

$$A_d = 2G_w \times r_w - G_w \tag{25}$$

where r_w represents a random number with a uniform distribution $\in [0,1]$, d represents the optimization problem dimension, $iter_{max}$ represents the maximum number of iterations, and t represents the current iteration.

The searching for prey stage, which occurs when $|A_d| > 1$, forces agents to diverge from the global best solution found so far, x_α, favouring the search for new solutions in unexplored regions (diversification mechanism). The remaining stages favour the intensification mechanism, forcing the construction of new solutions in regions close to the global best solution. Therefore, for a multidimensional search space, the new position of each agent is obtained by Eqs. (26), (27) and (28).

$$\begin{aligned} D_{\alpha,d} &= \left| C_{1,d}\, x_{\alpha,d}(t) - x_{a,d}(t) \right|, \quad D_{\beta,d} = \left| C_{2,d}\, x_{\beta,d}(t) - x_{a,d}(t) \right|, \\ D_{\delta,d} &= \left| C_{3,d}\, x_{\delta,d}(t) - x_{a,d}(t) \right| \end{aligned} \tag{26}$$

$$\begin{aligned} s_{1,d} &= x_{\alpha,d}(t) - A_{1,d}\, D_{\alpha,d}, \quad s_{2,d} = x_{\beta,d}(t) - A_{2,d}\, D_{\beta,d}, \\ s_{3,d} &= x_{\delta,d}(t) - A_{3,d}\, D_{\delta,d} \end{aligned} \tag{27}$$

$$x_{a,d}(t+1) = \frac{s_{1,d} + s_{2,d} + s_{3,d}}{3} \tag{28}$$

where, $x_a(t)$ represents the position of agent a at iteration t, $x_\alpha(t)$ represents the position of agent α at iteration t, $x_\beta(t)$ represents the position of agent β at iteration t, $x_\delta(t)$ represents the position of agent δ at iteration t, and C_d is a random number with a uniform distribution between 2 and 0.

4 Photovoltaic Parameter Estimation: Results and Discussion

This section aims to evaluate the accuracy and reliability of the PU formulation in estimating the PV parameters. To evaluate accuracy, the classical PV models (SDM, DDM and TDM) and the PV models with PU formulation (PUSDM, PUDDM and PUTDM) were considered in two case studies. Both case studies were initially proposed by [11] and have been profusely referenced by researchers in the literature [19, 21, 22, 26, 27, 36–41]. Case study 1 refers to the RTC France commercial silicon

PV cell with 57 mm diameter, operating under an irradiance of 1000 W/m^2 and a temperature of 33 °C. Case study 2 refers to the Photowatt-PWP201 PV module with 36 polycrystalline silicon cells connected in series, operating under an irradiance of 1000 W/m^2 and a temperature of 45 °C. To assess reliability, five metaheuristic methods with distinct diversification and intensification mechanisms were considered. For each, a population was considered depending on the size of the optimization problem (15 agents per dimension) and a maximum number of 5000 iterations was established, corresponding to 375,000, 525,000 and 675,000 evaluations for the PV models with one, two and three diodes, respectively. The number of maximum evaluations was the second stop condition, while the first stop condition was a 60% convergence of agents within the region defined by Eq. (29), where x_{gbest} represents the global best position and x_a the current solution of each agent of the population P_T.

$$x_{gbest,d} - x_{gbest,d} \times 0.000001 \leq x_{a,d}(t) \leq x_{gbest,d} + x_{gbest,d} \times 0.000001 \quad (29)$$

For all metaheuristic methods, defining its control parameters is crucial, because when improperly defined, exploration capacity during the search process may be compromised. Thus, the control parameters for the five metaheuristic methods were defined according to suggestions in the literature and are presented in Table 2.

All computational work was performed in Matlab® using a computer with an Intel® Xeon® E5-2630 v3 @2.40 GHz CPU processor, 64 GB RAM, and Windows 10 Professional 64-bit operating system. For each case study, 100 independent runs were performed with the different PV models.

Table 2 Parameter settings of the several metaheuristic methods

Metaheuristic methods	Control parameters
GSK	Knowledge rate: $k_{rt} = 10$
	Knowledge factor: $k_f = 0.5$
	Knowledge ratio: $k_r = 0.9$
	p parameter = 0.1
DE	Constant mutation factor: $F = 0.6$
	Constant crossover factor: $CR = 0.9$
FPA	Switch probability: $p_s = 0.8$
PSO	Cognitive acceleration coefficient: $c_1 = 1$
	Social acceleration coefficient: $c_2 = 2$
	Inertia weight coefficient χ: 0.9–0.4
GWO	G_w decrease linearly from 2 to 0

4.1 Case Study 1: Standard Photovoltaic Cell

The first case study, widely used in the literature, considers an experimental curve with 26 pairs of I–V data and refers to the RTC France PV cell (1000 W/m^2 at 33 °C). The unknown parameters τ, referring to each PV model, were estimated with the five metaheuristic methods, allowing their comparison in terms of RMSE and computational costs and, consequently, the evaluation the PU formulation's accuracy and reliability. The optimal PV parameters, corresponding to the best run, are presented only for the metaheuristic method that achieved the best performance with each of the PV models.

Results with SDM and PUSDM

Firstly, the PV parameters were estimated with both models that admit only one diode in the equivalent circuit (SDM and PUSDM). Table 3 presents the RMSE values (minimum, mean, maximum and standard deviation) obtained by each of the several metaheuristic methods, as well as the number of evaluations required (minimum, mean and maximum). As seen in Table 3, the minimum RMSE obtained with both models was 7.730063E-04, which agrees with the value in the literature [7] obtained by the GSK and DE metaheuristic methods. The GSK performed well in solving the PV parameter estimation problem, obtaining a standard deviation (STD) of 5.685287E-12 for the SDM and 6.602967E-12 for the PUSDM. The DE's accuracy was the closest to the GSK, as shown by the RMSE values in Table 3, obtaining an STD of 5.962252E-11 and 5.956031E-11 for both models. The PSO and FPA had a slightly lower accuracy, as indicated by the mean RMSE and STD values. The GWO had the worst accuracy, obtaining a minimum RMSE of 9.635614E-04 for the SDM and 8.440802E-04 for the PUSDM. The GSK and DE were not only the most accurate in estimating the PV parameters, but also presented the lowest computational cost, requiring between 60,150 and 74,475 evaluations for the GSK, and 47,700 and 67,875 evaluations for the DE. The computational cost for the PSO was between 215,100 and 375,000 evaluations. The FPA and GWO always used the second stop condition (maximum number of evaluations), since their search mechanisms maintained a great diversification in the search for solutions, making it difficult to reach the first stop condition as defined by the Eq. (29).

Regarding the PU formulation, the results were very similar between the two models (SDM and PUSDM) with the several metaheuristic methods, except for the GWO. In fact, the mean RMSE obtained by both models was mostly better with the SDM. A significant difference in accuracy between the two models only occurred with the GWO, indicating that the PU formulation can be beneficial with less effective metaheuristic methods in the search for quality solutions.

The optimal PV parameters corresponding to the best GSK run with both models were for the SDM: $I_{ph} = 0.76078796$ [A], $I_0 = 3.10685372$E-07 [μA], $n = 1.47726805$, $R_s = 0.03654690$ [Ω] and $R_p = 52.88937883$ [Ω]; and for the PUSDM: $I_{ph} = 0.99579572$, $I_0 = 4.06656186$E-07, $n = 1.47726803$, $R_s = 0.04732516$ and $R_p = 68.48746455$.

Table 3 RMSE results and number of evaluations for the RTC France PV cell with SDM and PUSDM

Method	Model	RMSE				Evaluations		
		Min	Mean	Max	STD	Min	Mean	Max
GSK	SDM	**7.730063E-04**	**7.730063E-04**	**7.730063E-04**	**5.685287E-12**	60,375	**64,936**	**71,250**
	PUSDM	**7.730063E-04**	**7.730063E-04**	**7.730063E-04**	6.602967E-12	**60,150**	65,386	74,475
DE	SDM	**7.730063E-04**	**7.730064E-04**	**7.730065E-04**	5.962252E-11	**47,700**	57,004	67,875
	PUSDM	**7.730063E-04**	**7.730064E-04**	**7.730065E-04**	**5.956031E-11**	49,200	**56,884**	**67,425**
FPA	SDM	**7.730071E-04**	**7.738876E-04**	7.859175E-04	**1.864894E-06**	**375,000**	**375,000**	**375,000**
	PUSDM	7.730083E-04	7.739462E-04	**7.856436E-04**	1.970881E-06	**375,000**	**375,000**	**375,000**
PSO	SDM	**7.730078E-04**	**7.730281E-04**	7.731350E-04	2.356731E-08	285,675	359,256	**375,000**
	PUSDM	7.730091E-04	7.730290E-04	**7.730969E-04**	**1.893757E-08**	**215,100**	**352,648**	**375,000**
GWO	SDM	9.635614E-04	3.895203E-03	6.693860E-03	1.393822E-03	**375,000**	**375,000**	**375,000**
	PUSDM	**8.440802E-04**	**3.636093E-03**	**6.422153E-03**	**1.347486E-03**	**375,000**	**375,000**	**375,000**

Results with DDM and PUDDM

Table 4 presents the RMSE and number of evaluations corresponding to the estimation of the PV parameters with DDM and PUDDM. Again, the minimum RMSE (7.182702E-04) with the two-diode models was similar to the value known in the literature [7] for the RTC France PV cell. The most accurate metaheuristic methods with DDM and PUDDM were GSK and DE, as verified with the single-diode models. Once again, the GSK stood out in terms of consistency of the solutions found, as shown by the STD values. Considering the PU model, both for GSK and DE, the STD values (1.293903E-12 and 8.027637E-11, respectively) were lower when compared to the classic model. Concretely, the classic DDM obtained on average a worse accuracy (RMSE = 7.209726E-04) with the DE, resulting in an STD of 1.184216E-05. Thus, the PU formulation favoured the results obtained with the DE. PSO was the third most accurate method, obtaining a mean RMSE of 7.562377E-04 for DDM and 7.554742E-04 for PUDDM, followed by FPA with a mean RMSE of 1.625391E-03 and 1.648370E-03, respectively. Although the GWO obtained a better minimum RMSE in relation to the FPA, on average, it presented the worst RMSE (2.761350E-03 and 2.443199E-03, respectively for both models). The increased model complexity, due to the greater number of variables, led to a higher computational cost regardless of the metaheuristic method. Again, DE had the best computational cost with an average of 159,798 and 160,866 evaluations for DDM and PUDDM, respectively, while the GSK required an average of 230,967 and 230,723 evaluations for both models, respectively. The remaining metaheuristic methods presented a computational cost close to the allowed maximum number of evaluations.

As mentioned above, on average, the PU formulation favoured the results obtained with DE, and the same was verified for PSO and GWO. Furthermore, it resulted in better STD for the more accurate metaheuristic methods and, in general, worse for the less accurate ones. As before, the PU formulation showed a slight benefit over less effective metaheuristic methods.

Again, GSK revealed the best performance. The optimal PV parameters corresponding to its best run with both models were for the DDM: $I_{ph} = 0.76082929$ [A], $I_{01} = 1.35122575\text{E-}07$ [μA], $I_{02} = 7.98105033\text{E-}06$ [μA], $n_1 = 1.40369142$, $n_2 = 2.50000000$, $R_s = 0.03795557$ [Ω] and $R_p = 60.92699304$ [Ω]; and for the PUDDM: $I_{ph} = 0.99584984$, $I_{01} = 1.76853971\text{E-}07$, $I_{02} = 1.04470036\text{E-}05$, $n_1 = 1.40368763$, $n_2 = 2.50000000$, $R_s = 0.04914932$ and $R_p = 78.89625113$.

Results with TDM and PUTDM

For the models that admit three diodes in the equivalent circuit, TDM and PUTDM, the results (RMSE and evaluations) obtained in the estimation of the unknown parameters are presented in Table 5. The results show that GSK and DE were again the most accurate when estimating the parameters associated with each model. However, GSK was surpassed by DE in terms of average accuracy and consistency, contrary to what had happened with models that only admit one and two diodes. Specifically, DE obtained an STD of 7.264205E-11 for TDM and 6.978501E-11 for PUTDM, while the GSK obtained an STD of 6.823247E-09 and 7.102653E-06 for both models,

Table 4 RMSE results and number of evaluations for the RTC France PV cell with DDM and PUDDM

Method	Model	RMSE				Evaluations		
		Min	Mean	Max	STD	Min	Mean	Max
GSK	DDM	**7.182702E-04**	**7.182702E-04**	**7.182702E-04**	1.579427E-12	186,585	230,967	**306,495**
	PUDDM	**7.182702E-04**	**7.182702E-04**	**7.182702E-04**	**1.293903E-12**	**183,750**	**230,723**	354,795
DE	DDM	**7.182702E-04**	7.209726E-04	7.730064E-04	1.184216E-05	**64,155**	**159,798**	**195,195**
	PUDDM	**7.182702E-04**	7.**182704E**-04	**7.182706E-04**	**8.027637E-11**	135,030	160,866	203,700
FPA	DDM	**9.117450E-04**	**1.625391E-03**	**2.726697E-03**	**3.387202E-04**	**525,000**	**525,000**	**525,000**
	PUDDM	9.536463E-04	1.648370E-03	2.822242E-03	3.954050E-04	**525,000**	**525,000**	**525,000**
PSO	DDM	7.191890E-04	7.562377E-04	**7.730145E-04**	**1.348151E-05**	461,895	521,322	**525,000**
	PUDDM	**7.183261E-04**	**7.554742E-04**	7.730307E-04	1.540685E-05	**419,790**	**517,448**	**525,000**
GWO	DDM	**7.492997E-04**	2.761350E-03	6.274684E-03	1.424685E-03	**525,000**	**525,000**	**525,000**
	PUDDM	7.813248E-04	**2.443199E-03**	**6.100428E-03**	**1.244268E-03**	**525,000**	**525,000**	**525,000**

Table 5 RMSE results and number of evaluations for the RTC France PV cell with TDM and PUTDM

Method	Model	RMSE				Evaluations		
		Min	Mean	Max	STD	Min	Mean	Max
GSK	TDM	**7.182702E-04**	**7.182709E-04**	**7.183383E-04**	**6.823247E-09**	**344,385**	**464,609**	**675,000**
	PUTDM	**7.182702E-04**	7.190102E-04	7.892636E-04	7.102653E-06	348,300	467,188	**675,000**
DE	TDM	**7.182702E-04**	**7.182704E-04**	**7.182706E-04**	7.264205E-11	**207,360**	**256,076**	**344,520**
	PUTDM	7.182703E-04	**7.182704E-04**	**7.182706E-04**	**6.978501E-11**	210,600	258,264	397,035
FPA	TDM	**9.999835E-04**	**2.158124E-03**	**3.244397E-03**	**4.539112E-04**	**675,000**	**675,000**	**675,000**
	PUTDM	1.127158E-03	2.263974E-03	3.545663E-03	5.409905E-04	**675,000**	**675,000**	**675,000**
PSO	TDM	**7.182716E-04**	**7.414715E-04**	**7.654873E-04**	1.442787E-05	**207,765**	**665,194**	**675,000**
	PUTDM	7.182829E-04	7.431167E-04	7.674959E-04	**1.399687E-05**	223,830	670,622	**675,000**
GWO	TDM	7.466770E-04	**2.098527E-03**	**5.848162E-03**	**1.201102E-03**	**675,000**	**675,000**	**675,000**
	PUTDM	**7.372886E-04**	2.186904E-03	8.071235E-03	1.349419E-03	**675,000**	**675,000**	**675,000**

respectively. As with the two-diode models, the PSO was the third most accurate, obtaining a mean RMSE of 7.414715E-04 for TDM and 7.431167E-04 for PUTDM. This method was followed by GWO and FPA, both with mean RMSE values in the order of 2E-03. Although the GWO was, on average, more accurate, it was less consistent than FPA, as shown by the STD values. Given the increased model complexity, the computational cost was considerably higher when compared to the single- and two-diode models. The DE required an average of 256,076 and 258,264 evaluations for TDM and PUTDM respectively. Meanwhile, the GSK required an average of 464,609 and 467,188 evaluations for each model, and the PSO 665,194 and 670,622, respectively. The FPA and GWO completed the maximum number of evaluations in all runs.

Regarding the PU formulation, the results showed a little significant benefit in terms of accuracy for the models with three diodes. In general, the results obtained with the PUTDM worsened when compared to the classic TDM in the unknown parameter estimation for the RTC France PV cell.

From the point of view of metaheuristic methods, the best performance was achieved by DE, for which the optimal PV parameters corresponding to its best run with both models are presented. Thus, the parameters assumed the following values for TDM: $I_{ph} = 0.76082917$ [A], $I_{01} = 4.38991723\text{E-}06$ [μA], $I_{02} = 3.58842512\text{E-}06$ [μA], $I_{03} = 1.35163167\text{E-}07$ [μA], $n_1 = 2.50000000$, $n_2 = 2.50000000$, $n_3 = 1.40371659$, $R_s = 0.03795517$ [Ω] and $R_p = 60.92617476$ [Ω]; and for the PUTDM: $I_{ph} = 0.99584947$, $I_{01} = 1.04348552\text{E-}05$, $I_{02} = 1.76924223\text{E-}07$, $I_{03} = 8.92099787\text{E-}09$, $n_1 = 2.50000000$, $n_2 = 1.40372179$, $n_3 = 2.49995848$, $R_s = 0.04914806$ and $R_p = 78.89954103$.

4.2 Case Study 2: Standard Photovoltaic Module

In the second case study, the estimation problem of the unknown parameters τ was based on experimental data referring to the I-V characteristic curve (with 26 pairs of data) of the Photowatt-PWP201 PV module (1000 W/m^2 at 45 °C), also an important reference in the literature. As before, the parameter estimation for the Photowatt-PWP201 PV module considered the five metaheuristic methods, allowing comparison of results obtained with the PU formulation in terms of accuracy, reliability and computational cost with those of the classic models. As in the previous case study, only the optimal PV parameters, corresponding to the best run, for the metaheuristic method that achieved the best performance are presented.

Results with SDM and PUSDM

Table 6 presents the RMSE values and number of evaluations obtained by each metaheuristic method with the single-diode models (SDM and PUSDM). The minimum RMSEs indicate that only the GWO obtained a value far from the minimum value indicated in the literature [42] (RMSE = 2.046535E-03) for the Photowatt-PWP201 PV module when considering single-diode models. In fact, given the mean RMSE,

Table 6 RMSE results and number of evaluations for the Photowatt-PWP201 PV module with SDM and PUSDM

Method	Model	RMSE				Evaluations		
		Min	Mean	Max	STD	Min	Mean	Max
GSK	SDM	**2.046535E-03**	**2.046535E-03**	**2.046535E-03**	1.112840E-11	56,850	**61,582**	**67,425**
	PUSDM	**2.046535E-03**	**2.046535E-03**	**2.046535E-03**	**9.951934E-12**	**55,800**	61,679	69,375
DE	SDM	**2.046535E-03**	**2.046535E-03**	2.046536E-03	1.981267E-10	**26,100**	**31,907**	40,650
	PUSDM	**2.046535E-03**	**2.046535E-03**	**2.046535E-03**	**1.543982E-10**	26,400	32,395	**39,825**
FPA	SDM	**2.046535E-03**	**2.046536E-03**	**2.046587E-03**	**6.072896E-09**	317,700	367,743	**375,000**
	PUSDM	**2.046535E-03**	2.046538E-03	2.046654E-03	1.504705E-08	**308,100**	**366,426**	**375,000**
PSO	SDM	**2.046538E-03**	2.046657E-03	2.047924E-03	1.701231E-07	**186,150**	**352,166**	**375,000**
	PUSDM	2.046543E-03	**2.046632E-03**	**2.047219E-03**	**1.245739E-07**	253,800	353,339	**375,000**
GWO	SDM	**2.071145E-03**	3.161454E-03	6.987550E-03	8.057118E-04	**375,000**	**375,000**	**375,000**
	PUSDM	2.092484E-03	**3.129321E-03**	**5.689905E-03**	**5.968961E-04**	**375,000**	**375,000**	**375,000**

the GWO was significantly less accurate. The remaining methods achieved similar accuracy, as indicated by the mean RMSE, obtaining good STD values with both PV models. However, a more detailed analysis shows that GSK was the most accurate and consistent, with an STD of 1.112840E-11 for SDM and 9.951934E-12 for PUSDM. This was followed by DE with an STD of 1.981267E-10 and 1.543982E-10 for both models, respectively. Regarding the FPA, the PU model obtained slightly worse results compared to the classic model, with an STD of 6.072896E-09 for SDM and 1.504705E-08 for PUSDM. Meanwhile, the PSO had a lower average accuracy, with an STD of 1.701231E-07 and 1.245739E-07 for both models, respectively. The GWO was the least accurate and consistent, with an STD of only 8.057118E-04 and 5.968961E-04 for SDM and PUSDM, respectively.

Regarding the computational cost, and as with case study 1, the number of evaluations was not significantly different when comparing the classic and PU models. However, the difference between metaheuristic methods regarding computational cost was significant. Specifically, the most efficient metaheuristic methods required an average of 32,000 (DE) and 62,000 (GSK) evaluations. Meanwhile, the PSO and FPA, on average, required around 353,000 and 367,000 evaluations, respectively. GWO required the maximum number of evaluations on all runs. Table 6 indicates that, in general, the PU formulation favoured the consistency of the different metaheuristic methods but did not present a benefit in terms of accuracy, as can be seen by the minimum RMSE.

The optimal PV parameters corresponding to the best GSK run with both models were for the SDM: $I_{ph} = 1.03238251$ [A], $I_0 = 2.51292971\text{E-}06$ [μA], $n = 1.31730516$, $R_s = 1.23928732$ [Ω] and $R_p = 744.70403998$ [Ω]; and for the PUSDM: $I_{ph} = 0.99795312$, $I_0 = 2.42912196\text{E-}06$, $n = 1.31730505$, $R_s = 0.07330784$ and $R_p = 44.05187359$.

Results with DDM and PUDDM

Table 7 presents the results with the two-diode models (DDM and PUDDM). Considering the minimum RMSE, the increase in model complexity was notable with the FPA, since it reached only a value of 2.104130E-03 and 2.132261E-03 for the DDM and PUDDM models, respectively. Meanwhile, the remaining methods obtained values similar to the minimum known value (2.046535E-03) [42]. However, on average, the GWO performed poorly again and the FPA significantly lost accuracy due to greater complexity. GSK was again the most accurate and consistent method, obtaining an STD of 1.666725E-11 and 1.691194E-11 for DDM and PUDDM, respectively. This was followed by DE with an STD of 1.791019E-10 and 1.490439E-10 for both models, respectively. The PSO was on average the third most accurate and consistent, obtaining an STD of 9.386547E-06 for DDM and 9.721714E-06 for PUDDM.

Table 7 RMSE results and number of evaluations for the Photowatt-PWP201 PV module with DDM and PUDDM

Method	Model	RMSE				Evaluations		
		Min	Mean	Max	STD	Min	Mean	Max
GSK	DDM	**2.046535E-03**	**2.046535E-03**	**2.046535E-03**	**1.666725E-11**	105,210	130,501	**196,035**
	PUDDM	**2.046535E-03**	**2.046535E-03**	**2.046535E-03**	1.691194E-11	**102,585**	**130,030**	229,320
DE	DDM	**2.046535E-03**	**2.046535E-03**	2.046536E-03	1.791019E-10	40,740	47,656	56,385
	PUDDM	**2.046535E-03**	**2.046535E-03**	**2.046535E-03**	**1.490439E-10**	**38,955**	**46,591**	**53,445**
FPA	DDM	**2.104130E-03**	**2.350088E-03**	**2.635216E-03**	**1.177677E-04**	**525,000**	**525,000**	**525,000**
	PUDDM	2.132261E-03	2.362837E-03	2.734528E-03	1.257453E-04	**525,000**	**525,000**	**525,000**
PSO	DDM	**2.046535E-03**	**2.048999E-03**	2.104693E-03	**9.386547E-06**	320,355	502,002	**525,000**
	PUDDM	2.046536E-03	2.049254E-03	**2.096536E-03**	9.721714E-06	**285,705**	**493,889**	**525,000**
GWO	DDM	2.072332E-03	**3.048873E-03**	8.648283E-03	8.772250E-04	**525,000**	**525,000**	**525,000**
	PUDDM	**2.058874E-03**	3.100867E-03	**6.961847E-03**	**8.233606E-04**	**525,000**	**525,000**	**525,000**

Since the respective PV models include two diodes in the equivalent circuit, the computational cost was higher, reflecting the increase in complexity. Thus, on average, DE and GSK required around 47,000 and 130,000 evaluations, respectively. The PSO required around 500,000 evaluations, while the FPA and PSO required the maximum number of evaluations. Regarding the PU formulation, there was no significant benefit according to the results in Table 7.

Again, the best performance was achieved by GSK. The optimal PV parameters corresponding to its best run with both models were for the DDM: $I_{ph} = 1.03238256$ [A], $I_{01} = 2.51291274$E-06 [μA], $I_{02} = 1.00000000$E-12 [μA], $n_1 = 1.31730452$, $n_2 = 2.39688301$, $R_s = 1.23928692$ [Ω] and $R_p = 744.69465435$ [Ω]; and for the PUDDM: $I_{ph} = 0.99795318$, $I_{01} = 2.42919485$E-06, $I_{02} = 1.08578437$E-12, $n_1 = 1.31730810$, $n_2 = 2.40276623$, $R_s = 0.07330751$ and $R_p = 44.05169258$.

Results with TDM and PUTDM

The results for the three-diode models (TDM and PUTDM) are shown in Table 8. Considering the mean RMSE, once again the most accurate methods were the GSK and DE, with a mean accuracy of 2.046535E-03. Solution consistency was maintained regardless of greater model complexity. Specifically, the GSK obtained an STD of 2.744962E-11 for TDM and 2.019481E-11 for PUTDM, while the DE obtained 1.531065E-10 and 1.716132E-10 for both models, respectively. Once again, the PSO held the third best position in terms of accuracy and consistency, obtaining an STD of 1.437060E-05 for TDM and 1.007919E-05 for PUTDM. This was followed by FPA, with a mean accuracy in the order of 2.5E-03, and the GWO with a mean accuracy in the order of 2.9E-03. Both presented the worst consistencies relative to the solutions found, as indicated by the STD.

Computational cost was obviously higher when compared to previous models. However, it should be noted that, regardless of the added complexity, the DE showed a high efficiency, converging with an average of 74,656 and 74,558 evaluations with TDM and PUTDM respectively. Meanwhile, the GSK required an average of 240,503 and 242,395 evaluations for each model, and the PSO 636,071 and 629,609, respectively. The FPA and GWO once again completed the maximum number of evaluations. Regarding the PU formulation, GSK displayed a slight benefit in terms of consistency, and PSO and GWO in terms of consistency and accuracy.

Finally, GSK again obtained the best performance. The optimal PV parameters corresponding to its best run with both models were for the TDM: $I_{ph} = 1.03238232$ [A], $I_{01} = 2.51299128$E-06 [μA], $I_{02} = 3.60097649$E-12 [μA], $I_{03} = 1.00000000$E-12 [μA], $n_1 = 1.31730781$, $n_2 = 1.33751644$, $n_3 = 2.17421390$, $R_s = 1.23928449$ [Ω] and $R_p = 744.72076947$ [Ω]; and for the PUTDM: $I_{ph} = 0.99795275$, $I_{01} = 9.66650556$E-13, $I_{02} = 9.66650556$E-13, $I_{03} = 2.42916962$E-06, $n_1 = 1.96324103$, $n_2 = 2.43119603$, $n_3 = 1.31730698$, $R_s = 0.07330772$ and $R_p = 44.05386488$.

Table 8 RMSE results and number of evaluations for the Photowatt-PWP201 PV module with TDM and PUTDM

Method	Model	RMSE				Evaluations		
		Min	Mean	Max	STD	Min	Mean	Max
GSK	TDM	**2.046535E-03**	**2.046535E-03**	**2.046535E-03**	2.744962E-11	168,750	**240,503**	570,105
	PUTDM	**2.046535E-03**	**2.046535E-03**	**2.046535E-03**	**2.019481E-11**	**160,650**	242,395	**433,620**
DE	TDM	**2.046535E-03**	**2.046535E-03**	**2.046535E-03**	**1.531065E-10**	**61,290**	74,656	100,170
	PUTDM	**2.046535E-03**	**2.046535E-03**	2.046536E-03	1.716132E-10	62,775	**74,558**	**94,365**
FPA	TDM	**2.140281E-03**	**2.544101E-03**	**2.855873E-03**	1.580980E-04	**675,000**	**675,000**	**675,000**
	PUTDM	2.211967E-03	2.581074E-03	2.986570E-03	**1.542842E-04**	**675,000**	**675,000**	**675,000**
PSO	TDM	**2.046535E-03**	2.051659E-03	2.147475E-03	1.437060E-05	**398,115**	636,071	**675,000**
	PUTDM	**2.046535E-03**	**2.049970E-03**	**2.097955E-03**	**1.007919E-05**	398,385	**629,609**	**675,000**
GWO	TDM	2.113749E-03	2.935864E-03	8.010470E-03	6.947736E-04	**675,000**	**675,000**	**675,000**
	PUTDM	**2.079652E-03**	**2.826653E-03**	**5.851815E-03**	**5.408912E-04**	**675,000**	**675,000**	**675,000**

5 Result Comparison

In this section, the accuracy and reliability of the PU formulation are compared with the classic models in terms of PV parameter estimation with both case studies, allowing a better interpretation of the results presented above. To ensure a fair comparison of performance, it is important to compare the accuracy of the classic and PU models using the same metaheuristic method. However, to compare reliability of the PU formulation, it is important to compare the effectiveness of the several metaheuristic methods in determining accurate solutions with each of the classical and PU models. Specifically, we compared convergence results, the RMSE distribution, and the rank of the several metaheuristic methods with different PV models for the two case studies. The convergence results correspond to the best of 100 runs, while the RMSE and rank consider the 100 runs performed.

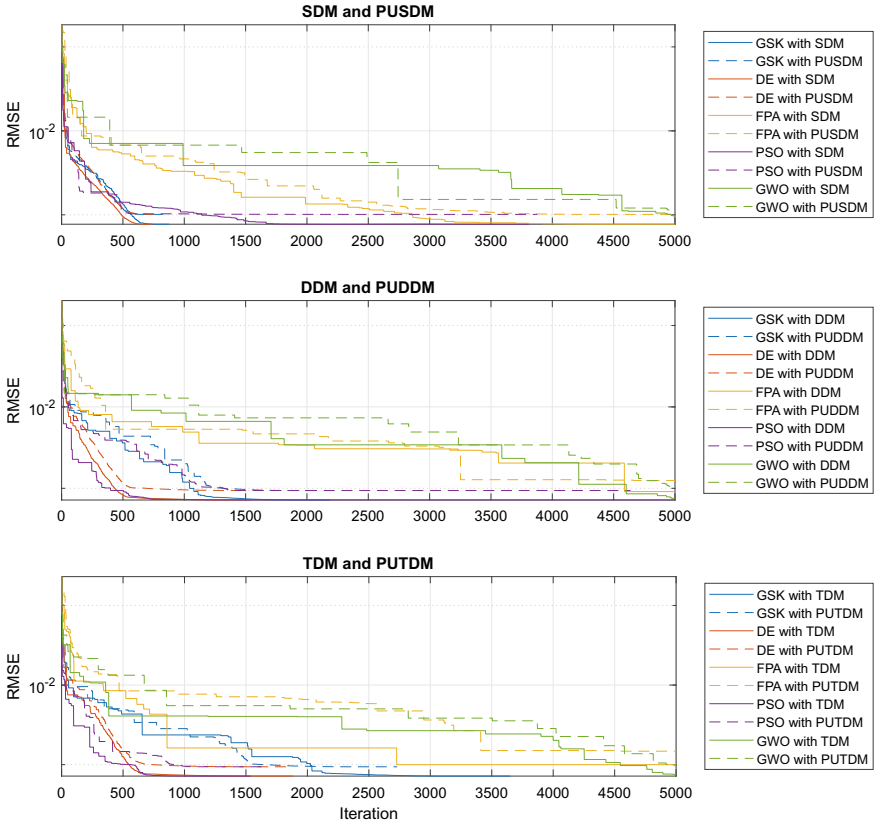


Fig. 4 Convergence curves of the five metaheuristics with different PV models for the RTC France PV cell

For case study 1 (RTC France PV cell), Fig. 4 shows the convergence curves of the five metaheuristic methods with the single-, two- and three-diode models, respectively. For the single-diode models (SDM and PUSDM), the DE and GSK converged fastest, not reaching 1000 iterations regardless of the model. The third best convergence speed was obtained by PSO, while FPA and GWO were clearly the slowest to converge. There was no significant difference in convergence between the two models, as can be seen from the proximity of the curves referring to the same metaheuristic method, although the RMSE value was higher with the PU model. For the two-diode models (DDM and PUDDM), initially the PSO and DE displayed the highest convergence rate, followed by the GSK; however the PSO required a greater number of iterations to achieve a similar RMSE. Again, FPA and GWO presented the slowest convergence rate, with the search process terminating at 5000 iterations with RMSE values far from the minimum obtained by the other metaheuristic methods. The convergence curves for the three-diode models (TDM and PUTDM) presented a behaviour similar to that verified with the DDM and PUDDM. The PSO and DE once again had the fastest convergence in the initial phase of the search process, followed by the GSK. The GWO and FPA required the largest number of iterations to minimize the objective function, and the FPA, regardless of the model, was farthest from the best solution. Overall, the PU formulation did not lead to significant differences regarding convergence speed with the RTC France PV cell but resulted in slightly higher RMSE values, when considering a large number of decimal places, even with the most efficient metaheuristic methods.

Figure 5 for case study 2 (Photowatt-PWP201 PV module) shows the convergence curves for the five metaheuristic methods with the different PV models. Considering SDM and PUSDM, DE and GSK demonstrated the highest convergence speeds, requiring a reduced number of iterations with both models. Specifically, DE required less than 500 iterations until it converged according to the first stop condition and GSK required less than 1000 iterations. The behaviour of the DE and GSK convergence curves was similar with both models (SDM and PUSDM). The third fastest to converge was the PSO, which approached the minimum RMSE value after 500 iterations, but only converged around 3000 iterations with SDM and 4350 iterations with PUSDM. These methods were followed by the FPA, which only approached the minimum RMSE value after 1500 iterations, requiring a number of iterations close to the maximum allowed with both models. The GWO clearly had the lowest convergence speed, approaching the minimum RMSE only after 4800 iterations, but never reaching this value. For DDM and PUDDM, DE again had the highest convergence rate, requiring the lowest number of iterations. The PSO had the second-best initial convergence rate, followed by the GSK, which initially showed a slower convergence, but performed considerably less iterations than the PSO until it converged. The slowest to converge were again FPA and GWO. In this case study, regardless of the metaheuristic method, PUDDM obtained a better RMSE value when compared to DDM. Considering the convergence curves with TDM and PUTDM, again DE achieved the best convergence speed despite the increased complexity of the models. As for the DDM and PUDDM, the PSO had the second-best initial convergence rate and the GSK the third, although the PSO required a greater number of iterations

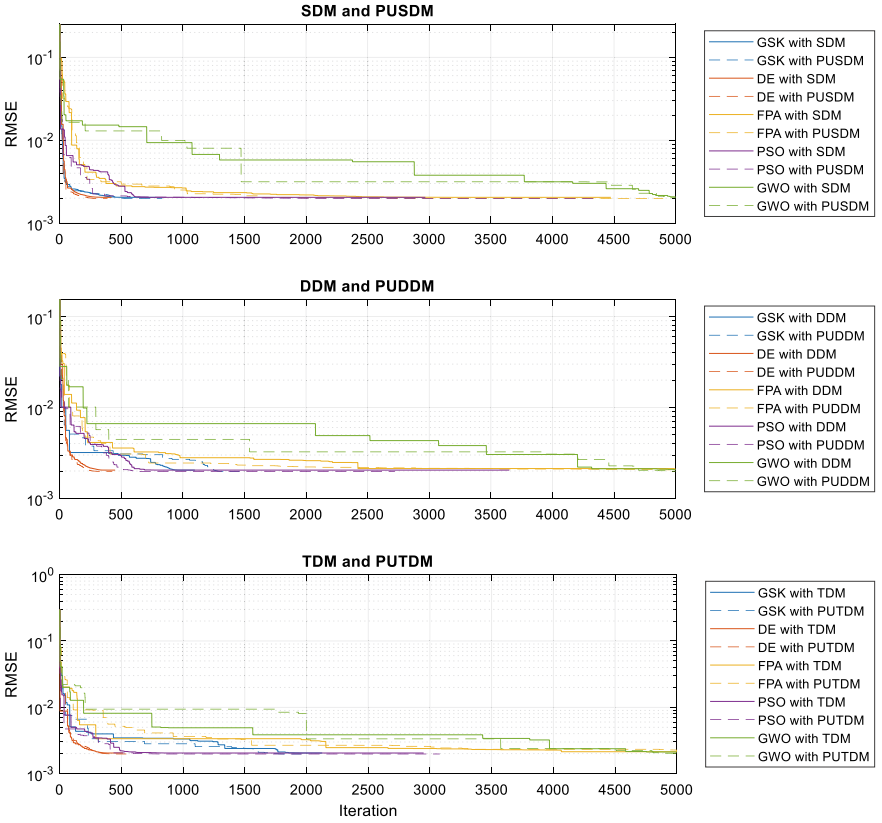


Fig. 5 Convergence curves of the five metaheuristics with different PV models for the Photowatt-PWP201 PV module

until it converged. Again, FPA and GWO showed the lowest convergence speed, approaching the minimum RMSE value only in the final phase of the search process, which ended at 5000 iterations with both models. Regarding the PU formulation, the convergence behaviour with the different models (one, two or three diodes) was similar regardless of the metaheuristic method; however this difference was more notable with the GWO.

For case study 1, the RMSE distribution referring to the five metaheuristic methods with the different PV models is shown in Fig. 6. A first analysis of the respective RMSE distributions, obtained for the RTC France PV cell, indicates that the GWO found a wide range of solutions with any of the PV models under consideration, proving to be very unreliable in estimating the PV parameters, as shown in Tables 3, 4 and 5. FPA also showed reduced reliability when estimating the PV parameters of different models, especially those that include two or three diodes in the equivalent circuit. However, a more detailed analysis of Fig. 6 clearly shows that the GSK obtained the smallest variation of solutions, according to its RMSE distribution,

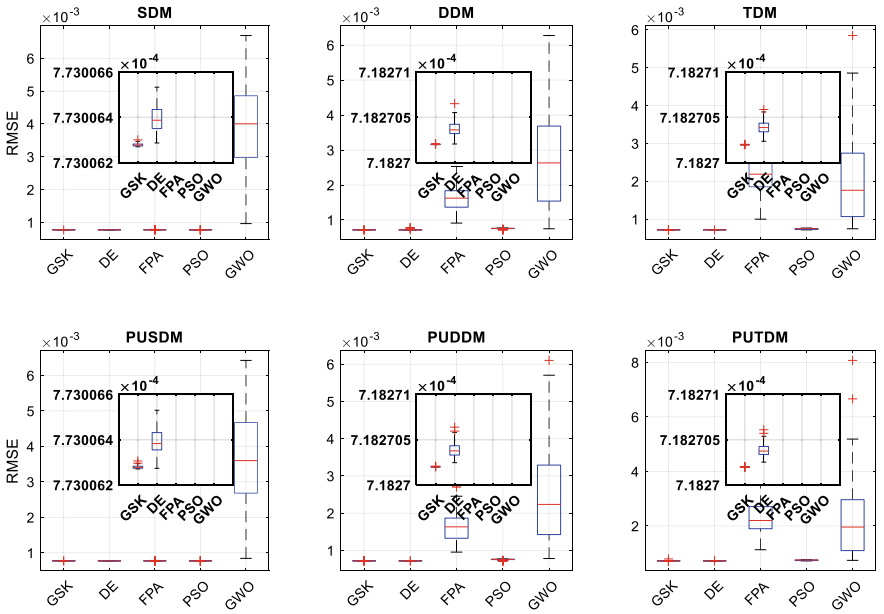


Fig. 6 RMSE distribution achieved by the five metaheuristics with different PV models for the RTC France PV cell

proving to be the most reliable in estimating the PV parameters, independently of the model considered. With a very tight RMSE distribution, in the different models, the DE and PSO were the second and third most reliable in PV parameter estimation of the RTC France PV cell, respectively.

Figure 7 shows the RMSE distribution for the Photowatt-PWP201 PV module considered in case study 2. As in case study 1, the GWO obtained the greatest variation regarding the RMSE distribution, and was therefore the least reliable with any of the PV models. Although, FPA also had low reliability, particularly with models that included two or three diodes, it was more reliable than PSO on models that included only one diode (SDM PUSDM). In general, the GSK and DE presented again the smallest variation of solutions, being therefore the most reliable in the PV parameter estimation of case study 2, although DE presented greater variation than FPA for the SDM. In both case studies, Fig. 6 and Fig. 7, the GSK displayed high reliability with both the classic and PU models, as already demonstrated in Sect. 4 by the STD values. Although DE had, in general, a slightly lower reliability, it was the most efficient from a computational point of view. It should be noted that, regarding the PU formulation, the RMSE distributions obtained with the several metaheuristic methods were very similar to those of the classical models, so performance in terms of reliability of both formulations was similar.

Figure 8 shows the rank obtained by the five metaheuristic methods with the different PV models for the RTC France PV cell. The respective ranks consider the 100 runs performed and were determined by the Friedman test (a nonparametric

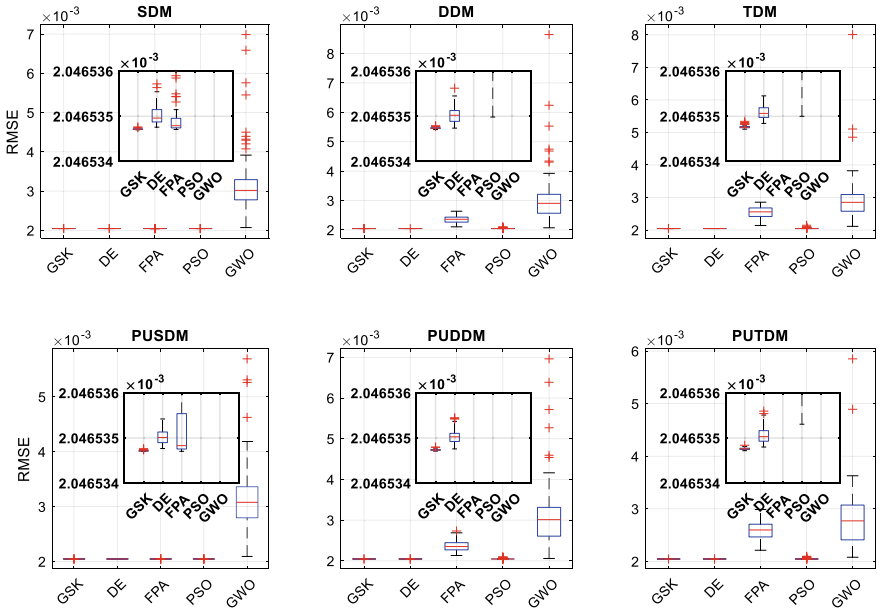


Fig. 7 RMSE distribution achieved by the five metaheuristics with different PV models for the Photowatt-PWP201 PV module

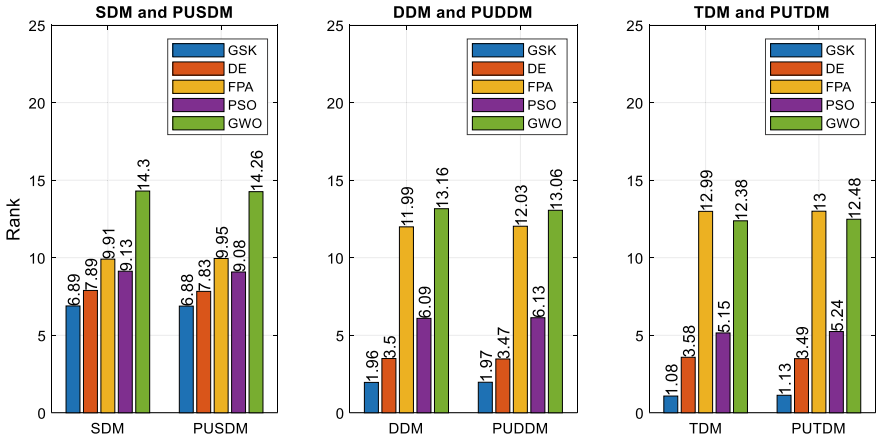


Fig. 8 Rank obtained by the five metaheuristics with different PV models for the RTC France PV cell

statistical test), considering the performance of the different metaheuristic methods for each of the PV models. As illustrated in Fig. 8, GSK achieved the best rank with all of the PV models, followed by DE, PSO, FPA and GWO. With the TDM and PUTDM models, the GWO obtained better ranks when compared to the FPA. For

the SDM and PUSDM, GSK obtained a rank of 6.89 and 6.88, respectively, while the DE obtained 7.89 and 7.83; PSO 9.13 and 9.08; FPA 9.91 and 9.95; and GWO 14.3 and 14.26. With DDM and PUDDM, GSK obtained rank values of 1.96 and 1.97; DE 3.5 and 3.47; PSO 6.09 and 6.13; FPA 11.99 and 12.03; and GWO 13.16 and 13.06. Finally, with TDM and PUTDM, the GSK obtained a rank of 1.08 and 1.13, respectively; the DE 3.58 and 3.49; PSO 5.15 and 5.24; GWO 12.38 and 12.48; and FPA 12.99 and 13. Regardless of the PV model, GSK, DE and PSO were the most competitive in PV parameter estimation for case study 1. However, regarding the PU formulation (PUSDM, PUDDM and PUTDM), the ranks obtained were very similar to those obtained with the classic models (SDM, DDM and TDM). In fact, the rank values obtained with the PU models were slightly better in only about fifty percent of the cases.

For the Photowatt-PWP201 PV module of case study 2, the rank results of the five metaheuristic methods with the different PV models are presented in Fig. 9. The several metaheuristic methods kept the same ranking trend of case study 1, except for the SDM and PUSDM models, where FPA surpassed PSO and DE, obtaining the second-best rank with both models. In this case study, GSK also obtained the best rank regardless of the PV model, obtaining a rank of 1.19 and 1.13, for SDM and PUSDM respectively. Meanwhile, the FPA obtained 4.3 and 4.85; DE 5.47 and 5.53; PSO 9.46 and 9.52; and GWO 13.94 and 14.01. For the DDM and PUDDM models, GSK obtained rank values of 2.25 and 2.32; DE 5.64 and 5.53; PSO 8.94 and 8.87; FPA 11.47 and 11.55; and GWO 13.62 and 13.7. For TDM and PUTDM, GSK obtained a rank of 3.2 and 3.1; DE 6.05 and 5.67; PSO 8.5 and 8.48; FPA 12.42 and 12.6; and finally, GWO 13.55 and 13.14. As in case study 1, from the point of view of the PU formulation, the rank results were again very similar. Here too, there was a slight improvement in the rank obtained with the PU models only about fifty percent of the time.

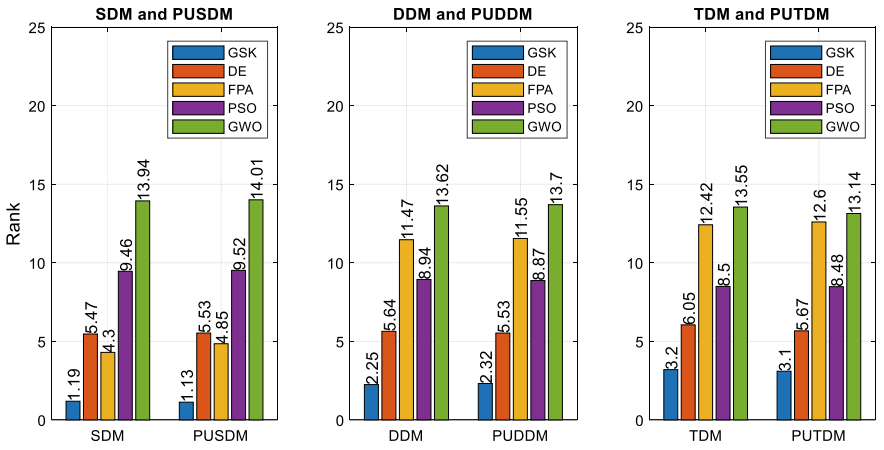


Fig. 9 Rank obtained by the five metaheuristics with different PV models for the Phtowatt-PWP201 PV module

This study demonstrated that the use of PU models in solving the PV parameter estimation problem using metaheuristic methods can be a valid alternative to classic models. The results show that the two modelling approaches had similar performance in terms of accuracy, reliability and computational cost. In fact, the results achieved with the PU formulation were sometimes slightly better and sometimes worse. Furthermore, the two modelling approaches under study (classical and PU models) depend on the same information.

6 Conclusions

This chapter evaluates the performance of classical and per-unit (PU) mathematical models in the PV parameter estimation problem, specifically comparing the accuracy and reliability of the three classical models most used in the literature (SDM, DDM and TDM) with that achieved by the respective models in their PU formulation (PUSDM, PUDDM and PUTDM). For this purpose, optimization algorithms that used different metaheuristic methods combined with the Newton–Raphson method were implemented. Thus, the respective optimization problem was solved differently, ensuring an adequate performance evaluation of the PU models when compared to the classic models. Five competitive metaheuristic methods with different inspirations were selected, namely: GSK, DE, FPA, PSO and GWO. The performance of both modelling approaches (classical and PU models) was evaluated in two case studies, a PV cell and a PV module, both standard in the literature. The results show that the performance regarding accuracy, reliability and computational cost with both modelling approaches was very similar. The GSK was the most effective metaheuristic method in the PV parameter estimation, with the statistical results of both modelling approaches differing only in the STD values. Therefore, the use of PU models in PV parameter estimation through metaheuristic methods can be a valid alternative to classical models. In the future, a more comprehensive study, considering cells and modules from other PV technologies, as well as extended irradiance and temperature levels, will contribute to better clarify the use of the PU formulations. Furthermore, the results obtained will always depend on the optimization algorithm used, whether by analytical, deterministic or metaheuristic methods. Thus, it is up to the author to choose the modelling approach that best meets the intended purpose.

Acknowledgements H.G.G. Nunes gives his special thanks to the Fundação para a Ciência e a Tecnologia (FCT), Portugal, for the Ph.D. Grant (SFRH/BD/140304/2018). This work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020.

References

1. IRENA.: Future of solar photovoltaic: deployment, investment, technology, grid integration and socio-economic aspects (A Global Energy Transformation: paper) (2019)
2. Villalva, M.G., Gazoli, J.R., Filho, E.R.: Comprehensive approach to modeling and simulation of photovoltaic arrays. IEEE Trans. Power Electron. **24**, 1198–1208 (2009). https://doi.org/10.1109/TPEL.2009.2013862
3. Ishaque, K., Salam, Z., Taheri, H.: Simple, fast and accurate two-diode model for photovoltaic modules. Sol. Energy Mater. Sol. Cells **95**, 586–594 (2011). https://doi.org/10.1016/j.solmat.2010.09.023
4. Khanna, V., Das, B.K., Bisht, D., et al.: A three diode model for industrial solar cells and estimation of solar cell parameters using PSO algorithm. Renew. Energy **78**, 105–113 (2015). https://doi.org/10.1016/j.renene.2014.12.072
5. Lim, L.H.I., Ye, Z., Ye, J., et al.: A linear identification of diode models from single I–V characteristics of PV panels. IEEE Trans. Ind. Electron. **62**, 4181–4193 (2015). https://doi.org/10.1109/TIE.2015.2390193
6. Soon, J.J., Low, K.-S.: Optimizing photovoltaic model for different cell technologies using a generalized multidimension diode model. IEEE Trans. Ind. Electron. **62**, 6371–6380 (2015). https://doi.org/10.1109/TIE.2015.2420617
7. Nunes, H.G.G., Pombo, J.A.N., Mariano, S.J.P.S., et al.: A new high performance method for determining the parameters of PV cells and modules based on guaranteed convergence particle swarm optimization. Appl. Energy **211**, 774–791 (2018). https://doi.org/10.1016/J.APENERGY.2017.11.078
8. Pindado, S., Cubas, J.: Simple mathematical approach to solar cell/panel behavior based on datasheet information. Renew. Energy **103**, 729–738 (2017). https://doi.org/10.1016/j.renene.2016.11.007
9. Qais, M.H., Hasanien, H.M., Alghuwainem, S.: Identification of electrical parameters for three-diode photovoltaic model using analytical and sunflower optimization algorithm. Appl. Energy **250**, 109–117 (2019). https://doi.org/10.1016/J.APENERGY.2019.05.013
10. Chen, X., Tianfield, H., Li, K.: Self-adaptive differential artificial bee colony algorithm for global optimization problems. Swarm. Evol. Comput. **45**, 70–91 (2019). https://doi.org/10.1016/J.SWEVO.2019.01.003
11. Easwarakhanthan, T., Bottin, J., Bouhouch, I., Boutrit, C.: Nonlinear minimization algorithm for determining the solar cell parameters with microcomputers. Int. J. Sol. Energy **4**, 1–12 (1986). https://doi.org/10.1080/01425918608909835
12. Tossa, A.K., Soro, Y.M., Azoumah, Y., Yamegueu, D.: A new approach to estimate the performance and energy productivity of photovoltaic modules in real operating conditions. Sol. Energy **110**, 543–560 (2014). https://doi.org/10.1016/j.solener.2014.09.043
13. Wu, L., Chen, Z., Long, C., et al.: Parameter extraction of photovoltaic models from measured I–V characteristics curves using a hybrid trust-region reflective algorithm. Appl. Energy **232**, 36–53 (2018). https://doi.org/10.1016/J.APENERGY.2018.09.161
14. Nunes, H.G.G., Pombo, J.A.N., Bento, P.M.R., et al.: Collaborative swarm intelligence to estimate PV parameters. Energy Convers. Manag. **185**, 866–890 (2019). https://doi.org/10.1016/j.enconman.2019.02.003
15. Laudani, A., Riganti Fulginei, F., Salvini, A.: High performing extraction procedure for the one-diode model of a photovoltaic panel from experimental I–V curves by using reduced forms. Sol. Energy **103**, 316–326 (2014). https://doi.org/10.1016/j.solener.2014.02.014
16. Panchal, A.K.: A per-unit-single-diode-model parameter extraction algorithm: a high-quality solution without reduced-dimensions search. Sol. Energy **207**, 1070–1077 (2020). https://doi.org/10.1016/j.solener.2020.07.051
17. Alam, D.F., Yousri, D.A., Eteiba, M.B.: Flower pollination algorithm based solar PV parameter estimation. Energy Convers. Manag. **101**, 410–422 (2015). https://doi.org/10.1016/j.enconman.2015.05.074

18. Kiani, A.T., Nadeem, M.F., Ahmed, A., et al.: Optimal PV parameter estimation via double exponential function-based dynamic inertia weight particle swarm optimization. Energies **13**, 4037 (2020). https://doi.org/10.3390/en13154037
19. Xiong, G., Zhang, J., Shi, D., et al.: Parameter extraction of solar photovoltaic models with an either-or teaching learning based algorithm. Energy Convers. Manag. **224**, 113395 (2020). https://doi.org/10.1016/j.enconman.2020.113395
20. Diab, A.A.Z., Sultan, H.M., Aljendy, R., et al.: Tree growth based optimization algorithm for parameter extraction of different models of photovoltaic cells and modules. IEEE Access **8**, 119668–119687 (2020). https://doi.org/10.1109/ACCESS.2020.3005236
21. Agwa, A.M., El-Fergany, A.A., Maksoud, H.A.: Electrical characterization of photovoltaic modules using farmland fertility optimizer. Energy Convers. Manag. **217**, 112990 (2020). https://doi.org/10.1016/j.enconman.2020.112990
22. Zhang, Y., Ma, M., Jin, Z.: Comprehensive learning Jaya algorithm for parameter extraction of photovoltaic models. Energy **211**, 118644 (2020). https://doi.org/10.1016/j.energy.2020.118644
23. Hao, Q., Zhou, Z., Wei, Z., Chen, G.: Parameters identification of photovoltaic models using a multi-strategy success-history-based adaptive differential evolution. IEEE Access **8**, 35979–35994 (2020). https://doi.org/10.1109/ACCESS.2020.2975078
24. DIab, A.A.Z., Sultan, H.M., Do, T.D., et al.: Coyote optimization algorithm for parameters estimation of various models of solar cells and PV modules. IEEE Access **8**, 111102–111140 (2020). https://doi.org/10.1109/ACCESS.2020.3000770
25. Deotti, L.M.P., Pereira, J.L.R., da Silva Júnior, I.C.: Parameter extraction of photovoltaic models using an enhanced Lévy flight bat algorithm. Energy Convers. Manag. **221**, 113114 (2020). https://doi.org/10.1016/j.enconman.2020.113114
26. Zhang, Y., Huang, C., Jin, Z.: Backtracking search algorithm with reusing differential vectors for parameter identification of photovoltaic models. Energy Convers. Manag. **223**, 113266 (2020). https://doi.org/10.1016/j.enconman.2020.113266
27. Sallam, K.M., Hossain, M.A., Chakrabortty, R.K., Ryan, M.J.: An improved gaining-sharing knowledge algorithm for parameter extraction of photovoltaic models. Energy Convers. Manag. **237**, 114030 (2021). https://doi.org/10.1016/J.ENCONMAN.2021.114030
28. Zhang, H., Heidari, A.A., Wang, M., et al.: Orthogonal Nelder-Mead moth flame method for parameters identification of photovoltaic modules. Energy Convers. Manag. **211**, 112764 (2020). https://doi.org/10.1016/j.enconman.2020.112764
29. Yousri D, Rezk H, Fathy A (2020) Identifying the parameters of different configurations of photovoltaic models based on recent artificial ecosystem-based optimization approach. Int. J. Energy Res. 1–21. https://doi.org/10.1002/er.5747
30. Zhang, Y., Jin, Z., Mirjalili, S.: Generalized normal distribution optimization and its applications in parameter extraction of photovoltaic models. Energy Convers. Manag. **224**, 113301 (2020). https://doi.org/10.1016/j.enconman.2020.113301
31. Mohamed, A.W., Hadi, A.A., Mohamed, A.K.: Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. Int. J. Mach. Learn. Cybern. **11**, 1501–1529 (2020). https://doi.org/10.1007/S13042-019-01053-X
32. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997). https://doi.org/10.1023/A:1008202821328
33. Yang, X.-S.: Flower pollination algorithm for global optimization. In: International Conference on Unconventional Computing and Natural Computation, pp. 240–249. Springer, Berlin, Heidelberg (2012)
34. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
35. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/J.ADVENGSOFT.2013.12.007
36. Cotfas, D.T., Deaconu, A.M., Cotfas, P.A.: Hybrid successive discretisation algorithm used to calculate parameters of the photovoltaic cells and panels for existing datasets. IET Renew. Power Gener. 1–27 (2021). https://doi.org/10.1049/RPG2.12262

37. Chen, X., Yue, H., Yu, K.: Perturbed stochastic fractal search for solar PV parameter estimation. Energy **189**, 116247 (2019). https://doi.org/10.1016/j.energy.2019.116247
38. Yang, X., Gong, W., Wang, L.: Comparative study on parameter extraction of photovoltaic models via differential evolution. Energy Convers. Manag. **201**, 112113 (2019). https://doi.org/10.1016/j.enconman.2019.112113
39. Cotfas, D.T., Deaconu, A.M., Cotfas, P.A.: Application of successive discretization algorithm for determining photovoltaic cells parameters. Energy Convers. Manag. **196**, 545–556 (2019). https://doi.org/10.1016/J.ENCONMAN.2019.06.037
40. Long, W., Cai, S., Jiao, J., et al.: A new hybrid algorithm based on grey wolf optimizer and cuckoo search for parameter extraction of solar photovoltaic models. Energy Convers. Manag. **203**, 112243 (2020). https://doi.org/10.1016/j.enconman.2019.112243
41. Weng, X., Heidari, A.A., Liang, G., et al.: Laplacian Nelder-Mead spherical evolution for parameter estimation of photovoltaic models. Energy Convers. Manag. **243**, 114223 (2021). https://doi.org/10.1016/J.ENCONMAN.2021.114223
42. Nunes, H.G.G., Silva, P.N.C., Pombo, J.A.N., et al.: Multiswarm spiral leader particle swarm optimisation algorithm for PV parameter identification. Energy Convers. Manag. **225**, 113388 (2020). https://doi.org/10.1016/j.enconman.2020.113388

Space–Time Concept in Social Network Search Algorithm

Siamak Talatahari, Hadi Bayzidi, and Mehdi Bayzidi

Abstract Nowadays, social networks have become popular platforms in which users can interact with each other virtually. The relationship among users, their influence on each other, and similarity in their behavior are the main features of users in the social networks. These features are utilized in the development of an optimization method, which is called the Social Network Search (SNS) algorithm. The SNS method is a recently developed optimizer, which models the treatment of users in expressing their new views. Four novel optimization operators with new formulations are invented, which are called decision moods. These moods are named imitation, conversation, disputation, and innovation, which are real-world behaviors of users in social networks and model how users are affected and motivated to share their new views. This chapter embeds the concept of space–time in the implementation process of the SNS algorithm to developing space–time social network search (STSNS) algorithm. Single-objective, bound-constraint benchmark problems of IEEE congress on evolutionary computation 2014 (CEC 2014) are utilized to study the efficiency of the STSNS algorithm in solving challenging optimization problems. In addition, the results of the STSNS method is compared with a variety range of optimization methods, including eight state-of-the-art, eight popular, and eight novel methods from the literature. Two well-known non-parametric statistical methods, Friedman and Wilcoxon signed-rank, are utilized to analyze the performance of the developed method, and results demonstrate its superiority in dealing with most of the selected complex optimization problems.

Keywords Metaheuristic · Optimization algorithm · Social network search

S. Talatahari (✉) · H. Bayzidi
Department of Civil Engineering, University of Tabriz, Tabriz, Iran
e-mail: siamak.talat@gmail.com

S. Talatahari
Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW, Australia

M. Bayzidi
Department of Civil and Environmental Engineering, Tarbiat Modares University, Tehran, Iran

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_7

Abbreviations

ABC	Artificial bee colony
ACO	Ant colony optimization
AF	Admission factor
AOS	Atomic orbital search
CEC	Congress on evolutionary computation
CryStAl	Crystal structure algorithm
CGO	Chaos game optimization
CLPSO	Comprehensive learning PSO
CMA-ES	Covariance matrix adaptation evolutionary strategy
CS	Cuckoo search
CSA	Crow search algorithm
CSS	Charged system search
DE	Differential evolution
DMSPSO	Dynamic multi-swarm PSO
ES	Evolutionary strategy
FA	Firefly algorithm
FCDE	Fitness values classification DE
GA	Genetic algorithm
GWO	Grey wolf optimizer
HGS	Henry gas solubility
HHO	Harris hawks optimization
LIPS	Distance-based locally informed particle swarm
L-SHADE	Success-history based parameter adaptation DE with linear population size reduction
MGA	Material generation algorithm
NFEs	Number of function evaluations
PSO	Particle swarm optimization
PVADE	Population's variance-based adaptive DE
SNS	Social network search
STSNS	Space-time social network search
SPO	Stochastic paint optimizer
SSA	Salp swarm algorithm
SOS	Symbiotic organisms search
TLBO	Teaching-learning-based optimization
TS	Tabu search
WCA	Water cycle algorithm
WOA	Whale optimization algorithm
WSR	Wilcoxon signed-rank

1 Introduction

Optimization refers to the procedure of finding the best possible solution to a problem that aims at minimizing (maximizing) a cost function (fitness function) that corresponds to a specific purpose. Solving optimization problems need some methods, which are called optimization algorithms. Optimization algorithms are separated into two general groups: deterministic and stochastic approaches. Most of the deterministic methods are based on mathematical principles and calculate the gradient of the objective function to solve problems. Deterministic methods can detect the exact solution to the problems, however, these approaches are susceptible to the initial starting point, and a true starting point can direct the search process to the global optimum point. In addition, the execution time of these methods is increased exponentially according to the dimension of the problems. These methods cannot solve complex optimization problems and mostly find local solutions. On the other, stochastic optimization methods generate and employ random variables to solve optimization problems. Randomized search methods, which are called metaheuristic algorithms, are one of the most famous types of stochastic optimization methods. Metaheuristic methods utilize random variables during the search process to speed up the search progress, and this randomness makes the search method less sensitive to modeling errors and enables them to escape local optimums. The main advantage of metaheuristic methods is that they are gradient-free and do not employ the gradient information of the objective function. Metaheuristic algorithms are problem-independent, and they can be utilized for solving all types of optimization problems [1].

Metaheuristic algorithms are based on two principles: heuristic rules and randomness in the search process. The word 'heuristic' originates from 'heuriskein', which is an old Greek word that means the ability in the invention of intelligence processes for solving problems. In the context of metaheuristics, the heuristic procedures are taken from the intelligence behavior of natural phenomena and creatures. Combine the basic heuristic methods with randomization, embed an upper level of methodology in the heuristic rules, and able them to solve a problem more accurately. In addition, the presence of suffix 'Meta', is due to adding the randomization in the structure of the heuristic methods [2].

It can be said that the initial study in the field of metaheuristic methods is related to the development of evolutionary strategy (ES) [3] in the early 1960s in which the process of natural evolution is simulated as an optimization algorithm. Holland developed the genetic algorithm (GA) [4] in the early 1970s based on the Darwinian Theory about the evolution. Glover proposed the tabu search (TS) [5, 6] in 1986, and he used the word 'metaheuristic' for the first time to refer to algorithms that solve optimization problems according to natural principles. In 1992, Dorigo developed the ant colony optimization (ACO) [7]; in 1995, Kennedy and Eberhart invented the particle swarm optimization (PSO) [8]; and in 1997, Storn and Price proposed the differential evolution (DE) [9]. These methods can be known as the basements of modern metaheuristic algorithms, in which the concepts of heuristic computation are

based on their principles. After 1997, various methods were introduced based on the intelligence behavior of natural phenomena for solving optimization problems. Algorithms like artificial bee colony (ABC) [10], firefly algorithm (FA) [11], cuckoo search (CS) [12], charged system search (CSS) [13], teaching–learning-based optimization (TLBO) [14], crow search algorithm (CSA) [15], water cycle algorithm (WCA) [16], grey wolf optimizer (GWO) [17], and symbiotic organisms search (SOS) [18], are among the most popular methods. Besides, algorithms such as salp swarm algorithm (SSA) [19], whale optimization algorithm (WOA) [20], harris hawks optimization (HHO) [21], stochastic paint optimizer (SPO) [22], chaos game optimization (CGO) [2, 23], atomic orbital search (AOS) [24, 25], crystal structure algorithm (CryStAl) [26], social network search (SNS) [1], and material generation algorithm (MGA) [27] are among the most recent metaheuristic methods.

As it is seen, the study in the field of metaheuristic methods is an active research area. It is noteworthy that each of these algorithms can deal with problems differently, such that one method may not find the optimum solution for some problems. Therefore, developing novel high-performance optimization algorithms is necessary. New algorithms should solve complex and large-scale optimization problems in less time in comparison with previously developed ones. These goals are met when the developed algorithms have higher capability in searching the space of problems.

The SNS algorithm is a new metaheuristic algorithm that developed for solving complex problems. The results of this method demonstrated that it is capable in outperforming different metaheuristics in solving various types of optimization problems [1, 28]. The SNS algorithm is developed based on the behavior of users in social networks. In social networks, users affect the thoughts of other people by sharing their views. In this chapter, the concept of space–time is integrated into the structure of the SNS algorithm, and the space–time social network search (STSNS) algorithm is developed. The performance of the STSNS algorithm is investigated in dealing with 30 single-objective, bound-constraint benchmark problems of IEEE congress on evolutionary computation 2014 (CEC 2014) [29]. The results of the STSNS method in dealing with these complex optimization benchmark problems are compared with various optimization algorithms, including eight state-of-the-art, eight popular, and eight novel methods from the literature. To have a valid judge about the capability of the STSNS method in solving these challenging optimization problems, two well-known non-parametric statistical methods, Friedman and Wilcoxon signed-rank, are utilized to analyze the performance of the STSNS method. The performance of the STSNS method shows its advantage in dealing with most of the selected problems.

The rest of this chapter is organized as follows: Sect. 2 reviews the basic concepts of the social network search algorithm. Section 3 discusses the space–time concept and utilizes it to develop the STSNS algorithm. Section 4 presents the numerical results, and Sect. 5 concludes the chapter.

2 Social Network Search Algorithm

Human beings always attempt to associate with each other. Virtual tools called social networks are created for this purpose with the advent of technology. SNS models the intelligent pattern of interactions among users in social networks. On these platforms, users interact with each other virtually. The opinions of users are shared during these social interactions, which may affect the view of other people. The interaction between users in social networks has an optimal pattern in which users try to improve their popularity level among others in the network.

The main property of social networks is their high capability in propagating new ideas. Figure 1 models the communication of users in networks in which each individual follows some people. This characteristic has made networks a powerful tool for publishing information.

In the process of the SNS algorithm, the views of users will be influenced by other peoples during four moods which are called: (1) imitation, (2) conversation, (3) disputation, and (4) innovation. These moods mimic the optimal pattern of users in gaining more popularity by sharing their views on social networks. Definition and mathematical formulation of these patterns are presented in the following subsections [1].

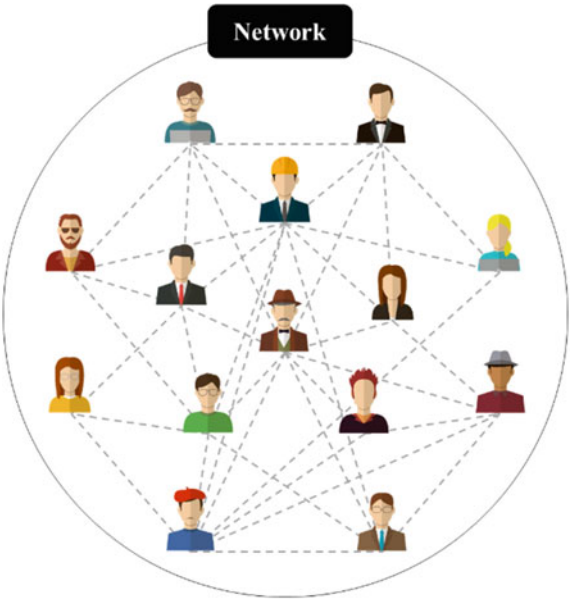


Fig. 1 A schematic model of the connection between users in social networks

2.1 Mood 1: Imitation

In social networks, users like to mimic the opinion of others in publishing their views. In the SNS algorithm, this mood is called imitation, and its process is modeled as follows:

$$\begin{aligned} X_i^{new} &= X_j + rand(-1, 1) \times \mathrm{R} \\ R &= rand(0, 1) \times r \\ r &= X_j - X_i \end{aligned} \tag{1}$$

where, X_i is the opinion vector of the ith user and X_j is the view vector of the jth user that selected randomly somehow $i \neq j$; $rand(-1, 1)$ is a random vector in interval $[-1, 1]$ and $rand(0, 1)$ is a random vector in interval $[0, 1]$. In the imitation mood, the new view $\left(X_i^{new}\right)$ is created based on the imitation space (Fig. 2a). In addition, imitation space is modeled according to the radii of shock and popularity. Shock radius (R) shows the influence of the jth user on the opinion of the ith individual in developing new view, and its value is as a multiple of r. The magnitude of the popularity radius (r) represents the popularity of the jth user. The distance between X_i and X_j determine the value of popularity radius. The search direction (shock radius effect) in the imitation mood is determined by multiplying the value of R to $rand(-1, 1)$, in which positive components of the random vector mean that those dimensions of the new view have the same direction as the jth user and vice versa. Figure 2a illustrates the procedure of the first mood. Based on this procedure, at first, the imitation space is formed, and then a new idea is shared on the network.

2.2 Mood 2: Conversation

The second state is the conversation mood in which users communicate with each other and speak about various subjects. During this type of communication, users benefit from the idea of other users and share their new views as follows:

$$\begin{aligned} X_i^{new} &= X_k + R \\ R &= rand(0, 1) \times D \\ D &= sign\left(f_i - f_j\right) \times \left(X_j - X_i\right) \end{aligned} \tag{2}$$

where, X_j the view vector of the jth user, X_k is the vector of an issue, which is randomly chosen to speak about it (j and k are selected randomly somehow $i \neq j \neq k$), R is the result of conversation according to the difference between the opinions of ith and jth users, and D is the difference between the opinions of users. The value of D shows the variation in the beliefs of ith and jth users about the X_k. Also, f_i

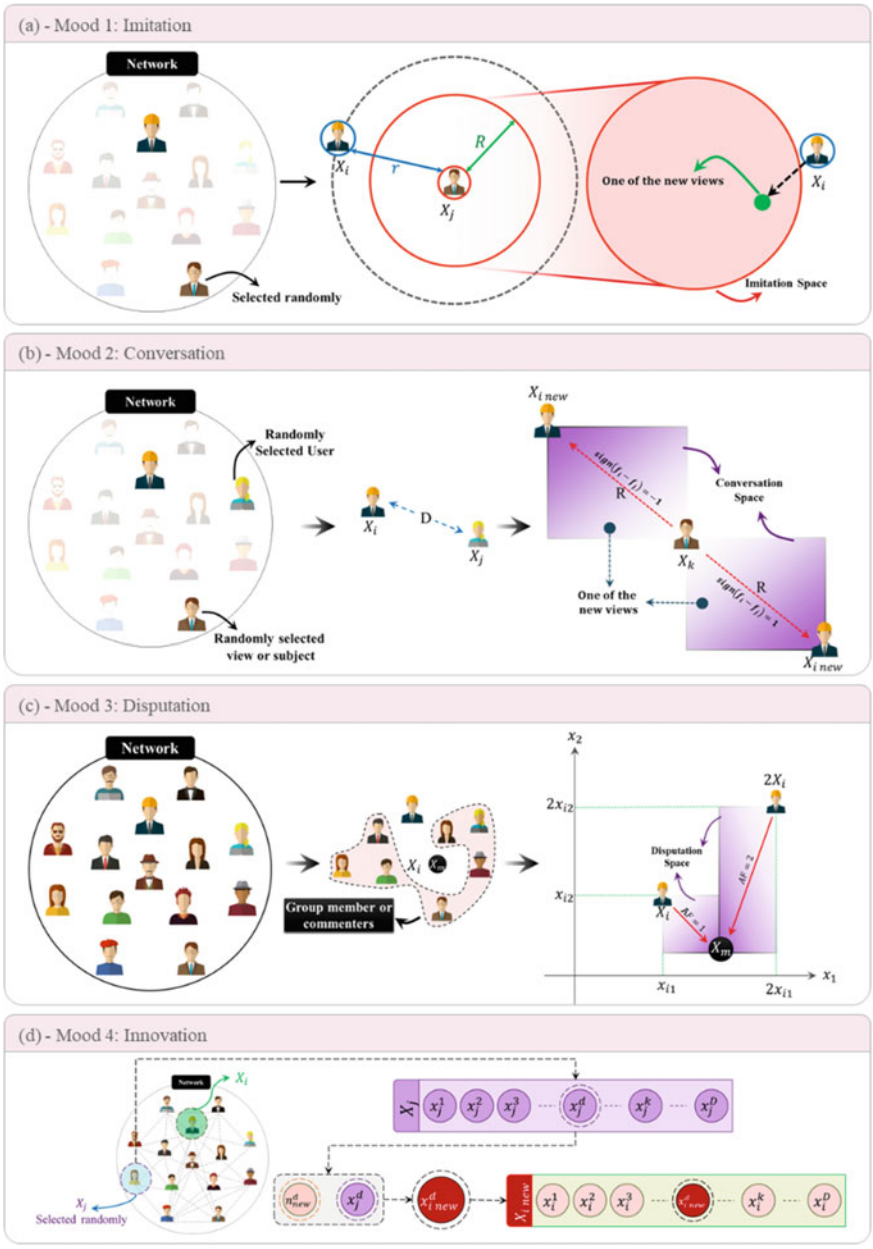


Fig. 2 The processes of decision moods in the SNS algorithm. **a** Imitation, **b** Conversation, **c** Disputation, and **d** Innovation

and f_j are the objective functions of X_i and X_j, respectively, and $sign(.)$ is the sign function.

In this mood users benefit from the ideas of the other individuals. Besides, $sign(f_i - f_j)$ shows the evolution orientation of issue vector (X_k). Figure 2b illustrates the procedure of the conversation mood. Based on this process, the *i*th view about the issue is changed according to $sign(f_i - f_j)$. Changing the user's view about the events is considered as the relocation of the events.

2.3 Mood 3: Disputation

Disputation defines a state in which users present their ideas about events to a group of individuals and defend their opinion. In disputation mood, users will be familiar with the views of other individuals and will be affected. The mathematical model of this mood is presented as follows:

$$\begin{aligned} X_i^{new} &= X_i + rand(0,1) \times (M - AF \times X_i) \\ M &= \frac{\sum_t^{N_r} X_t}{N_r} \\ AF &= 1 + round(rand) \end{aligned} \tag{3}$$

where, $rand(0, 1)$ is a random vector in [0, 1], and N_r is the number of individuals that take part in discussions, in which the contributors are elected randomly. AF is the Admission Factor that is a random number that can be either 1 or 2. Figure 2c shows the procedure of this mood.

2.4 Mood 4: Innovation

Innovation means that sometimes the shared subject originated based on new ideas. During the innovation mood, new views are developed by altering a random variable of X_i. The process of this state is formulated as follows:

$$\begin{aligned} X_i^{new} &= \left[x_{\mathrm{i},1}, x_{\mathrm{i},2}, x_{\mathrm{i},3}, \ldots . x_{i,d} \ldots . x_D\right] \\ x_{i,d}^{new} &= t \times x_{j,d} + (1-t) \times n_{new,d} \\ n_{new,d} &= lb_d + rand_1 \times (ub_d - lb_d) \\ t &= rand_2 \end{aligned} \tag{4}$$

where, d is a randomly selected design variable in the interval $(1, D)$, which D is the number of design variables. $rand_1$ and $rand_2$ are random numbers in the interval [0,

1], and ub_d and lb_d are maximum and minimum values for the dth variable. $n_{d,new}$ is the new view from the perspective of the dth design variable. $x_{j,d}$ is the prevalent view about the dth variable introduced by the jth user (jth individual that is chosen randomly somehow $i \neq j$).

Innovation models a condition that someone thinks about a subject, maybe in a new way, and presents a novel definition from the nature of that subject. Each subject has special characteristics, and changing one of them can influence the general concept of the subject. Therefore, by alteration the view about one of them $\left(x_{i,d}\right)$, the substantial definition of the subject will change, and consequently a new idea can be derived. $x_{i,\mathrm{d}}^{new}$ is a novel understanding about the issue under consideration from the dth standpoint and swap with the current view $\left(x_{i,d}\right)$. Figure 2d shows the process of new view construction during the innovation mood.

In each iteration of the SNS, one of these moods is selected randomly for creating new views. In other words, the proper assumption is that just one of these moods happens at a specific time (iteration) for each user. The chance of each mood is determined during a random process with a uniform distribution. Figure 3 illustrates the flowchart of this method, and its MATLAB code is available at [30, 31].

3 Space–Time Based Social Network Search (STSNS)

Space and time are the main components in studying the behavior of physical phenomena. In physics, space and time are combined into a single construct by a mathematical model called space–time. This model allows studying the physical process in a four-dimensional space in which space is three-dimensional and time plays the role of the fourth dimension. On the other, a field is a physical quantity that directs the motion of particles associated with a point in space at a specific time (for instance, the relocation of objects in the gravitational field). The description of some space–time principals and their equal concepts in the field of metaheuristics are presented as follows [32]:

- **Probe**: In population-based metaheuristic algorithms, each agent is considered as a particle or probe that moves in the search space of the problem. In the search space (or view space in the SNS algorithm), two d-dimensional vector determines the position of each probe at two times: current time and previous times.
- **Space–time**: This concept is utilized for the search space at a specified time. The number of the design variables determines the dimension of space–time.
- **Time**: In the iterative optimization algorithms, the term iteration is employed for the time. In addition, the value of time is changed discretely in metaheuristic algorithms.

In the basic SNS, the updating process of the algorithm is repeated after creating a new view. Therefore, the updated view of each user will affect the next position of other agents in the same iteration. In this chapter, an enhanced version of the SNS

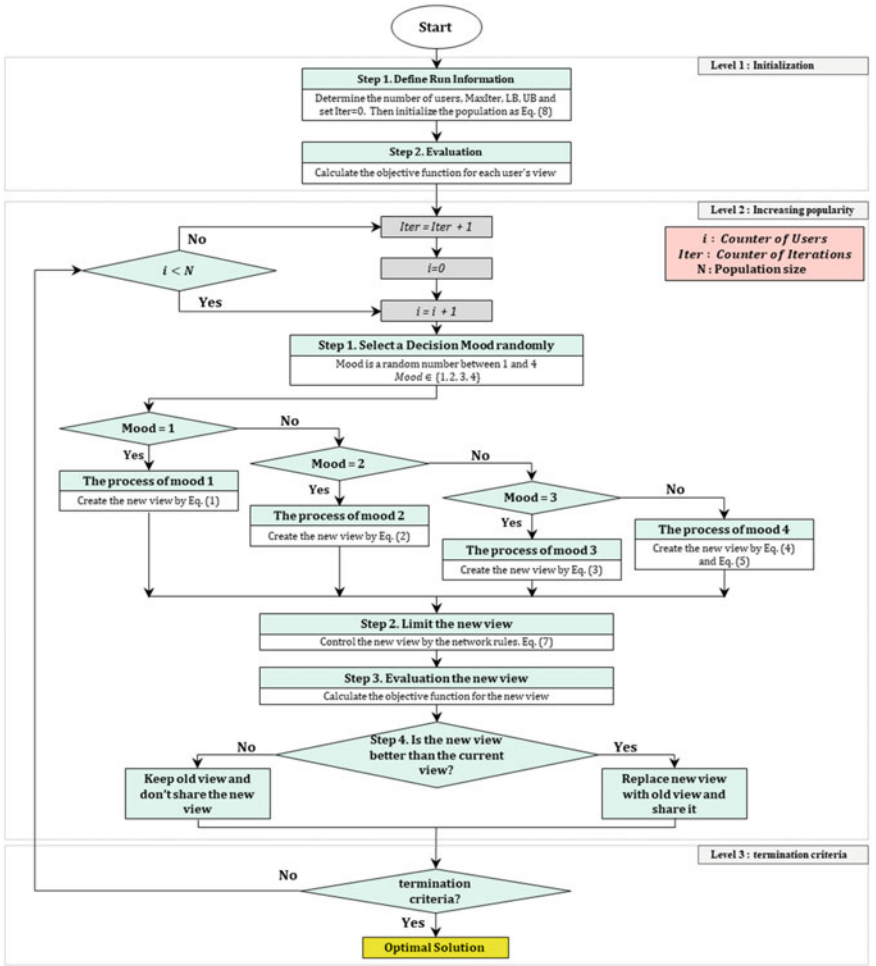


Fig. 3 The flowchart of the SNS algorithm

algorithm is developed utilizing the concept of discrete space–time called space–time based social network search (STSNS) algorithm. In the STSNS algorithm, the updating process is performed after evaluating all the new views. In this type of space–time, all positions are updated when all users evaluate their new views. Based on this inference, the flowchart of the STSNS algorithm is presented in Fig. 4.

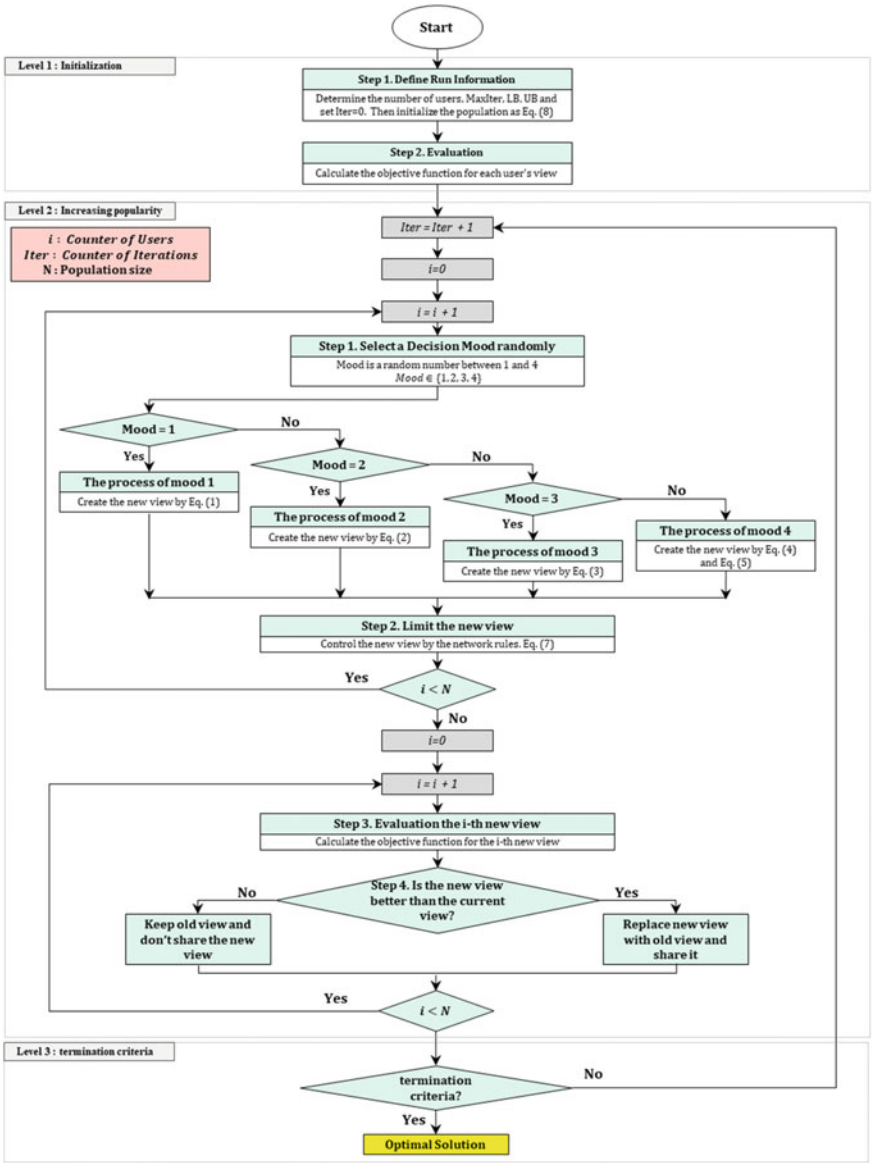


Fig. 4 The flowchart of the STSNS algorithm

4 Numerical Results

In this section, the performance of the STSNS algorithm is investigated in dealing with different optimization problems and compares its results with various methods from the literature. The utilized optimization problems, contender metaheuristic algorithms, evaluation criteria, numerical results, comparisons, and discussions are presented in the following subsections.

4.1 Benchmark Optimization Problems

In testing the capability of an algorithm, a set of qualified optimization problems should be utilized, which contain benchmark functions with various characteristics. The employed set should be able to simulate the complexity of real-world optimization problems. To this aim, several attempts have been made in the literature to innovate novel approaches for developing complex optimization problems. Congaree of Evolutionary Computation (CEC) developed one of the most successful approaches, in which they utilized interesting ways to increase the complexity of traditional benchmark optimization problems.

In the present study, CEC 2014 special season on single objective real-parameter numerical optimization problems are employed for evaluating the ability of the STSNS algorithm. These problems consists of 30 complex mathematical test functions. These problems include three rotated, thirteen shifted and rotated, six hybrids, and eight composite test functions. Table 1 presents the list of discussed benchmark problems, and their mathematical details are presented by the CEC 2014 competition committee in [29].

4.2 Contender Methods

Selecting proper metaheuristic algorithms for determining the performance of an algorithm is an important task. In other words, the real performance of a method becomes obvious when compared with a variety range of valid optimization methods. In this chapter, various valuable algorithms are selected from the literature that are categorized into three groups: (1) state-of-the-art methods; (2) popular algorithms; and (3) newly developed optimizers. Therefore, STSNS is face with three experiments as follow:

Experiment 1: Comparing with state-of-the-art algorithms. The first group contains the PSO [8], dynamic multi-swarm PSO (DMSPSO) [33], comprehensive learning PSO (CLPSO) [34], distance-based locally informed particle swarm [35] (LIPS), DE, fitness values classification DE (FCDE) [36], success-history based parameter adaptation DE with linear population size reduction (L-SHADE) [37], and

Table 1 Details of the CEC 2014 special season benchmark problems

No	Function	Range	D	Min
F1	Rotated high conditioned elliptic function	[−100, 100]	30	0
F2	Rotated bent cigar function	[−100, 100]	30	0
F3	Rotated discus function	[−100, 100]	30	0
F4	Shifted and rotated Rosenbrock's function	[−100, 100]	30	0
F5	Shifted and rotated Ackley's function	[−100, 100]	30	0
F6	Shifted and rotated Weierstrass function	[−100, 100]	30	0
F7	Shifted and rotated Griewank's function	[−100, 100]	30	0
F8	Shifted Rastrigin's function	[−100, 100]	30	0
F9	Shifted and rotated Rastrigin's function	[−100, 100]	30	0
F10	Shifted Schwefel's function	[−100, 100]	30	0
F11	Shifted and rotated Schwefel's function	[−100, 100]	30	0
F12	Shifted and rotated Katsuura function	[−100, 100]	30	0
F13	Shifted and rotated HappyCat function	[−100, 100]	30	0
F14	Shifted and rotated HGBat function	[−100, 100]	30	0
F15	Shifted and rotated expanded Griewank's plus Rosenbrock's function	[−100, 100]	30	0
F16	Shifted and rotated expanded Schaffer's F6 function	[−100, 100]	30	0
F17	Hybrid function 1 (N = 3)	[−100, 100]	30	0
F18	Hybrid function 2 (N = 3)	[−100, 100]	30	0
F19	Hybrid function 3 (N = 4)	[−100, 100]	30	0
F20	Hybrid function 4 (N = 4)	[−100, 100]	30	0
F21	Hybrid function 5 (N = 5)	[−100, 100]	30	0
F22	Hybrid function 6 (N = 5)	[−100, 100]	30	0
F23	Composition function 1 (N = 5)	[−100, 100]	30	0
F24	Composition function 2 (N = 3)	[−100, 100]	30	0
F25	Composition function 3 (N = 3)	[−100, 100]	30	0
F26	Composition function 4 (N = 5)	[−100, 100]	30	0
F27	Composition function 5 (N = 5)	[−100, 100]	30	0
F28	Composition function 6 (N = 5)	[−100, 100]	30	0
F29	Composition function 7 (N = 3)	[−100, 100]	30	0
F30	Composition function 8 (N = 3)	[−100, 100]	30	0

Range: variable range
D: variable dimension
Min: global minimum value

population's variance-based adaptive DE (PVADE) [38]. The employed algorithms in this experiment can be considered as state-of-the-art metaheuristic methods in the literature.

Experiment 2: Comparing with popular algorithms. The second challenging issue is determining the capability of the STSNS algorithm in comparison with popular methods, which are developed in the last two decades. Covariance matrix adaptation evolutionary strategy (CMA-ES) [39], FA, CS, TLBO, CSA, WCA, GWO, and WOA are selected for this propose.

Experiment 3: Comparing with newly developed algorithms. Comparing with algorithms that are developed in the last years is the third experiment. SNS, SPO, CryStAl, AOS, CGO, SOS, HHO, and henry gas solubility (HGS) [40] are selected to this aim.

According to the mentioned details, the STSNS will participate in three comparisons. In each comparison, the STSNS will compete with eight different algorithms. Therefore, the performance of the STSNS can be properly identified.

4.3 Evaluation Criteria

In the present study, 30-dimensional benchmark problems of CEC 2014 are considered for comparisons. According to CEC 2014 instructions, each of the algorithms must run 51 times, since the result of one optimization run is not adequate in evaluating the efficiency of an optimization algorithm. In addition, the maximum number of function evaluations (NFEs) is considered as $10^4 \times \mathrm{D}$ ($\mathrm{D} = 30$) for all of the algorithms and the error of 1×10^{-8} from the global optimum is considered as the threshold value. In other words, based on these criteria, if the best solution of a method reaches a tolerance less than the threshold value, the search process stops, otherwise, the search procedure will be continued until the maximum NFE reaches to 300,000 evaluations.

Some contender algorithms have specific parameters. The values of these parameters should be determined sensitively since they significantly influence the performance of algorithms. In this chapter, the parameters of algorithms are selected according to the previously published works. Also, our simulation results indicate that the utilized values can be employed, confidently.

All employed algorithms are population-based methods, and their population size should be defined based on the capability of algorithms and the complexity of problems. According to the author's experience, the best value for population size is 80 in solving 30-dimensional CEC 2014 benchmark problems, and this value is considered for all of the algorithms except for methods that their previously published papers prosed another values.

4.4 Non-Parametric Statistical Tests

For evaluating the performance of an algorithm, the result of one optimization run is not adequate since metaheuristic algorithms have a random nature. Therefore, each of the contender algorithms runs more than one time, independently for each problem. Consequently, a set of results is achieved for each problem, and comparing the performance of the algorithms using common statistical information such as mean and standard deviation cannot be determined well.

Non-parametric statistical tests are a family of statistical methods that can determine the difference between two or more sets of data. These statistical methods are utilized to determine which algorithm performs better. Non-parametric statistical tests are categorized into two groups: pairwise comparisons and multiple comparisons. The first type looks into the results of two methods, while the multiple comparisons analyze the performance of more than two algorithms. In this chapter, the Wilcoxon signed-rank (WSR) and the Friedman tests are utilized in comparisons.

4.5 Results and Discussions

This subsection presents the statistical results of STSNS and other methods in dealing with the selected optimization problems. Tables 2, 3 and 4 present the performance of algorithms (including mean and standard deviation (Std. Dev.) of the best solutions during optimization runs for each problem). In addition, the WSR test is performed as a pairwise comparison between the results of STSNS and other contender algorithms on each of the problems using the results of 51 independent runs. The results of this non-parametric test are reflected by '+', '−', and '=', where '+' denotes the results of the STSNS is better than the contender algorithm, '−' means that the results of peer method is better than the STSNS, and '=' shows that there is no significant difference in performance. The summary results of the WSR test are presented in the last row of each table.

The performance of STSNS and eight state-of-the-art algorithms are presented in Table 2. Based on these results, STSNS can outperform PSO, DMSPSO, CLPSO, LIPS, DE, FCDE, PVADE, and L-SHADE on 22, 19, 24, 20, 21, 20, 13, and two problems out of 30, respectively. SNS performs worse than PSO, DMSPSO, CLPSO, LIPS, DE, FCDE, PVADE, and L-SHADE on four, seven, two, seven, eight, ten, nine, and 26 problems, respectively, and in some cases, there is no significant difference between SNS and other contender methods. This experiment shows that the STSNS algorithm faces two main contenders, PVADE, and L-SHADE. The results of the WSR test demonstrates that the SNS can outperforms PVADE in most of problems, while L-SHADE outperforms the SNS and all of the state-of-the-art algorithms in most problems.

Table 3 summarizes the statistical results of STSNS and popular algorithms (second group of contender methods). Based on the summarized results of the WSR

Table 2 Comparing the statistical results of SNS and state-of-the-art algorithms on 30-D CEC2014 problem suite

No.	STSNS		PSO			DMSPSO			CLPSO			LIPS		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F1	4.73E+05	2.68E+05	1.99E+06	2.32E+06	+	3.95E+06	2.45E+06	+	1.62E+07	4.94E+06	+	1.67E+07	2.05E+07	+
F2	5.91E+00	1.03E+01	1.69E+08	5.03E+08	=	1.94E+03	1.65E+03	+	5.01E+03	1.59E+03	+	1.43E+03	2.90E+03	+
F3	1.04E+02	2.28E+02	3.74E+00	6.41E+00	−	5.43E+02	5.01E+02	+	3.11E+02	2.23E+02	+	1.87E+03	1.88E+03	+
F4	5.82E+01	3.24E+01	9.37E+01	3.66E+01	+	8.30E+01	1.83E+01	+	1.04E+02	1.06E+01	+	1.84E+02	8.63E+01	+
F5	2.04E+01	3.52E-02	2.03E+01	3.75E-01	−	2.03E+01	4.84E-02	−	2.06E+01	5.54E-02	+	2.00E+01	2.11E-04	−
F6	4.46E+00	1.93E+00	1.18E+01	3.47E+00	+	5.54E+00	2.43E+00	+	1.63E+01	1.33E+00	+	1.20E+01	3.00E+00	+
F7	1.43E-02	1.55E-02	2.34E+00	4.43E+00	=	6.66E-03	8.73E-03	−	1.32E-02	5.36E-03	=	4.48E-04	1.75E-03	−
F8	7.08E+00	1.17E+00	8.75E+01	1.65E+01	+	2.33E+01	5.98E+00	+	7.36E-04	4.33E-04	−	3.96E+01	1.03E+01	+
F9	6.89E+01	1.47E+01	9.84E+01	2.33E+01	+	3.98E+01	1.09E+01	−	8.01E+01	7.50E+00	+	4.60E+01	1.27E+01	−
F10	7.08E+01	1.30E+01	2.03E+03	4.94E+02	+	6.56E+02	3.00E+02	+	1.47E+01	3.03E+00	−	1.74E+03	3.20E+02	+
+11	2.83E+03	2.77E+02	3.33E+03	6.44E+02	+	2.56E+03	5.18E+02	−	3.32E+03	3.15E+02	+	2.45E+03	3.17E+02	−
F12	5.61E-01	6.52E-02	2.34E-01	7.46E-02	−	6.47E-01	1.01E-01	+	6.27E-01	6.96E-02	+	1.37E-01	3.60E-02	−
F13	2.85E-01	4.67E-02	4.74E-01	1.14E-01	+	2.12E-01	3.79E-02	−	3.04E-01	4.31E-02	+	2.69E-01	4.81E-02	=
F14	2.37E-01	3.54E-02	7.78E-01	1.49E+00	+	2.99E-01	1.41E-01	+	2.67E-01	3.24E-02	+	2.40E-01	4.21E-02	=
F15	7.73E+00	3.33E+00	5.57E+00	4.42E+00	−	9.13E+00	1.56E+00	+	1.09E+01	1.16E+00	+	6.95E+00	2.60E+00	=
F16	1.02E+01	3.15E-01	1.13E+01	6.94E-01	+	1.03E+01	5.08E-01	=	1.10E+01	2.78E-01	+	1.10E+01	4.80E-01	+
F17	3.67E+04	2.81E+04	1.86E+05	2.64E+05	+	2.76E+05	1.88E+05	+	1.24E+06	4.73E+05	+	2.29E+05	1.61E+05	+
F18	1.12E+03	1.54E+03	2.96E+03	4.48E+03	+	2.93E+04	1.62E+05	+	3.66E+02	1.59E+02	=	2.44E+02	2.16E+02	−
F19	1.10E+01	1.62E+01	1.30E+01	1.13E+01	+	7.06E+00	1.20E+00	=	1.03E+01	9.06E-01	+	1.43E+01	8.22E+00	+
F20	3.37E+02	2.18E+02	3.79E+02	8.44E+02	=	7.10E+02	3.56E+02	+	3.54E+03	1.52E+03	+	1.05E+04	6.21E+03	+

(continued)

Table 2 (continued)

No.	STSNS		PSO			DMSPSO			CLPSO			LIPS		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F21	1.18E+04	6.93E+03	8.54E+04	8.62E+04	+	7.92E+04	6.20E+04	+	1.19E+05	5.46E+04	+	6.95E+04	5.12E+04	+
F22	1.59E+02	8.46E+01	4.23E+02	1.68E+02	+	1.62E+02	7.45E+01	=	1.76E+02	6.20E+01	=	2.24E+02	9.27E+01	+
F23	3.08E+02	2.71E+01	3.16E+02	2.94E+00	=	3.15E+02	6.82E-01	–	3.15E+02	6.25E-03	+	3.20E+02	2.92E+00	+
F24	2.00E+02	1.03E-03	2.24E+02	5.95E+00	+	2.25E+02	1.06E+00	+	2.26E+02	5.73E-01	+	2.34E+02	4.46E+00	+
F25	2.00E+02	0.00E+00	2.10E+02	3.74E+00	+	2.06E+02	2.39E+00	+	2.10E+02	8.95E-01	+	2.14E+02	2.25E+00	+
F26	1.04E+02	1.94E+01	1.53E+02	4.97E+01	+	1.12E+02	3.22E+01	–	1.00E+02	6.71E-02	+	1.08E+02	2.37E+01	+
F27	4.76E+02	1.07E+02	5.96E+02	1.63E+02	+	4.22E+02	3.97E+01	=	4.27E+02	7.23E+00	+	5.09E+02	1.31E+02	+
F28	8.42E+02	4.64E+01	1.82E+03	8.14E+02	+	1.10E+03	2.33E+02	+	9.52E+02	5.84E+01	+	1.57E+03	3.16E+02	+
F29	6.76E+05	2.31E+06	2.63E+06	6.88E+06	+	4.24E+06	5.56E+06	+	1.72E+03	2.85E+02	=	5.34E+03	2.73E+04	–
F30	2.43E+03	1.01E+03	5.55E+03	6.85E+03	+	2.14E+04	3.05E+04	+	4.82E+03	1.15E+03	+	2.76E+04	1.25E+04	+
	+/=/–		22/4/4			19/4/7			24/4/2			20/3/7		
No.	STSNS		DE			FCDE			PVADE			L-SHADE		
	Mean	Std. Dev	Mean	Std. Dev	W	Mean	Std. Dev	W	Mean	Std. Dev	W	Mean	Std. Dev	W
F1	4.73E+05	2.68E+05	9.93E+04	8.75E+04	+	1.05E+05	9.01E+04	–	1.82E+04	1.58E+04	–	0.00E+00	0.00E+00	–
F2	5.91E+00	1.03E+01	6.32E-02	5.36E-02	–	0.00E+00	0.00E+00	–	5.39E+02	1.92E+03	=	0.00E+00	0.00E+00	–
F3	1.04E+02	2.28E+02	1.38E-06	1.33E-06	–	1.09E+02	4.01E+02	–	0.00E+00	0.00E+00	–	0.00E+00	0.00E+00	–
F4	5.82E+01	3.24E+01	1.18E+00	6.43E-01	–	2.99E+01	3.02E+01	–	4.84E+01	3.29E+01	=	0.00E+00	0.00E+00	–
F5	2.04E+01	3.52E-02	2.08E+01	8.45E-02	+	2.09E+01	7.59E-02	+	2.06E+01	4.78E-02	+	2.01E+01	3.65E-02	–
F6	4.46E+00	1.93E+00	7.85E+00	3.41E+00	+	2.14E+01	3.52E+00	+	4.55E+00	1.84E+00	=	1.38E-07	9.79E-07	–

(continued)

Table 2 (continued)

No.	STSNS		DE			FCDE			PVADE			L-SHADE		
	Mean	Std. Dev	Mean	Std. Dev	W	Mean	Std. Dev	W	Mean	Std. Dev	W	Mean	Std. Dev	W
F7	1.43E-02	1.55E-02	9.00E-08	8.19E-08	–	2.79E-02	2.71E-02	+	2.71E-02	3.00E-02	+	0.00E+00	0.00E+00	–
F8	7.08E+00	1.17E+00	1.68E+02	2.44E+01	+	9.49E+01	2.21E+01	+	5.49E+01	8.88E+00	+	0.00E+00	0.00E+00	–
F9	6.89E+01	1.47E+01	2.02E+02	1.13E+01	+	1.35E+02	3.11E+01	+	8.26E+01	1.12E+01	+	6.78E+00	1.47E+00	–
F10	7.08E+01	1.30E+01	4.06E+03	9.43E+02	+	2.27E+03	5.30E+02	+	2.08E+03	4.71E+02	+	1.63E-02	1.56E-02	–
F11	2.83E+03	2.77E+02	6.45E+03	8.02E+02	+	3.49E+03	6.73E+02	+	3.85E+03	3.03E+02	+	1.23E+03	1.81E+02	–
F12	5.61E-01	6.52E-02	2.15E+00	5.19E-01	+	1.37E+00	5.44E-01	+	9.52E-01	1.32E-01	+	1.61E-01	2.27E-02	–
F13	2.85E-01	4.67E-02	5.03E-01	5.95E-02	+	5.68E-01	1.15E-01	+	2.96E-01	5.92E-02	=	1.24E-01	1.73E-02	–
F14	2.37E-01	3.54E-02	3.10E-01	5.62E-02	+	4.44E-01	2.25E-01	+	2.24E-01	4.36E-02	=	2.42E-01	2.95E-02	=
F15	7.73E+00	3.33E+00	1.75E+01	1.29E+00	+	1.54E+01	6.63E+00	+	7.22E+00	1.14E+00	=	2.15E+00	2.49E-01	–
F16	1.02E+01	3.15E-01	1.27E+01	3.42E-01	+	1.22E+01	5.68E-01	+	1.05E+01	4.41E-01	+	8.50E+00	4.54E-01	–
F17	3.67E+04	2.81E+04	1.86E+03	2.35E+02	+	6.54E+03	8.63E+03	–	1.23E+03	4.02E+02	–	1.88E+02	7.42E+01	–
F18	1.12E+03	1.54E+03	6.53E+01	1.01E+01	+	1.23E+02	6.63E+01	–	1.09E+02	3.85E+01	–	5.91E+00	2.86E+00	–
F19	1.10E+01	1.62E+01	5.59E+00	6.17E-01	+	1.32E+01	1.16E+01	+	7.79E+00	8.29E+00	=	3.68E+00	6.74E-01	–
F20	3.37E+02	2.18E+02	4.07E+01	1.22E+01	–	1.33E+02	7.91E+01	–	4.48E+01	2.59E+01	–	3.08E+00	1.45E+00	–
F21	1.18E+04	6.93E+03	8.35E+02	2.99E+02	+	3.01E+03	3.29E+03	–	3.59E+02	1.79E+02	–	8.68E+01	8.90E+01	–
F22	1.59E+02	8.46E+01	1.42E+02	1.13E+02	=	4.57E+02	1.66E+02	+	2.33E+02	8.37E+01	+	2.76E+01	1.77E+01	–
F23	3.08E+02	2.71E+01	3.15E+02	1.57E-09	–	3.15E+02	1.23E-12	–	3.15E+02	1.83E-08	=	3.15E+02	2.27E-13	–
F24	2.00E+02	1.03E-03	2.10E+02	1.11E+01	+	2.50E+02	6.79E+00	+	2.23E+02	3.68E+00	+	2.24E+02	1.05E+00	+
F25	2.00E+02	0.00E+00	2.03E+02	1.09E-01	+	2.07E+02	3.97E+00	+	2.04E+02	2.90E+00	+	2.03E+02	4.91E-02	+

(continued)

Table 2 (continued)

No.	STSNS		DE			FCDE			PVADE			L-SHADE		
	Mean	Std. Dev	Mean	Std. Dev	W	Mean	Std. Dev	W	Mean	Std. Dev	W	Mean	Std. Dev	W
F26	1.04E+02	1.94E+01	1.00E+02	6.92E-02	+	1.01E+02	1.18E-01	+	1.47E+02	4.96E+01	+	1.00E+02	1.54E-02	–
F27	4.76E+02	1.07E+02	3.93E+02	6.82E+01	–	6.47E+02	2.48E+02	+	4.21E+02	5.27E+01	–	3.00E+02	4.93E-13	–
F28	8.42E+02	4.64E+01	8.09E+02	3.54E+01	–	1.60E+03	5.86E+02	+	9.35E+02	1.11E+02	+	8.40E+02	1.39E+01	=
F29	6.76E+05	2.31E+06	5.10E+05	2.04E+06	+	7.55E+05	2.59E+06	–	4.70E+05	2.33E+06	–	7.17E+02	5.08E+00	–
F30	2.43E+03	1.01E+03	1.56E+03	9.24E+02	+	2.98E+03	1.25E+03	+	1.37E+03	6.79E+02	–	1.25E+03	6.14E+02	–
	+/=/–		21/1/8			20/0/10			13/8/9			2/2/26		

Table 3 Comparing the statistical results of SNS and popular algorithms on 30-D CEC2014 problem suite

No.	STSNS		CMA-ES			FA			CS			TLBO		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F1	4.73E+05	2.68E+05	3.58E+07	1.69E+07	+	3.05E+06	1.10E+06	+	3.29E+06	6.64E+05	+	4.64E+05	4.19E+05	=
F2	5.91E+00	1.03E+01	1.91E+05	1.35E+05	+	1.48E+06	2.09E+05	+	5.03E+02	1.68E+02	+	6.33E+00	4.69E+00	+
F3	1.04E+02	2.28E+02	2.45E+05	7.36E+04	+	2.10E+03	1.13E+03	+	3.83E+00	1.23E+00	−	2.19E+02	2.32E+02	+
F4	5.82E+01	3.24E+01	2.00E+01	6.43E-01	−	7.48E+01	3.71E+01	+	6.24E+01	1.96E+01	=	5.63E+01	4.22E+01	=
F5	2.04E+01	3.52E-02	2.09E+01	4.61E-02	+	2.09E+01	4.79E-02	+	2.08E+01	6.51E-02	+	2.09E+01	5.99E-02	+
F6	4.46E+00	1.93E+00	0.00E+00	0.00E+00	−	8.09E+00	2.84E+00	+	2.56E+01	1.10E+00	+	1.35E+01	2.17E+00	+
F7	1.43E-02	1.55E-02	0.00E+00	0.00E+00	−	9.32E-01	2.68E-02	+	1.78E-03	1.34E-03	=	3.67E-02	7.67E-02	+
F8	7.08E+00	1.17E+00	1.51E+02	1.01E+01	+	3.91E+01	9.75E+00	+	7.85E+01	8.63E+00	+	6.21E+01	1.27E+01	+
F9	6.89E+01	1.47E+01	1.56E+02	8.85E+00	+	5.22E+01	1.45E+01	−	1.47E+02	2.00E+01	+	6.45E+01	1.56E+01	=
F10	7.08E+01	1.30E+01	7.09E+03	2.87E+02	+	1.06E+03	4.94E+02	+	2.04E+03	1.73E+02	+	1.39E+03	5.55E+02	+
F11	2.83E+03	2.77E+02	7.13E+03	3.92E+02	+	3.44E+03	7.05E+02	+	3.52E+03	2.18E+02	+	6.37E+03	5.39E+02	+
F12	5.61E-01	6.52E-02	0.00E+00	0.00E+00	−	1.21E+00	2.17E-01	+	8.77E-01	1.03E-01	+	2.46E+00	3.17E-01	+
F13	2.85E-01	4.67E-02	2.58E-01	4.17E-02	−	3.09E-01	4.86E-02	+	3.13E-01	3.29E-02	+	4.23E-01	7.25E-02	+
F14	2.37E-01	3.54E-02	4.03E-01	5.76E-02	+	3.35E-01	1.18E-01	+	2.33E-01	2.16E-02	=	2.72E-01	1.04E-01	+
F15	7.73E+00	3.33E+00	1.36E+01	8.78E-01	+	1.21E+01	1.21E+00	+	1.09E+01	1.32E+00	+	1.19E+01	5.66E+00	+
F16	1.02E+01	3.15E-01	1.30E+01	3.19E-01	+	1.09E+01	4.62E-01	+	1.24E+01	1.89E-01	+	1.16E+01	4.04E-01	+
F17	3.67E+04	2.81E+04	2.15E+06	1.40E+06	+	3.02E+05	2.35E+05	+	1.35E+04	3.66E+03	−	1.55E+05	1.17E+05	+
F18	1.12E+03	1.54E+03	1.96E+06	1.07E+06	+	3.09E+04	7.58E+03	+	1.63E+02	2.38E+01	−	2.46E+03	2.90E+03	+
F19	1.10E+01	1.62E+01	1.32E+01	1.04E+00	+	1.25E+01	1.57E+01	+	9.40E+00	5.35E-01	+	1.28E+01	1.44E+01	+
F20	3.37E+02	2.18E+02	7.40E+04	4.28E+04	+	1.29E+03	7.62E+02	+	1.29E+02	1.51E+01	−	6.67E+02	2.76E+02	+

(continued)

Table 3 (continued)

No.	STSNS		CMA-ES			FA			CS			TLBO		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F21	1.18E+04	6.93E+03	9.99E+05	6.61E+05	+	1.04E+05	9.81E+04	+	1.73E+03	1.98E+02	−	6.91E+04	5.23E+04	+
F22	1.59E+02	8.46E+01	5.84E+02	1.27E+02	+	3.21E+02	1.52E+02	+	2.72E+02	8.49E+01	+	2.51E+02	1.16E+02	+
F23	3.08E+02	2.71E+01	3.28E+02	5.23E+00	+	3.15E+02	2.63E-03	+	3.15E+02	1.03E-06	+	3.15E+02	4.22E-12	=
F24	2.00E+02	1.03E-03	2.46E+02	2.62E+00	+	2.28E+02	6.40E+00	+	2.31E+02	2.26E+00	+	2.00E+02	1.20E-03	+
F25	2.00E+02	0.00E+00	2.07E+02	2.39E+00	+	2.05E+02	1.35E+00	+	2.08E+02	8.31E-01	+	2.00E+02	8.72E-01	+
F26	1.04E+02	1.94E+01	1.12E+02	4.70E+01	+	1.19E+02	5.25E+01	+	1.00E+02	3.48E-02	+	1.06E+02	2.34E+01	+
F27	4.76E+02	1.07E+02	8.21E+02	4.75E+02	=	6.57E+02	1.34E+02	+	4.13E+02	3.49E+00	+	5.60E+02	1.37E+02	+
F28	8.42E+02	4.64E+01	4.24E+02	2.35E+01	−	1.14E+03	2.42E+02	+	9.87E+02	3.08E+01	+	1.09E+03	1.65E+02	+
F29	6.76E+05	2.31E+06	2.30E+02	5.01E+00	−	1.38E+06	3.85E+06	+	1.85E+03	2.58E+02	+	1.71E+06	3.72E+06	+
F30	2.43E+03	1.01E+03	1.64E+03	2.16E+02	−	2.89E+03	1.09E+03	=	2.65E+03	4.42E+02	=	3.40E+03	4.98E+03	=
	+/=/−		21/1/8			28/1/1			21/4/5			25/5/0		
No.	STSNS		CSA			WCA			GWO			WOA		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F1	4.73E+05	2.68E+05	1.69E+06	8.78E+05	+	5.40E+05	5.32E+05	=	3.84E+07	2.51E+07	+	3.23E+07	1.31E+07	+
F2	5.91E+00	1.03E+01	1.06E+04	7.40E+03	+	4.29E+03	4.77E+03	+	1.04E+09	1.17E+09	+	3.39E+06	2.01E+06	+
F3	1.04E+02	2.28E+02	1.79E+03	8.63E+02	+	1.60E+01	1.99E+01	−	2.36E+04	8.66E+03	+	3.73E+04	2.86E+04	+
F4	5.82E+01	3.24E+01	1.05E+02	4.92E+01	+	3.30E+01	4.13E+01	−	2.11E+02	4.66E+01	+	2.00E+02	6.89E+01	+
F5	2.04E+01	3.52E-02	2.00E+01	5.42E-04	−	2.01E+01	1.33E-01	−	2.06E+01	3.45E-01	=	2.03E+01	1.67E-01	−
F6	4.46E+00	1.93E+00	2.55E+01	3.12E+00	+	2.74E+01	3.96E+00	+	1.26E+01	3.06E+00	+	3.65E+01	3.97E+00	+

(continued)

Table 3 (continued)

No.	STSNS		CSA			WCA			GWO			WOA		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F7	1.43E-02	1.55E-02	1.22E-02	2.01E-02	=	1.40E-02	1.79E-02	=	1.41E+01	1.23E+01	+	1.02E+00	3.94E-02	+
F8	7.08E+00	1.17E+00	1.19E+02	2.00E+01	+	1.05E+02	3.34E+01	+	8.26E+01	2.08E+01	+	1.80E+02	3.82E+01	+
F9	6.89E+01	1.47E+01	1.18E+02	2.08E+01	+	1.81E+02	3.91E+01	+	9.39E+01	2.11E+01	+	2.51E+02	7.46E+01	+
F10	7.08E+01	1.30E+01	3.20E+03	6.27E+02	+	2.16E+03	6.52E+02	+	2.20E+03	4.47E+02	+	3.94E+03	8.00E+02	+
F11	2.83E+03	2.77E+02	3.54E+03	5.93E+02	+	4.04E+03	6.48E+02	+	2.92E+03	5.87E+02	=	5.01E+03	8.55E+02	+
F12	5.61E-01	6.52E-02	5.28E-01	2.13E-01	−	7.80E-01	3.63E-01	+	1.71E-01	3.20E-01	−	1.68E+00	5.03E-01	+
F13	2.85E-01	4.67E-02	4.61E-01	1.11E-01	+	5.25E-01	1.28E-01	+	3.57E-01	6.76E-02	+	5.32E-01	1.20E-01	+
F14	2.37E-01	3.54E-02	2.78E-01	4.03E-02	+	4.50E-01	2.41E-01	+	2.08E+00	3.76E+00	+	2.74E-01	5.17E-02	+
F15	7.73E+00	3.33E+00	1.27E+01	4.20E+00	+	2.29E+01	8.60E+00	+	5.03E+01	1.92E+02	+	7.07E+01	2.30E+01	+
F16	1.02E+01	3.15E-01	1.18E+01	4.84E-01	+	1.25E+01	3.81E-01	+	1.09E+01	6.94E-01	+	1.26E+01	5.15E-01	+
F17	3.67E+04	2.81E+04	1.90E+04	1.82E+04	−	5.50E+04	9.25E+04	=	9.33E+05	7.69E+05	+	3.64E+06	2.58E+06	+
F18	1.12E+03	1.54E+03	4.17E+02	3.53E+02	=	5.88E+03	7.44E+03	+	4.48E+06	1.34E+07	+	2.66E+04	7.68E+04	+
F19	1.10E+01	1.62E+01	2.28E+01	1.79E+01	+	1.64E+01	1.62E+01	+	2.94E+01	1.93E+01	+	4.94E+01	3.68E+01	+
F20	3.37E+02	2.18E+02	4.03E+02	1.49E+02	+	3.84E+02	1.13E+02	+	1.17E+04	6.71E+03	+	2.47E+04	2.09E+04	+
F21	1.18E+04	6.93E+03	1.01E+04	5.38E+03	=	2.39E+04	2.59E+04	+	4.12E+05	3.99E+05	+	1.11E+06	1.54E+06	+
F22	1.59E+02	8.46E+01	5.04E+02	1.43E+02	+	5.31E+02	2.09E+02	+	3.39E+02	1.51E+02	+	7.86E+02	2.46E+02	+
F23	3.08E+02	2.71E+01	3.16E+02	4.72E-01	+	3.15E+02	7.14E-03	+	3.31E+02	7.96E+00	+	3.34E+02	7.53E+00	+
F24	2.00E+02	1.03E-03	2.11E+02	1.12E+01	+	2.30E+02	4.50E+00	+	2.00E+02	4.91E-04	−	2.06E+02	5.16E+00	+
F25	2.00E+02	0.00E+00	2.03E+02	3.75E+00	+	2.18E+02	6.27E+00	+	2.11E+02	2.07E+00	+	2.18E+02	1.54E+01	+

(continued)

Table 3 (continued)

No.	STSNS		CSA			WCA			GWO			WOA		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F26	1.04E+02	1.94E+01	1.00E+02	1.13E-01	+	1.12E+02	3.20E+01	+	1.20E+02	3.95E+01	+	1.00E+02	1.43E-01	+
F27	4.76E+02	1.07E+02	4.81E+02	1.95E+02	=	8.92E+02	2.74E+02	+	6.08E+02	1.24E+02	+	1.07E+03	3.61E+02	+
F28	8.42E+02	4.64E+01	3.25E+03	6.99E+02	+	1.62E+03	4.35E+02	+	1.04E+03	2.02E+02	+	2.14E+03	6.16E+02	+
F29	6.76E+05	2.31E+06	1.83E+06	1.29E+07	+	4.19E+06	6.12E+06	+	4.57E+05	1.90E+06	+	4.91E+06	4.63E+06	+
F30	2.43E+03	1.01E+03	7.29E+03	3.29E+03	+	4.24E+03	1.80E+03	+	3.92E+04	2.36E+04	+	9.50E+04	7.07E+04	+
	+/=/−		23/4/3			24/3/3			26/2/2			29/0/1		

Table 4 Comparing the statistical results of SNS and newly developed metaheuristic algorithms on 30-D CEC2014 problem suite

No.	STSNS		SNS			SPO			CryStAl			AOS		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F1	4.73E+05	2.68E+05	3.14E+05	2.03E+05	–	2.26E+06	3.50E+06	+	9.57E+06	3.40E+06	+	1.72E+07	5.16E+06	+
F2	5.91E+00	1.03E+01	5.00E+01	4.40E+01	+	4.84E+03	5.25E+03	+	6.17E+08	1.73E+08	+	2.30E+06	8.72E+05	+
F3	1.04E+02	2.28E+02	4.02E+01	8.73E+01	=	2.14E+03	2.26E+03	+	2.91E+04	8.04E+03	+	1.19E+04	3.13E+03	+
F4	5.82E+01	3.24E+01	7.74E+01	2.93E+01	+	5.53E+01	3.57E+01	=	2.97E+02	5.44E+01	+	1.40E+02	3.31E+01	+
F5	2.04E+01	3.52E-02	2.04E+01	3.88E-02	=	2.10E+01	6.75E-02	+	2.09E+01	5.18E-02	+	2.09E+01	6.81E-02	+
F6	4.46E+00	1.93E+00	1.11E+01	2.65E+00	+	9.65E+00	3.56E+00	+	2.79E+01	1.98E+00	+	2.58E+01	4.03E+00	+
F7	1.43E-02	1.55E-02	1.38E-02	1.67E-02	=	4.10E-03	7.97E-03	–	8.52E+00	2.61E+00	+	1.02E+00	2.57E-02	+
F8	7.08E+00	1.17E+00	7.64E+00	1.05E+00	+	4.80E+01	3.33E+01	+	1.72E+02	1.63E+01	+	1.14E+02	2.66E+01	+
F9	6.89E+01	1.47E+01	6.76E+01	1.46E+01	=	1.22E+02	8.92E+01	=	1.77E+02	1.29E+01	+	1.39E+02	2.46E+01	+
F10	7.08E+01	1.30E+01	6.62E+01	1.22E+01	–	3.46E+03	1.43E+03	+	5.43E+03	3.63E+02	+	3.42E+03	5.29E+02	+
F11	2.83E+03	2.77E+02	2.77E+03	2.42E+02	=	5.63E+03	1.82E+03	+	5.76E+03	3.70E+02	+	3.52E+03	6.01E+02	+
F12	5.61E-01	6.52E-02	5.64E-01	6.83E-02	=	2.80E+00	5.74E-01	+	2.07E+00	2.59E-01	+	1.45E+00	6.48E-01	+
F13	2.85E-01	4.67E-02	2.95E-01	4.97E-02	=	5.99E-01	1.20E-01	+	5.11E-01	8.55E-02	+	6.08E-01	1.33E-01	+
F14	2.37E-01	3.54E-02	2.31E-01	3.70E-02	=	5.84E-01	2.38E-01	+	6.21E-01	1.26E+00	+	4.18E-01	2.62E-01	+
F15	7.73E+00	3.33E+00	8.35E+00	2.43E+00	=	1.55E+01	6.81E+00	+	3.20E+01	7.01E+00	+	2.13E+01	4.42E+00	+
F16	1.02E+01	3.15E-01	1.02E+01	3.15E-01	=	1.30E+01	3.56E-01	+	1.26E+01	2.30E-01	+	1.17E+01	6.54E-01	+
F17	3.67E+04	2.81E+04	2.61E+04	1.44E+04	=	1.50E+05	3.38E+05	+	2.84E+05	1.18E+05	+	8.55E+05	5.07E+05	+
F18	1.12E+03	1.54E+03	6.13E+02	6.80E+02	=	1.60E+04	9.03E+03	+	1.80E+06	7.94E+05	+	1.04E+04	7.16E+03	+
F19	1.10E+01	1.62E+01	1.07E+01	1.60E+01	=	5.95E+00	1.46E+00	=	2.31E+01	4.90E+00	+	3.68E+01	3.26E+01	+
F20	3.37E+02	2.18E+02	3.96E+02	3.31E+02	=	1.59E+04	1.39E+04	+	1.80E+04	1.11E+04	+	1.31E+04	7.47E+03	+

(continued)

Table 4 (continued)

No.	STSNS		SNS			SPO			CryStAl			AOS		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F21	1.18E+04	6.93E+03	1.64E+04	1.09E+04	+	6.44E+04	1.24E+05	+	1.46E+05	6.56E+04	+	3.24E+05	2.59E+05	+
F22	1.59E+02	8.46E+01	2.17E+02	1.09E+02	+	4.42E+02	2.32E+02	+	3.78E+02	1.10E+02	+	5.17E+02	1.89E+02	+
F23	3.08E+02	2.71E+01	3.15E+02	1.55E-11	=	3.15E+02	3.44E-13	−	2.00E+02	1.43E-04	−	3.19E+02	1.82E+00	+
F24	2.00E+02	1.03E-03	2.00E+02	1.27E-04	−	2.24E+02	8.40E+00	+	2.00E+02	6.06E-06	−	2.00E+02	7.37E-05	−
F25	2.00E+02	0.00E+00	2.00E+02	0.00E+00	=	2.06E+02	4.00E+00	+	2.00E+02	3.20E-07	+	2.02E+02	5.08E+00	+
F26	1.04E+02	1.94E+01	1.20E+02	3.96E+01	+	1.01E+02	4.01E-01	+	1.00E+02	7.38E-02	+	1.01E+02	1.51E-01	+
F27	4.76E+02	1.07E+02	5.07E+02	1.16E+02	=	6.22E+02	1.93E+02	+	2.06E+02	3.85E+01	−	6.33E+02	2.76E+02	+
F28	8.42E+02	4.64E+01	9.12E+02	1.02E+02	+	1.13E+03	2.05E+02	+	2.00E+02	3.16E-03	−	2.22E+03	5.02E+02	+
F29	6.76E+05	2.31E+06	6.76E+05	2.31E+06	=	5.62E+06	5.68E+06	+	7.41E+05	2.54E+06	−	1.11E+07	1.87E+07	+
F30	2.43E+03	1.01E+03	2.48E+03	7.92E+02	=	5.52E+04	9.10E+04	+	1.14E+05	1.11E+05	+	6.52E+04	7.68E+04	+
	+/=/−		8/19/3			25/3/2			25/0/5			29/0/1		
No.	STSNS		CGO			SOS			HHO			HGS		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F1	4.73E+05	2.68E+05	5.88E+04	1.08E+05	−	1.05E+06	1.02E+06	+	3.74E+07	1.84E+07	+	1.99E+06	1.28E+06	+
F2	5.91E+00	1.03E+01	1.60E-07	8.66E-07	−	4.31E+00	4.62E+00	=	3.86E+07	1.12E+07	+	1.50E+04	1.40E+04	+
F3	1.04E+02	2.28E+02	4.12E-02	2.03E-01	−	9.74E+01	7.97E+01	+	2.23E+04	7.88E+03	+	7.59E+03	5.80E+03	+
F4	5.82E+01	3.24E+01	6.65E+00	1.91E+01	−	5.23E+01	4.87E+01	=	2.29E+02	7.72E+01	+	9.95E+01	2.86E+01	+
F5	2.04E+01	3.52E-02	2.08E+01	2.55E-01	+	2.05E+01	7.47E-02	+	2.03E+01	1.70E-01	−	2.01E+01	5.24E-02	−
F6	4.46E+00	1.93E+00	1.90E+01	3.44E+00	+	9.14E+00	2.43E+00	+	3.37E+01	3.74E+00	+	1.69E+01	2.82E+00	+

(continued)

Table 4 (continued)

No.	STSNS		CGO			SOS			HHO			HGS		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F7	1.43E-02	1.55E-02	2.55E-02	5.34E-02	=	1.27E-02	1.32E-02	=	1.32E+00	1.04E-01	+	3.39E-02	3.93E-02	+
F8	7.08E+00	1.17E+00	6.35E+01	1.65E+01	+	4.73E+01	1.04E+01	+	1.38E+02	1.60E+01	+	3.76E+00	3.12E+00	−
F9	6.89E+01	1.47E+01	7.10E+01	1.56E+01	=	5.20E+01	1.87E+01	−	1.87E+02	2.45E+01	+	1.25E+02	2.29E+01	+
F10	7.08E+01	1.30E+01	2.45E+03	7.09E+02	+	1.02E+03	2.16E+02	+	3.24E+03	7.57E+02	+	1.69E+02	1.86E+02	+
F11	2.83E+03	2.77E+02	4.51E+03	1.81E+03	+	2.58E+03	9.54E+02	=	4.68E+03	7.37E+02	+	2.54E+03	5.67E+02	−
F12	5.61E-01	6.52E-02	1.91E+00	8.29E-01	+	5.09E-01	1.34E-01	−	1.87E+00	4.27E-01	+	1.47E-01	5.52E-02	−
F13	2.85E-01	4.67E-02	4.04E-01	8.92E-02	+	3.90E-01	8.36E-02	+	5.15E-01	1.13E-01	+	6.90E-01	1.09E-01	+
F14	2.37E-01	3.54E-02	4.04E-01	1.75E-01	+	3.30E-01	1.35E-01	+	2.72E-01	4.90E-02	+	9.18E-01	2.44E-01	+
F15	7.73E+00	3.33E+00	9.95E+00	4.29E+00	+	1.56E+01	2.87E+00	+	5.41E+01	1.23E+01	+	9.82E+00	2.81E+00	+
F16	1.02E+01	3.15E-01	1.10E+01	8.23E-01	+	1.04E+01	5.84E-01	=	1.23E+01	4.23E-01	+	1.11E+01	6.71E-01	+
F17	3.67E+04	2.81E+04	4.37E+03	2.53E+03	−	1.18E+05	1.37E+05	+	5.57E+06	4.00E+06	+	3.61E+05	2.03E+05	+
F18	1.12E+03	1.54E+03	4.73E+03	6.53E+03	+	5.19E+03	5.56E+03	+	2.15E+05	1.61E+05	+	1.07E+04	8.48E+03	+
F19	1.10E+01	1.62E+01	1.35E+01	1.35E+01	+	1.13E+01	1.38E+01	+	4.71E+01	3.52E+01	+	1.62E+01	1.85E+01	+
F20	3.37E+02	2.18E+02	2.19E+02	8.36E+01	−	1.21E+03	1.03E+03	+	2.85E+04	1.41E+04	+	3.78E+03	3.11E+03	+
F21	1.18E+04	6.93E+03	1.91E+03	1.36E+03	−	4.98E+04	9.09E+04	+	8.15E+05	5.90E+05	+	1.89E+05	1.64E+05	+
F22	1.59E+02	8.46E+01	3.81E+02	1.85E+02	+	2.93E+02	1.33E+02	+	8.64E+02	2.37E+02	+	5.63E+02	2.06E+02	+
F23	3.08E+02	2.71E+01	2.00E+02	0.00E+00	−	3.15E+02	6.55E-13	−	2.00E+02	0.00E+00	−	2.00E+02	0.00E+00	−
F24	2.00E+02	1.03E-03	2.00E+02	0.00E+00	−	2.00E+02	1.12E-03	−	2.00E+02	1.57E-04	−	2.00E+02	2.79E-04	−
F25	2.00E+02	0.00E+00	2.00E+02	0.00E+00	=	2.00E+02	0.00E+00	=	2.00E+02	0.00E+00	=	2.00E+02	0.00E+00	=

(continued)

Table 4 (continued)

No.	STSNS		CGO			SOS			HHO			HGS		
	Mean	Std. Dev.	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W	Mean	Std. Dev.	W
F26	1.04E+02	1.94E+01	1.00E+02	1.04E-01	+	1.00E+02	7.78E-02	+	1.47E+02	4.97E+01	+	1.14E+02	3.42E+01	+
F27	4.76E+02	1.07E+02	2.00E+02	0.00E+00	–	5.25E+02	1.34E+02	=	2.00E+02	0.00E+00	–	2.00E+02	0.00E+00	–
F28	8.42E+02	4.64E+01	2.00E+02	0.00E+00	–	9.91E+02	2.06E+02	+	2.00E+02	0.00E+00	–	2.00E+02	0.00E+00	–
F29	6.76E+05	2.31E+06	4.35E+02	3.90E+02	–	1.51E+06	3.26E+06	=	7.72E+02	3.10E+03	–	1.39E+03	2.80E+03	–
F30	2.43E+03	1.01E+03	2.48E+02	2.36E+02	–	2.50E+03	1.05E+03	=	2.68E+04	3.66E+04	=	4.29E+02	7.95E+02	–
	+/=/–		14/3/13			17/9/4			22/2/6			19/1/10		

test, the SNS algorithm outperforms CMA-ES, FA, CS, TLBO, CSA, WCA, GWO, and WOA algorithms on 21, 28, 21, 25, 23, 24, 26, and 29 cases out of 30 problems, respectively. The proposed SNS performs weaker or equal compared to contenders from popular algorithms on a few problems.

The third group of contender algorithms contains some newly developed methods. Table 4 presents the statistical results of these algorithms alongside the STSNS algorithm. The results of the WSR test shows that STSNS can outperform SNS, SPO, CryStAl, AOS, SSA, SOS, HHO, and HGS algorithms in dealing with eight, 25, 25, 29, 14, 17, 22, and 19 of 30 cases out of selected CEC 2014 benchmark problems, respectively. Besides, the STSNS has the same performance on 19, three, three, nine, two, and one cases of problems, comparing with SNS, SPO, CGO, SOS, HHO, and HGS algorithms, respectively.

In these experiments, 24 algorithms are utilized as contenders to compare the performance of the STSNS algorithm. The employed algorithms contain a varied range of valid methods in the literature, and in most of these experiments, the STSNS achieved better performance. To compare the ability of the STSNS algorithm with all algorithms, the Friedman test is employed, and the ranking results are presented in Table 5. According to these results, the STSNS placed in the second rank, after L-SHADE algorithm. This ranking shows that the STSNS algorithm can performs well in solving intricate problems.

Table 5 The results of Friedman test in ranking all of algorithms

No.	Algorithm	FR	Rank	No.	Algorithm	FR	Rank
1	**STSNS**	**7.45**	**2**	14	CSA	13.83	15
2	PSO	14.70	17	15	WCA	16.03	18
3	DMSPSO	11.30	9	16	GWO	17.57	22
4	CLPSO	11.67	10	17	WOA	21.13	25
5	LIPS	11.80	12	**18**	**SNS**	**8.35**	**3**
6	DE	10.50	7	19	SPO	16.27	19
7	FCDE	13.62	14	20	CryStAl	17.90	23
8	PVADE	8.58	4	21	AOS	18.90	24
9	**L-SHADE**	**2.92**	**1**	22	CGO	8.78	5
10	CMA-ES	16.65	20	23	SOS	9.68	6
11	FA	14.57	16	24	HHO	17.25	21
12	CS	10.90	8	25	HGS	11.78	11
13	TLBO	12.87	13				

5 Conclusion

The social network search (SNS) is a novel optimization method that introduce four new moods based on the behavior of users in sharing their views on the social networks. In this chapter, the concept of space–time is incorporated into the structure of the SNS algorithm, and STSNS method is introduced. The developed STSNS is employed for solving 30 complex benchmark problems of CEC 2014 and its performance is investigated in three experiments. In the first experiment, the STSNS is compared with eight state-of-the-art metaheuristic algorithm from the literature that placed in the second rank. In the second comparison, the STSNS compared with eight popular algorithms, and results of the WSR test demonstrate that the STSNS algorithm outperforms all the contender methods in most of the problems. The third experiment shows that the STSNS achieves the first rank in comparison with newly developed algorithms. Also, this experiment shows that there is a significant difference between the capability of the STSNS algorithm and newly developed contender methods. Finally, the Friedman test is employed for ranking all of the algorithms in solving selected problems, and results show that the STSNS and SNS are placed in the second and third ranks, respectively. Therefore, it can be concluded that both SNS and STSNS algorithms can solve various optimization problems, and they have good potential to be modified or redesigned for achieving better capability in solving optimization problems. In future studies, the performance of SNS and STSNS algorithms should be evaluated in solving other types of complex real-world optimization problems.

References

1. Talatahari, S., Bayzidi, H., Saraee, M.: Social network search for global optimization. IEEE Access **9**, 92815–92863 (2021). https://doi.org/10.1109/ACCESS.2021.3091495
2. Talatahari S., Azizi M.: Chaos game optimization: a novel metaheuristic algorithm. Artif. Intell. Rev. **54**(2), 917–1004 (2020). https://doi.org/10.1007/S10462-020-09867-W
3. Rechenberg, I.: Evolutionsstrategien, pp. 83–114. Springer, Berlin, Heidelberg (1978)
4. Holland J.H.: Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence (1975)
5. Glover, F.: Tabu search—part I. ORSA J. Comput. **1**, 190–206 (1989). https://doi.org/10.1287/ijoc.1.3.190
6. Glover, F.: Tabu search—part II. ORSA J. Comput. **2**, 4–32 (1990). https://doi.org/10.1287/ijoc.2.1.4
7. Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization artificial ants as a computational intelligence technique. IEEE Comput. Intell. Mag. **1**, 28–39 (2006). https://doi.org/10.1109/CI-M.2006.248054
8. Kennedy J., Eberhart R.: Particle swarm optimization. In: Proceedings of ICNN'95—International Conference on Neural Networks. IEEE, pp. 1942–1948 (1995)
9. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997). https://doi.org/10.1023/A:1008202821328

10. Karaboga D., Basturk B.: Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 789–798 (2007)
11. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimization. Int. J. Bio-Inspired Comput. **2**, 78–84 (2010). https://doi.org/10.1504/IJBIC.2010.032124
12. Yang, X.-S., Deb, S.: Engineering optimisation by cuckoo search. Int. J. Math. Model. Numer. Optim. **1**, 330–343 (2010)
13. Kaveh, A., Talatahari, S.: A novel heuristic optimization method: charged system search. Acta Mech. **213**, 267–289 (2010). https://doi.org/10.1007/s00707-009-0270-4
14. Rao, R.V., Savsani, V.J., Vakharia, D.P.: Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. CAD Comput. Aided Des. **43**, 303–315 (2011). https://doi.org/10.1016/j.cad.2010.12.015
15. Askarzadeh, A.: A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput. Struct. **169**, 1–12 (2016). https://doi.org/10.1016/j.compstruc.2016.03.001
16. Eskandar, H., Sadollah, A., Bahreininejad, A., Hamdi, M.: Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput. Struct. **110–111**, 151–166 (2012). https://doi.org/10.1016/j.compstruc.2012.07.010
17. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007
18. Cheng, M.Y., Prayogo, D.: Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput. Struct. **139**, 98–112 (2014). https://doi.org/10.1016/j.compstruc.2014.03.007
19. Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., et al.: Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. **114**, 163–191 (2017). https://doi.org/10.1016/j.advengsoft.2017.07.002
20. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016). https://doi.org/10.1016/j.advengsoft.2016.01.008
21. Heidari, A.A., Mirjalili, S., Faris, H., et al.: Harris hawks optimization: algorithm and applications. Futur. Gener. Comput. Syst. **97**, 849–872 (2019). https://doi.org/10.1016/j.future.2019.02.028
22. Kaveh, A., Talatahari, S., Khodadadi, N.: Stochastic paint optimizer: theory and application in civil engineering. Eng. Comput. **2020**, 1–32 (2020). https://doi.org/10.1007/S00366-020-01179-5
23. Talatahari, S., Azizi, M.: Optimization of constrained mathematical and engineering design problems using chaos game optimization. Comput. Ind. Eng. **145**, 106560 (2020). https://doi.org/10.1016/J.CIE.2020.106560
24. Azizi, M.: Atomic orbital search: a novel metaheuristic algorithm. Appl. Math. Model. **93**, 657–683 (2021). https://doi.org/10.1016/J.APM.2020.12.021
25. Azizi, M., Talatahari, S., Giaralis, A.: Optimization of engineering design problems using atomic orbital search algorithm. IEEE Access **9**, 102497–102519 (2021). https://doi.org/10.1109/ACCESS.2021.3096726
26. Talatahari, S., Azizi, M., Tolouei, M., et al.: Crystal structure algorithm (CryStAl): a metaheuristic optimization method. IEEE Access **9**, 71244–71261 (2021). https://doi.org/10.1109/ACCESS.2021.3079161
27. Talatahari, S., Azizi, M., Gandomi, A.H.: Material generation algorithm: a novel metaheuristic algorithm for optimization of engineering problems. Processes **9**, 859 (2021). https://doi.org/10.3390/PR9050859
28. Bayzidi, H., Talatahari, S., Saraee, M., Lamarche, C.-P.: Social network search for solving engineering optimization problems. Comput. Intell. Neurosci. **2021**, 1–32 (2021). https://doi.org/10.1155/2021/8548639
29. Liang J.J., Suganthan P.N., Qu B.Y.: Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization, Singapore (2013)

30. Hadi B.: Social Network Search for solving engineering problems. MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/97577-social-network-search-for-solving-engineering-problems (2021)
31. Talatahari S.: Social network search for global optimization. MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/94370-social-network-search-for-global-optimization (2021)
32. Kaveh A., Talatahari S.: An enhanced charged system search for configuration optimization using the concept of fields of forces. Struct. Multidiscip. Optim. **43**(3), 339–351 (2010). https://doi.org/10.1007/S00158-010-0571-1
33. Liang J.J., Suganthan P.N.: Dynamic multi-swarm particle swarm optimizer. In: Proceedings—2005 IEEE Swarm Intelligence Symposium, SIS 2005, pp. 127–132 (2005). https://doi.org/10.1109/SIS.2005.1501611
34. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. **10**, 281–295 (2006). https://doi.org/10.1109/TEVC.2005.857610
35. Qu, B.Y., Suganthan, P.N., Das, S.: A distance-based locally informed particle swarm model for multimodal optimization. IEEE Trans. Evol. Comput. **17**, 387–402 (2013). https://doi.org/10.1109/TEVC.2012.2203138
36. Li Z., Shang Z., Qu B.Y., Liang J.J.: Differential Evolution strategy based on the constraint of fitness values classification. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, pp. 1454–1460 (2014). https://doi.org/10.1109/CEC.2014.6900507
37. Tanabe R., Fukunaga A.S.: Improving the search performance of SHADE using linear population size reduction. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, pp. 1658–1665 (2014). https://doi.org/10.1109/CEC.2014.6900380
38. Dos Santos Coelho L., Ayala H.V.H., Freire R.Z.: Population's variance-based adaptive differential evolution for real parameter optimization. In: 2013 IEEE Congress on Evolutionary Computation, CEC 2013, pp. 1672–1677 (2013). https://doi.org/10.1109/CEC.2013.6557762
39. Hansen N.: The CMA evolution strategy: a comparing review. In: Towards a New Evolutionary Computation, pp. 75–102 (2006). https://doi.org/10.1007/3-540-32494-1_4
40. Hashim, F.A., Houssein, E.H., Mabrouk, M.S., et al.: Henry gas solubility optimization: a novel physics-based algorithm. Futur. Gener. Comput. Syst. **101**, 646–667 (2019). https://doi.org/10.1016/j.future.2019.07.015

Genetic Algorithm Golden Ratio Design Model for Auto Arts

Khaled Hossameldin Sadek Ibrahim, Ali Khater Mohamed, and Ahmed Farouk

Abstract Golden ratio denoted by the Greek letter *Phi* (φ) represents the irrational number 1.6180339887 approximately. The unique and mystifying properties of this number have attracted many researchers and mathematicians. Artists and designers tried to employ the golden ratio in eminent works of drawing, artifacts, painting, and architectures as golden ratio design is very pleasing to look at because the golden ratio itself is discovered from nature through the body proportions of living beings. Nowadays every big organization or company converts its design or logo to a golden ratio design and hires the biggest designers with the highest cost to make it as only the highest professional designers can make it and look good. This paper proposes a framework for generating a golden ratio design using golden circles through a sketched design from the user so that the output design is near to the input design. Genetic algorithm is used to optimize the output design. i.e., maximize the similarity between the input sketched design by the user and the output design using golden ratio in less computational time. The data set presented in this work is collected by the authors and could be downloaded from the following link: https://drive.google.com/file/d/1IEKIs0f1mLxSf7hPZhVKRU0Aiv4VXGax/view?usp=sharing.

Keywords Golden ratio · Genetic algorithm · Golden ratio design

K. H. S. Ibrahim · A. K. Mohamed (✉) · A. Farouk
Faculty of Computer Science, October University for Modern Sciences and Arts (MSA), Giza, Egypt
e-mail: akhter@msa.edu.eg

K. H. S. Ibrahim
e-mail: khaled.hossameldin@msa.edu.eg

A. Farouk
e-mail: ahmed.farouk@msa.edu.eg

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_8

1 Introduction

The main idea about making a design is to make it good and pleasing to look at. There are many ways and rules to make a good design. Almost all of them are made with a combination of lines, circles, rectangles, and any primitive shapes. One way of making a good design is using a golden ratio as it will be called a Golden Ratio Designs. Golden ratio can be used in many areas like typography, layout, photography, and logo design. It is also considered as the hardest one to achieve among designers. This research aims to save some time for the designer to come up with more ideas about a design. If the designer has the data about the golden circles needed to make this design, he will insert these circles' data to the Genetic Algorithm model, and it will do the rest. It will know how to divide circles to regions based on the intersections between the circles. It will also know which region will be drawn and which will be empty. Golden circles are circles made with the golden ratio as the ratio between every circle and the circle after it or before it is golden ratio ($\varphi = 1.618$) (Fig. 1).

It is prohibited to use any circle size other than these circles with the same ratio. If a resize is needed, then all circles must be resized with same ratio to maintain the golden ratio in the design. Normally, designers come up with ideas about a certain logo then try to come up with golden circles to make a golden ratio design. Then try to arrange every circle and delete the unneeded intersections based on the desired design using any vector-based design software like Adobe Illustrator. This way consumes a lot of time for the designer and a lot of money for the company as a golden ratio design is very expensive.

The next section, Sect. 2, introduces previous work. Section 3 represents meta-heuristic and genetic algorithms. Materials and methods are presented in Sect. 4. Results and evaluation are clarified in Sect. 5, and Sect. 6 concludes the presented work.

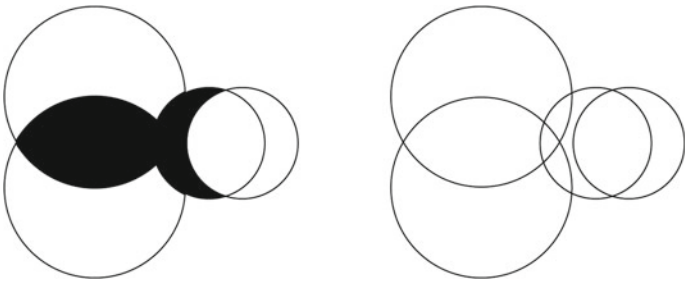


Fig. 1 Fish design

2 Previous Work

This section reviews some researches that was inspired by golden ratio and primitive shapes. [1] presented how to convert the input image into many small colored primitive shapes. The main goal was to resize the image and change its aspect ratio to whatever the user wants without losing the details or the quality of the image. It uses GAN to decide the size, location, and color of every primitive shape to recreate the inserted image again with these predicted primitive shapes.

Taking a sketch as an input and uses Image Processing and Image Analysis to recognize and improve the inserted sketch with as low error as possible was presented in [2].

Other researchers take a sketch as an input and also uses Image Processing and Image Analysis [3] but the detection in the previous research was better than the recognition of this research. However, the improvement is better in this research than the previous research.

Taking a sketch as an input and using [2, 3] to get the best results possible in recognizing and improving the inserted sketch is presented in [4]. It adds to these systems to calculated graphs Normalized Distance between Direction Extremes (NDDE) and Direction Change Ratio (DCR). These graphs improve the recognition and improvement even more as they are used in the two processes to get the best results possible.

3 Metaheuristic and Genetic Algorithms

3.1 Metaheuristic

During the last three decades, metaheuristic algorithms has gained a great popularity in solving real-life complex problems in many different fields like engineering, economics, management and politics [5].

The balance between exploration and exploitation is considered as the most important aspect for any metaheuristic algorithm as it is required to solve complex real-life problems effectively.

Metaheuristic algorithms are inspired from evolutionary process, swarm intelligence, physics, and human [6] as illustrated in Fig. 2.

Metaheuristic could be classified also into two categories:

1. Single point metaheuristic: in which, the algorithm uses only one single individual solution, and the local search is used for improving this solution (Simulated Annealing, Tabu Search, Microcanonical Annealing, and Guided Local Search) [7]. However, using a single population algorithm may stuck in local optimum.

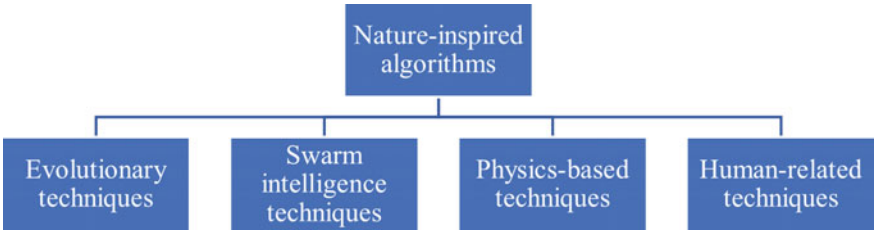


Fig. 2 Classification of nature inspired algorithms

2. Population-based metaheuristic: which uses multiple individual solutions during their search process. This kind of algorithms maintain the diversity in the population in order to avoid the premature or stagnation (Genetic algorithms, Particle Swarm, Ant Colony, Differential Evolution, and Gaining–Sharing Knowledge algorithms.

3.2 Genetic Algorithms (GA)

Genetic Algorithms is a population-based optimization algorithm which is inspired from biological evolution process [8]. GA mimics the Darwinian theory of survival of fittest in nature. So, it utilizes the concept of survival of fittest. Each individual solution in the population is called a chromosome, each chromosome consists of group of loci, each locus represents a specific position in the chromosome, and each locus has only one of two possible values (0 or 1) as depicted in Fig. 3.

The GA starts with a randomly initialized population of size n chromosomes (individual solutions), and the fitness value is evaluated for each chromosome. The new population is obtained by going through an iterative process using GA operators, which are: selection, crossover, and mutation.

- Selection phase is the phase in which two individuals are selected from the population based on their fitness value to pass their genes for the next generation.
- Crossover phase is considered as the most significant phase in GA as it affects the convergence speed of the algorithm. In this phase a random point is chosen as the crossover point and the offspring's are created through exchanging parent's genes till reaching the selected crossover point.
- The last phase is the mutation phase. In which a flip in a random gene is made with low probability in order to maintain the diversity of the population and maintaining the exploration of the search space. This is done to prevent the stagnation and/or the premature convergence.

Fig. 3 Chromosome representation in GA

1.	Read: Population size (n), Maximum number of iterations (MAX)
2.	Begin
3.	Generate an initial random population of size n
4.	Counter =0
5.	While (Counter <MAX)
6.	Select two individuals from the population according to their fitness value
7.	Apply cross over operation with crossover probability
8.	Apply mutation with mutation probability for the offspring
9.	Replace the old population with the new population
10.	Counter = Counter+1
11.	End while
12.	End GA

Fig. 4 Pseudocode of the Classical GA

A pseudocode of the Classical Genetic Algorithm is represented in Fig. 4.

4 Material & Methods

4.1 Materials

The input will be an image which is the desired golden ratio design and a list of circles which will be drawn. These images were found on the internet then remade to get the circles' data. This data is saved in an excel file with four columns **[Design, Center (X), Center (Y), Radius]** as shown in the table below but on a bigger scale and many designs. This process was made with Adobe Illustrator in approximately one month. This data set is made of 100 images which are golden ratio designs made by many designers then published on the internet on many different websites. All of them will have an image format as PNG without backgrounds. All images will have the same size which is the biggest size among all images as well as the same resolution as 72 ppi. Data needs to be prepossessed as it only has the final design without the specific data needed.

4.2 Methods

Data Preprocessing

Firstly, the image is just drawn on the screen without any changing. Secondly, it will create a list of circles using the circles' data inserted previously which should be golden circles. Thirdly, it will convert these circles into regions using Flood fill

algorithm based on the intersections between the circles then it saves these regions in a list as shown in Fig. 5.

In the beginning, all regions were taken into consideration even if it is entirely outside the image or it was on a non-colored area. It was improved afterwards as if the whole region is in a non-colored area or outside the image will not be taken into consideration. However, this method does not work on all designs as sometimes some colored pixels are inside the region as circles will not be exactly aligned with the image.

In Fig. 6 the circles used to create the turtle are shown on the image itself. These circles were created using the data mentioned in Table 1. It may seem like the circles are aligned with image exactly, but it is not. Some of the turtle's—colored pixels are outside the circle. These pixels affect the Genetic Algorithm as it works pixel wise

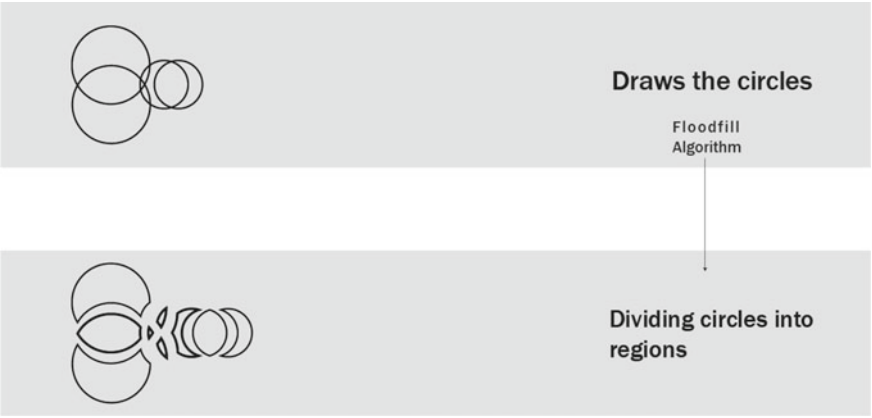


Fig. 5 Data preprocessing architecture

Fig. 6 Circles shown on the turtle

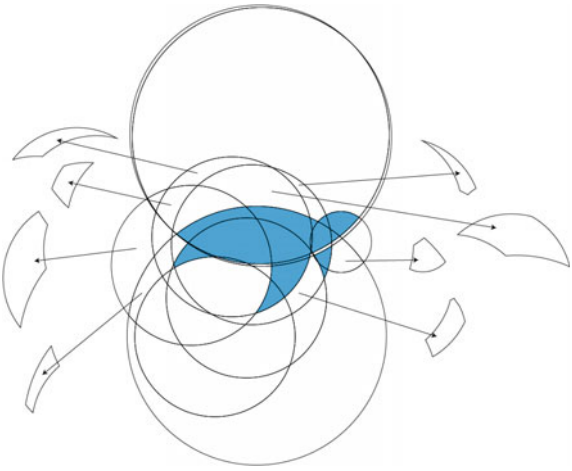


Table 1 Turtle design data

Designs	Center (X)	Center (Y)	Radius
	898	374	53.4775
	637	415	140
	678	540	140
	733	472	140
	742	379	140
	708	365	140
	751	538	226.52
	703	174	226.52
	756	186	226.52
	760	190	226.52

but for the human eye it may not be easy to see. As seen in Fig. 6 the left region is some of the excluded when setting boundaries. The genetic algorithm will never even try to see if these regions should be drawn or not which saves much time as mentioned in Sect. 4.2. The right regions are the regions which might be saved or not from the genetic algorithm perspective as it does not know yet. That depends on if their colored pixels in the region or not.

Genetic Algorithm

Genotype: Our genotype design in this genetic algorithm model is based on binary representation. Every gene in the chromosome has a Boolean value which means it can only take two values true or false. Each value of these Boolean values or let's say each genes refers to a region in the design as true means this region should be drawn and false means this region should not be drawn as shown in Figs. 7 and 8. Working with Boolean values reduces the time to mutate the genes or crossover between the genes significantly and also reduces the performance needed to do these tasks.

Crossover: As known in every genetic algorithm, the most important step in the model is the crossover between two parents. There are many techniques to crossover between two parents but the used technique in this research is Uniform Crossover. Uniform Crossover means that each gene has a probability that decides if it should be taken from the first parent or the second parent.

This technique was chosen to increase the mixing between the genes of the two parents which will be more useful with the large number of regions if the design is more complicated than usual.

Mutation: Mutation is used to maintain the generic diversity from one population to another. Which means it can reach the desired goal faster using mutation. Mutation is changing the value of a gene randomly with a new value. It does not have to be a different value; it can be the same current value, but it is preferable to be different from the current value . It does not have to be a value which does not exist in in the population. Normally, the mutation rate is between 1 and 2%. In this research it changes the Boolean value from true to false and vice versa.

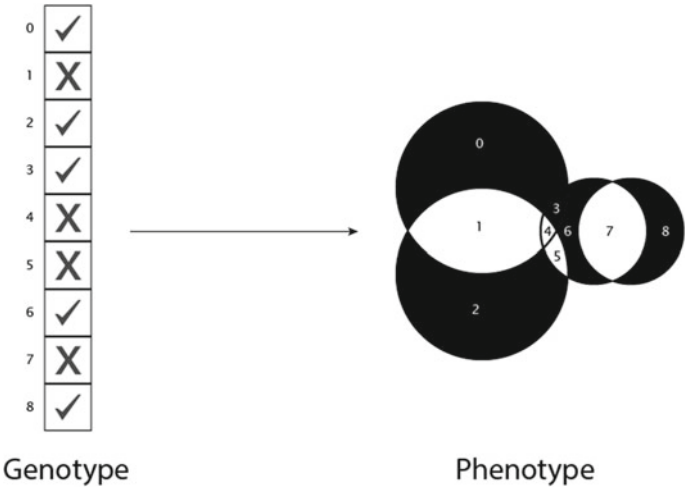


Fig. 7 Genotype and phenotype representative

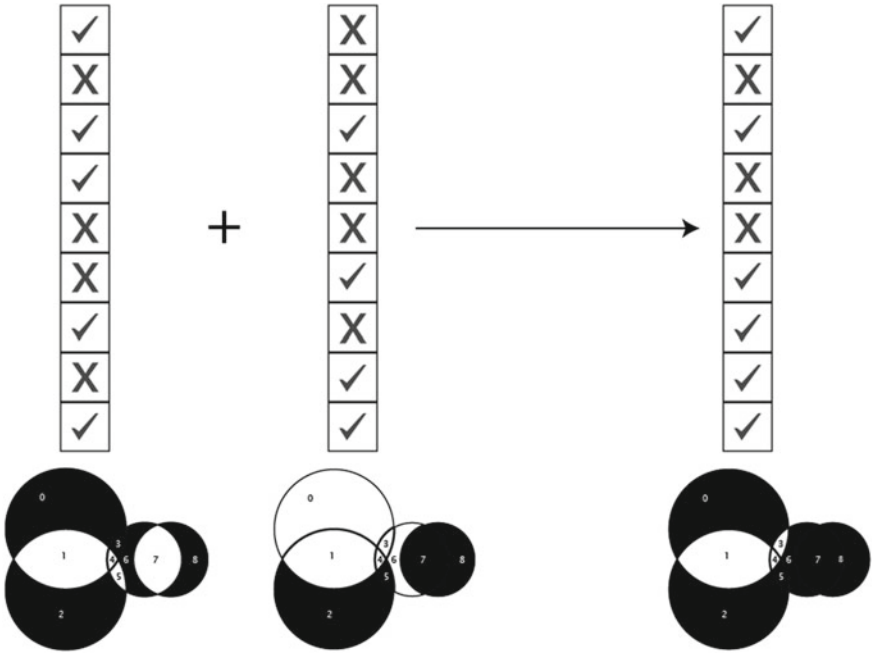


Fig. 8 Genotype and phenotype representative

Model Parameters: There are some parameters that must exist in every genetic algorithm model to work properly and as intended. Firstly, the population size which is how many chromosomes in the single population and in this research, it is equal to 200. Secondly, the chromosome size which is how many genes inside the single chromosome and in this research, it is equal to the number of regions resulted from the Sect. 4.2. The chromosome size can be dynamic sometimes but in this project it is static. Thirdly, the crossover technique and the used technique is the uniform crossover. Fourthly, the elitism which is the chromosomes with the best fitness values to be transferred to the next population without changes to improve the next population's chromosomes' fitness values and reach the desired result much faster. Elitism is represented as percentage value which means for example the best 10% chromosomes ordered by fitness values and is often between 30 and 40% and in this research is 40%. Lastly, the mutation rate which is the percentage to mutate the genes inside a chromosome and change it to another value to maintain the generic diversity and in this research is equal to 2%.

- Population size: 200
- Chromosome size: Number of regions
- Crossover technique: Uniform crossover
- Elitism: 40%
- Mutation rate: 2%.

5 Results & Evaluation

In this section, we will talk about the research's results before and after improving and setting boundaries. There was a slight change in the accuracy as the genetic algorithm usually works fine to reach the final design so no big change in accuracy. The accuracy is approximately from 97 to 99% with the needed time to complete. This percentage does not necessarily mean that it reached 97% of the design. It can mean that the design is successfully made but the percentage is reduced due to the circle lines based on the calculated fitness with the method mentioned in Sect. 4.2. With the set boundaries the time used to reach the final design is reduced by average 50% and maximum 83% based on how regions are connected to the design itself as mentioned in Sect. 4.2. Before setting boundaries, the genetic algorithm took from 8 to 15 hours to complete a design and know which regions to draw and which to not draw. After setting boundaries the time taken improved as it takes from 4 to 6 hours to complete a design based on that design and the circles creating it as mentioned in Sect. 4.2. These results were based on running on 4 virtual machine devices on Microsoft Azure simultaneously to reduce time running all the designs then saved on Microsoft OneDrive.

5.1 Before Setting Boundaries

Firstly, we will talk about the results before setting boundaries and the results generated with the genetic algorithm as shown in Fig. 9. Figure 9a shows the best genes created after the first population which is totally random. It reached about 90%. As seen in Fig. 9a there are some regions outside the images taken into consideration and these regions increase the effort and time consumed to reach the final design. It had also reached some regions to be drawn while they were supposed to be drawn. Most of this calculated fitness percentage came from the white areas as they were correct because it is not supposed to be drawn. It had also reached some of the turtle design like the head and some areas of the legs. The results are improved afterwards in Fig. 9b as more areas were detected to be drawn and they were supposed to be drawn. It reached about 97%. It does not care about the regions outside the design itself so it will keep trying to draw and not draw these regions based on nothing. It reached most of the turtle's body and better shape of the legs. Some colored regions were drawn in Fig. 9a but in Fig. 9b. That does not mean that the genetic algorithm does not work properly it means that bigger colored region was detected to be drawn or bigger non-colored region was detected to not be drawn so it would be taken as a better result as its fitness is higher because it has better genes as mentioned in Sect. 4.2. In Fig. 9c the results improved as now the whole body is detected. It reached about 98.5%. Some small regions in the head and legs are not detected and that means there is still some improvement to be made. Most of the non-colored regions are detected to be not drawn and that increases the accuracy as the design will get closer from the original design. In Fig. 9d the results improved even more and now the human eye can recognize that it is a turtle design. It reached about 98.67%. As seen in Fig. 9d the whole body of the turtle is recognized to be drawn except a small region in the stomach area and a small region in at the left of the or the neck area. There are some small pixels left in the left pair of legs. As we can see it gets closer and closer until it gets the original design. In Fig. 9e it finally reaches the original and the image can be exported as a golden ratio design. It reaches the final design at fitness equal to 98.685%. The remaining is either in some colored pixel in a very small region as it is not detected to the human eye or some non-colored circles

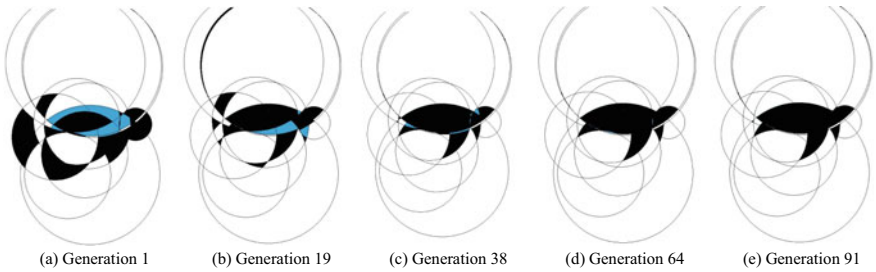
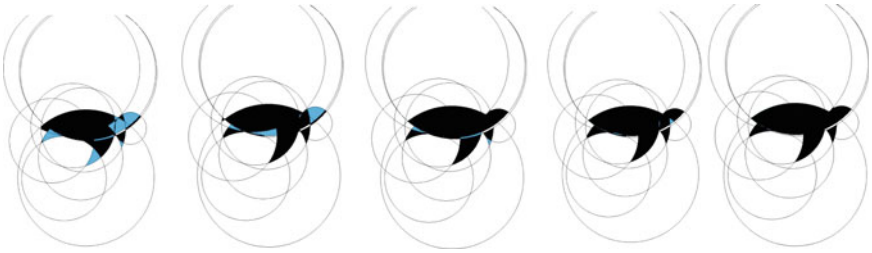


Fig. 9 Turtle improvement through generations before setting boundaries

in the image but colored in the final result as the circles are drawn in the design itself. The circles' borders are black pixels on non-colored pixels, so it gets counted in the fitness function as mentioned in Sect. 4.2.

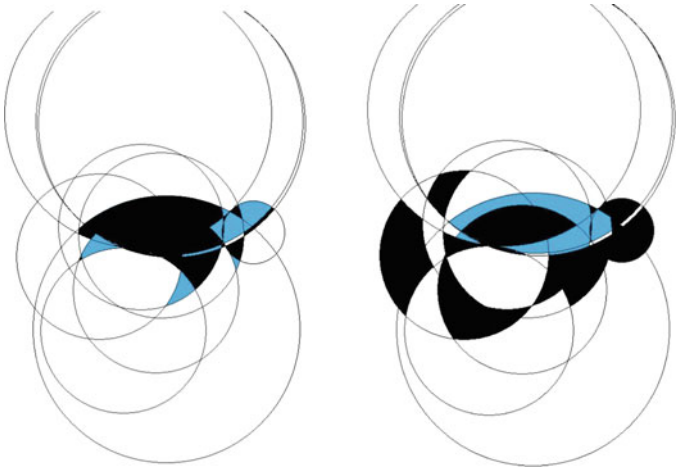
5.2 After Setting Boundaries

Now we will talk about the results after setting boundaries on the same design to see how much improvement happened. As we can see in Fig. 10 the best genes generated randomly are way better than the randomly generated before setting boundaries. It reached 97%. This is much higher than the first randomly generated genes before setting boundaries. Most of the body is detected and a small area from the legs. No areas were detected in the face area. the most important part is most of the white parts are not detected as most of them were not taken into consideration while converting circles to regions as mentioned in Sect. 4.2. The genetic algorithm does not even see or try with these regions. In Fig. 10b the results improved up to about 98%. Almost all of the legs area are detected correctly. Most of the body is also detected correctly. Almost all non-colored areas are detected correctly. In Fig. 10c it improved up to 98.5%. From now on the human eye can easily recognize as a turtle design. Most of the legs area are detected correctly. All body and head areas are detected correctly. All the non-colored area are detected correctly to not be drawn. In Fig. 10d the results improved up to 98.64%. As we can see all that is left to reach the final design is small region in the neck area. The whole body and legs area are detected correctly. All the non-colored regions are still detected correctly. Finally, in Fig. 10e the final is reached successfully with fitness equal to 98.695% like the final result before seating boundaries. All colored regions are detected correctly to be drawn. All non-colored regions are detected correctly to be not drawn.



(a) Generation 1 (b) Generation 10 (c) Generation 22 (d) Generation 29 (e) Generation 38

Fig. 10 Turtle improvement through generations before setting boundaries



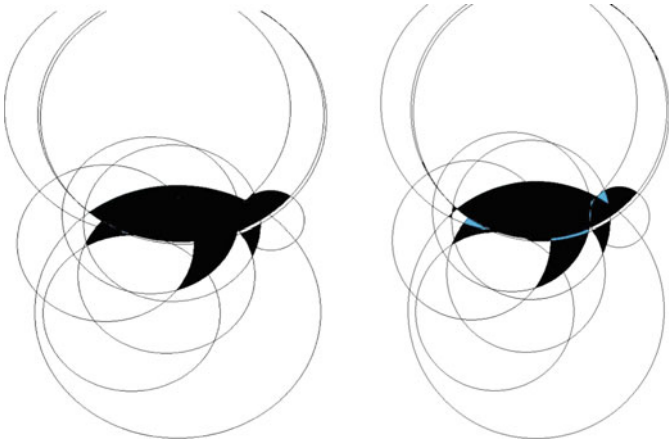
(a) Generation 1 with boundaries (b) Generation 1 without boundaries

Fig. 11 Comparison between first generation after and before boundaries

5.3 Evaluation

As seen in Sects. 5.1 and 5.2 there is a significant change between setting boundaries to the genetic algorithm and not setting it. It does the same process and reaches the same results but in a very less time. As seen in Fig. 11 the first genes which are generated randomly are way better in the method with boundaries as it starts with fitness equal to 97%. This is much higher than the first generation without boundaries which is equal to 90%. Also, almost no non-colored pixels were detected wrong as most of the non-colored region are white in Fig. 8a and most of them did not even try with them. Also, most of the body is detected right from the beginning unlike the method without boundaries.

Now as seen in Fig. 12 in the 38th generation in Fig. 12a after setting boundaries the final result was already reached and the genetic algorithm had already finished processing. All colored regions are detected correctly and drawn in the image and all non-colored regions are detected correctly and not drawn in the final image. In Fig. 12b before setting boundaries it only reached fitness equal to 98.5 and without detecting some regions correctly as mentioned in Sect. 5.1. There was still some room to improve to reach the final design and it did reach in generation 91 as mentioned Sect. 5.1. It also took more time not only because it completed the process in more generations but also because the single generation takes more time than setting the boundaries method. This happens because the genetic algorithm handles more regions and tries to reach the original design with more regions than setting the boundaries method, so it deals with much more pixels as mentioned in Sect. 4.2. The improvement in number of generations is about 58% in this turtle



(a) Generation 38 with boundaries (b) Generation 38 without boundaries

Fig. 12 Comparison between 38th generation after and before boundaries

design and time was reduced from 5 hours to about 2 hours. In summary after setting boundaries, it took less effort, time, and generations to complete the given design comparing to the without boundaries method.

6 Conclusion

In this paper a framework is proposed for generating a golden ratio design using golden circles through a sketched design from the user so that the output design is near to the input design.

Genetic algorithm is used to optimize the output design. i.e., maximize the similarity between the input sketched design by the user and the output design using golden ratio in less computational time.

The presented results showed a great significant between setting boundaries to the genetic algorithm and not setting it. It was shown that the first genes which are generated randomly are way better in the method with boundaries as it starts with fitness equal to 97%. This is much higher than the first generation without boundaries which is equal to 90%. Also, almost no non-colored pixels were detected wrong as most of the non-colored region are white and most of them did not even try with them. Also, most of the body is detected right from the beginning unlike the method without boundaries, and after setting boundaries the final result was already reached and the genetic algorithm had already finished processing. All colored regions are detected correctly and drawn in the image and all non-colored regions are detected correctly and not drawn in the final image.

References

1. Li, T.M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. ACM Trans. Graph. **39**(6), (2020)
2. Sezgin, T.M., Stahovich, T., Davis R.: Sketch Based Interfaces, p. 37 (2007)
3. Yu, B., Cai, S.: A domain-independent system for sketch recognition. In: Proc. 1st Int. Conf. Comput. Graph. Interact. Tech. Australas. South East Asia, Graph. '03, pp. 141–146 (2003)
4. Paulson, B., Hammond, T.: PaleoSketch: accurate primitive sketch recognition and beautification. In: Int. Conf. Intell. User Interfaces, Proc. IUI, pp. 1–10 (2008)
5. Gogna, A., Tayal, A.: Metaheuristics: review and application. **25**(4), 503–526 (2013). https://doi.org/10.1080/0952813X.2013.782347
6. Mohamed, A.W., Hadi, A.A., Mohamed, A.K.: Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. Int. J. Mach. Learn. Cybern. **11**(7), 1501–1529 (2020)
7. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimed. Tools Appl. **80**, 8091–8126 (2021)
8. Holland, J.H.: Adaptation in Natural and Artificial Systems. Univ. Michigan Press, Michigan (1975)

Antenna Array Design Using Differential Evolution with Ranking-Based Mutation Operators

Sotirios K. Goudos and Ali Wagdy Mohamed

Abstract The topic of designing linear arrays with design limitations such as the minimum and maximum distance between two adjacent elements is addressed in this book chapter. We apply a design framework based on Differential Evolution (DE) with ranking-based mutation operators. This DE variant uses a different mutation operator than the original DE. The basic idea in this DE variant is to probabilistically select the vectors for the mutation operator based on their ranking in the current population instead of using a random selection mechanism which is the case in the original DE. We perform a comparative study on four different linear array design cases among different DE variants with and without ranking-based mutation operator. The results indicate the ranking-based mutation operator gives an additional advantage to design process. The ranking-based DE variants obtain in general better results and seem to obtain faster convergence as well.

1 Introduction

One of the most important components of any wireless communication system is the antenna(s). This fact results to challenging research tasks regarding antenna design. Linear array synthesis is one of the most popular and widely studied research topics using both analytical or stochastic methods [7, 8, 23, 25, 27, 36, 37]. Several applications in both satellite and wireless communications in general require an efficient array design. Moreover, during the recent years there is growing application domain

S. K. Goudos (✉)
ELEDIA@AUTH, Department of Physics, Aristotle University of Thessaloniki, Thessaloniki, Greece
e-mail: sgoudo@physics.auth.gr

A. W. Mohamed
Operations Research Department, Faculty of Graduate Studies for Statistical Research, Cairo University, Giza 12613, Egypt

Department of Mathematics and Actuarial Science School of Sciences & Engineering, The American University in Cairo, New Cairo City, Egypt

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_9

for these arrays that as a result generates several new research challenges. There are more new and strict design requirements like pattern shaping, low profile, wideband/narrowband application, and interference cancellation. Additional limitations are imposed like power dissipation and small antenna size.

The main conclusion of the above requirements is the need for efficient, simple and fast optimization methods. In this context, during the last two decades, several new optimization methods have been proposed that are based on mathematical modelling of biological evolution or the way biological entities interact with each other in nature. Such methods belong to the family of Evolutionary algorithms (EAs) [11]. EAs among others include Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), and Differential Evolution (DE). The application of the above-mentioned algorithms to antenna array design problems is very common [16, 18–20, 23, 24, 29, 30, 33]. In this chapter, we apply several different DE variants to the linear array optimization problem.

2 Related Work

Differential Evolution (DE) is global optimizer that uses vectors to evolve with the use of three operators: mutation, crossover, and selection. DE is a mathematical design and it is not based on a certain biological phenomenon. There are several DE variants or strategies in the literature that depend on the form of these operators. The choice of the best DE strategy depends on the problem type. DE requires the setting of two control parameters, i.e., the mutation control parameter (F) and the crossover constant (CR), while it is based on a greedy selection scheme Moreover, DE is also very popular for antenna design, as it has been used widely, especially for antenna array design.

One may find a review paper that presents the application of DE in electromagnetics in [42]. Table 1 briefly reports representative papers that use DE for antenna array design. One may notice that the original DE algorithm with *DE/rand/1/bin* strategy is quite common in the literature for array design. Some of reported applications of the original DE strategy include linear array synthesis [30], synthesis of monopulse antennas [2], array pattern nulling in [50]; and conformal array design [22]. Moreover, different DE strategies have also been applied to array design problems. Some have strategies have been introduced especially for array design like the DDE/BoR/1/bin variant used for linear array design in [33]. Similarly the authors in [3] present a modified DE Strategy (MDES) for linear array synthesis. The design of time modulated arrays is another research topic that been studied in [31, 48, 49] using the *DE/best/1/bin* strategy.

Among others self-adapting DE variants that do not require setting of the mutation control parameter and the crossover constant are very popular in the literature. The authors in [1] provide a DE strategy (named jDE by the authors) that self-adapts the F, CR parameters. This jDE algorithm has been used in the electromagnetics

Table 1 List of different antenna array design cases using DE

DE variant	Array type	References
rand/1/bin	Linear	[30]
DDE/BoR/1/bin	Linear	[33]
rand/1/bin	Linear	[4]
rand/1/bin	Difference patterns of monopulse antennas	[2]
rand/1/bin	Array pattern nulling	[50]
Modified DE Strategy (MDES)	Linear	[3]
Multi-objective DE	Linear	[39]
Improved DE	Array design	[32]
CODE-EIG	Shaped beam synthesis	[15]
rand/1/bin	Conformal	[22]
SaDE	Sparse linear arrays synthesis	[13]
Memetic GDE3	Subarray design	[20]
best/1/bin	Time-modulated arrays design	[48]
Hybrid DE	Monopulse antenna with a subarray weighting	[34]

domain for solving optimization problems such as the microwave absorber design problem [14] and the linear array design. [5, 18].

Moreover, SaDE [26, 40, 41] is another DE algorithm that uses a self adaptation mechanism. The authors in [17] apply SaDE for microwave filter design [17], and for linear arrays synthesis design [18]. JADE [51] is another self-adapting DE algorithm that introduces the concept of an external archive. JADE has been applied in [11] for linear array design. Moreover, the Composite DE (CoDE) [47] is another adaptive DE variant. Moreover, a CoDE variant, CoDE-EIG is introduced in [10] and applied to shaped beam array synthesis.

Additionally, in authors in [38] present the Barebones DE. This algorithm requires only one control parameter. This is a DE algorithm based on concepts from the barebones particle swarm optimizer [28]. Hence, the authors in [46] introduce two new barebones DE variants, namely the Gaussian Barebones DE (GBDE) and the Modified Gaussian Barebones DE (GBDE). In [6] the authors apply different self-adaptive DE algorithms for waveform design for wireless power transfer. Although, DE inherently works in real spaces, binary extensions also exist. In [21] a binary DE algorithm is used for thinned array design.

3 Differential Evolution

The Differential evolution (DE) [44, 45] algorithm is a very well-known evolutionary algorithm with a wide range of applications. DE is a mathematical construct, it does not model any biological process. DE uses three operators that guide the evolution in each iteration. Namely, these operators are the mutation, crossover and selection. One may

DE algorithms apply three operators to evolve the population in each generation. These are mutation, crossover and selection. In the literature, there are several different DE variations [43, 45]. One of features that usually distinguishes these variations from the original algorithm is the mathematical description of the DE operators. The most suitable DE method or variant to use for a certain optimization problem depends on the characteristics of the problem [35]. Among others the popular DE strategies for obtaining a mutant vector include the *DE/rand/1/bin* and *DE/best/1/bin*. The mutant vector produced by the pth member of the population in the jth iteration is formulated as

$$\begin{array}{l} DE/rand/1/bin \\ \bar{M}_{j,p} = \bar{X}_{j,r_1} + F \times (\bar{X}_{j,r_2} - \bar{X}_{j,r_3}), r_1 \neq r_2 \neq r_3 \\ DE/best/1/bin \\ \bar{M}_{j,p} = \bar{X}_{j,best} + F \times (\bar{X}_{j,r_1} - \bar{X}_{j,r_2}), r_1 \neq r_2 \end{array} \tag{1}$$

where r_1, r_2, and r_3 denote the randomly selected population vectors, F represents the mutation control parameter, $\bar{M}_{j,p}$ is the mutant vector, and $\bar{X}_{j,p}$ is the original member of the population. Next, DE applies the crossover operator expressed as

$$C_{j,pm} = \begin{cases} M_{j,pm}, \text{ if } rand_{m[0,1)} \leq CR \text{ or } \mathrm{m} = \mathrm{rand_i ndex(n)} \\ X_{j,pm}, \text{ if } rand_{m[0,1)} > CR \text{ and } \mathrm{m} \neq \mathrm{rand_i ndex(n)} \end{cases} \tag{2}$$

where $m = 1, 2, \ldots, D$,$rand_{m[0,1)}$ denotes a uniformly distributed random number in the interval [0,1), $rand_i ndex(n)$ denotes a randomly chosen index from $(1, 2, \ldots, D)$, and CR is the DE crossover control parameter. The latter is usually set from within [0, 1]. Finally, DE applies a greedy selection operator formulated as

$$\bar{X}_{j+1,p} = \begin{cases} \bar{C}_{j,p}, \text{ if } f(\bar{C}_{j,p}) < f(\bar{X}_{j,p}) \\ \bar{X}_{j,p}, \text{ otherwise} \end{cases} \tag{3}$$

where $f(\bar{X}_{j,p}), f(\bar{C}_{j,p})$, denote the fitness function values of old population member and the trial vector, respectively. The above formulation is given for minimization problems.

Thus, in the jth iteration the previous population member $\bar{X}_{j,p}$ will be replaced by the new one $\bar{U}_{j,p}$ only if the new one obtains a smaller fitness value. The description of the DE algorithm is given in Algorithm 1.

Algorithm 1 DE algorithm

```
Initialization of the DE control parameters.
Set the values for the mutation control parameter F, and the crossover constant CR.
Set the maximum number of iterations MAXJ, and the population size Pop
Random initialization of the initial population
Get the fitness value for every population member
J = 1
while J < MAXJ do
  for each member p of the population do
    Mutation of the original X̄_{j,p} member of population using (1) and creation of the mutant
    vector M̄_{j,p}.
    Apply the crossover operator using (2) and create a trial vector C̄_{j,p}.
    Compute the fitness value of the trial vector.
    Selection operator using (3).
    If a new global best vector is found then update.
  end for
  J = J + 1
end while
```

3.1 Self-adaptive de Algorithms

DE needs the input of the mutation and crossover control parameters F and CR, respectively. As a result, the fact that only two control parameters are required is a DE advantage. The fine-tuning of these control settings, on the other hand, may entail a time-consuming trial-and-error method. In consideration of this, multiple self-adaptive DE variations have been discussed and analyzed in the literature. These variations include the jDE [1] and JADE [51] algorithms.

3.1.1 jDE Algorithm

The differential evolution control parameters F and CR should be set from the intervals [0.5, 1] and [0.8, 1], respectively, according to [45]. In [1], the authors offer a self-adaptive technique known as jDE. The mutation operator in the jDE variation is the same as in the *DE/rand/1/bin* original mutation scheme. The jDE approach is built on the idea of each vector having two additional variables. These are the F and CR values for it. These control settings are evolved by jDE. The jDE technique generates new vectors with improved control parameter values. These vectors are thought to have a higher chance of surviving and producing better offsprings in the next iteration. As a result, the improved control parameter values are transferred over to the upcoming iteration.

In each iteration, the control parameters are self-adjusted for each member of the population using the formulas:

$$F_{j+1,p} = \begin{cases} F_{low} + rnd_{1[0,1]} \times F_{upper} & \text{if } rnd_{2[0,1]} < p^1 \\ F_{j,p}, & \text{otherwise} \end{cases}$$
$$CR_{j+1,p} = \begin{cases} rnd_{3[0,1]} & \text{if } rnd_{4[0,1]} < p^2 \\ CR_{j,p}, & \text{otherwise} \end{cases} \quad (4)$$

where $rnd_{s[0,1]}, s = 1, 2, 3, 4$ are uniform random numbers $\in [0, 1]$, F_{low}, F_{upper} denote the lower and the upper limits of F, that are set to 0.1 and 0.9, respectively, and p^1 and p^2 denote the probabilities of adjusting the control parameters. The latter are set both to 0.1 according to the authors of the original jDE paper. Moreover, the authors in [1] claim that jDE performs better or equally as the original DE algorithm. Another jDE advantage is the fact it does not increase the time complexity of the DE algorithm. Furthermore, the performance of the jDE strategy is better or at least similar to that of the standard *DE/rand/1/bin* [1] mutation. In addition, using jDE does not contribute to the time complexity. The reader who is interested in learning more about the jDE technique may find additional details in [1].

3.1.2 The JADE Algorithm

The JADE [51] algorithm applies the *DE/current-to-pbest/1/bin* strategy. This technique is used in conjunction with an external archive, which is optional. The optional archive's purpose is to make use of earlier data in order to preserve population variety. The technique outlined above diversifies the population while also improving convergence. For each target vector, the JADE creates a new crossover constant and mutation control parameter using a normal and a Cauchy distribution. JADE generates new control parameters based on recently successful ones. The JADE technique employs not only the present population's best vector, but also additional good vectors. In order to generate new trial vectors, the mutation procedure may also use vectors from the external archive.

3.2 DE with Ranking-Based Mutation Operators (RankDE)

The basic concept of this DE variant is to select some of the vectors used in mutant vector generation according to their rankings in the population [9] and not randomly as in the original DE. In RankDE, first, each vector in the mutation operator is issued a rank based on its fitness. This is formulated as

$$R_k = NP - k, \quad k = 1, 2 \ldots N \quad (5)$$

RankDE defines the selection probability of each vector as

$$p_k = \frac{R_k}{NP}, \quad k = 1, 2 \ldots N \tag{6}$$

where N denotes the population size. As a result, the $r1$ and $r2$ vector indices are chosen based on their selection probabilities, while the remaining vector indices in the mutation operation are chosen at random. This means that in the mutation operator vectors with greater selection probabilities are more likely to be chosen. The DE exploitation ability can be improved in this way. Algorithm 2 briefly describes the ranking-based mutation operators DE. Algorithm 2.

Algorithm 2 RankDE algorithm

1: Initialization of the DE control parameters.
2: Set the values for the mutation control parameter F, and the crossover constant CR.
3: Set the maximum number of iterations $MAXJ$, and the population size Pop
4: Random initialization of the initial population
5: Get the fitness value for every population member
6: $J = 1$
7: Sort population based on fitness value
8: Compute the selection probability of each population member using (6).
9: **while** $J < MAXJ$ **do**
10: **for** each member p of the population **do**
11: Mutation of the original $\bar{X}_{j,p}$ member of population using (1) and by selecting r_1 and r_2 based on ranking of the current population.
12: Creation of the mutant vector $\bar{M}_{j,p}$.
13: Apply the crossover operator using (2) and create a trial vector $\bar{C}_{j,p}$.
14: Compute the fitness value of the trial vector.
15: Selection operator using (3).
16: If a new global best vector is found then update.
17: **end for**
18: Sort population based on fitness value
19: Update the selection probability of each population member using (6).
20: $J = J + 1$
21: **end while**

4 Linear Array Design

Lets assume an T-element linear-array arranged symmetrically along the x-axis, which is shown in Fig. 1. Then we can express array factor in the $x - z$ plane by the following equation

$$AF(\vartheta, \bar{x}, \bar{\phi}, \bar{A}) = \sum_{t=1}^{T} A_n e^{j\left(\frac{2\pi}{\lambda} x_t \sin\vartheta + \phi_t\right)} \tag{7}$$

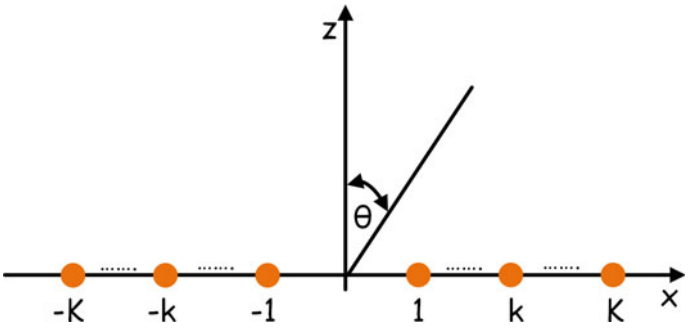


Fig. 1 Geometry of a linear array

where the wavelength is denoted with λ, ϑ denotes the steering angle measured with respect to the positive direction of the z-axis, x_t, A_t, and ϕ_t denote the position, amplitude, and the phase of the tth element, respectively, and $\bar{x}$, $\bar{A}$ and $\bar{\phi}$ denote the corresponding vectors. If the linear is symmetrically excited array (7) then due to symmetry we may write (7) as

$$AF(\vartheta, \bar{x}, \bar{\phi}, \bar{A}) = 2\sum_{k=1}^{K} A_k \cos\left[\frac{2\pi}{\lambda} x_k \sin\vartheta + \phi_k\right] + A_0 \tag{8}$$

where $K = \left\lfloor \frac{T}{2} \right\rfloor$ represents the largest integer number less than or equal to $T/2$. If the array has an even number of elements then $A_0 = 0$, otherwise it is $A_0 = 1$. The following constraints apply to the placement of the array elements:

$$x_1 \geq \frac{\delta_{\min}}{2},\ x_1 \leq \frac{\delta_{\max}}{2}, \qquad \text{when } T = 2K \tag{9}$$

$$|x_0 - x_1| \geq \delta_{\min},\ |x_0 - x_1| \leq \delta_{\max}, \ \text{when } T = 2K + 1 \tag{10}$$

$$\left.\begin{array}{l} |x_i - x_{i+1}| \geq \delta_{\min} \\ |x_i - x_{i+1}| \leq \delta_{\max} \end{array}\right\}, 1 \leq i \leq K - 1$$

where $\delta_{\min}$ and $\delta_{\max}$ denote the minimum and maximum allowable interelement distance, respectively. We assume a symmetrically arranged linear array along the x-axis. that has uniform amplitudes equal to 1. In this case, we may formulate the array factor in the $x - z$ plane as:

$$AF(\vartheta, \bar{y}, \bar{\phi}) = 2\sum_{k=1}^{K} \cos\left[\frac{2\pi}{\lambda} y_k \sin\vartheta + \phi_k\right] \tag{11}$$

If we express Equation (11) in dB it becomes:

$$AF_{\mathrm{dB}}\left(\vartheta, \bar{x}, \bar{\phi}\right) = 20 \cdot \log_{10}\left|\frac{AF\left(\vartheta, \bar{x}, \bar{\phi}\right)}{AF\left(\vartheta_o, \bar{x}, \bar{\phi}\right)}\right| \tag{12}$$

where ϑ_o denotes the direction of the maximum. The suppression of sidelobe level (SLL) is the optimization target. This is accomplished in the case of position only or phase-position design by determining the best element positions and phases. The objective function that should be minimized in order to solve this problem is given by the expression:

$$F_1(\bar{y}, \bar{\phi}) = \max_{\vartheta \in O}\{AF_{dB}(\vartheta, \bar{y}, \bar{\phi})\} \tag{13}$$

where O denotes the set of theta angles that are outside the angular range of the mainlobe.

5 Numerical Results

As in [12] the linear-array synthesis design cases are executed 50 times for each algorithm. The algorithms selected for this case are the original DE with *DE/rand/1/bin* mutation strategy and the two self-adapting strategies the jDE [1] and JADE [51]. Moreover, we compare the above algorithms with their ranking based mutation operators.

We examine two different design options for uniformly excited arrays, which include position-only, and position-phase synthesis. For position-only synthesis the number of unknowns is K, while for position-phase synthesis this number is $2K$. We set $\delta_{\min} = 0.5\lambda$ for all cases, where $\delta_{\min}$ denotes the minimum interelement distance. In all runs we set the population size to 100 and the maximum number of iterations to 2000. In DE we set the control parameters as $F = 0.5$, $CR = 0.9$.

5.1 Position only Synthesis

We consider first two different position-only synthesis cases. First, we set the maximum interelement distance to $\delta_{\max} = 0.8\lambda$. In this case the decision variables are the interelement distances within the interval [0.5, 0.8]. We set the number of elements to 64 so that the problem dimensions in both cases is 32. Table 2 reports the comparative results after 50 independent runs. We notice that all the algorithms results are very close. This is due to the small search space. However, one may clearly see that the ranking versions in two cases slightly outperform the original versions, which is true for DE and the jDE. It seems that the ranking version performs equally well the original JADE. In terms of best results found, RankDE and RankjDE obtain a better result than the original versions.

Table 2 Position only synthesis with 64-elements $d_{\max} = 0.8\lambda$. Statistical results

Algorithm	Best	Worst	Mean	Median	St.Dev
DE	−20.84	−20.8	−20.82	−20.82	1.07E-02
RankDE	−20.89	−20.85	−20.88	−20.88	8.22E-03
jDE	−20.87	−20.82	−20.84	−20.84	1.31E-02
RankjDE	−20.88	−20.84	−20.86	−20.86	1.04E-02
JADE	−20.89	−20.77	−20.84	−20.86	3.90E-02
RankJADE	−20.86	−20.82	−20.84	−20.84	1.08E-02

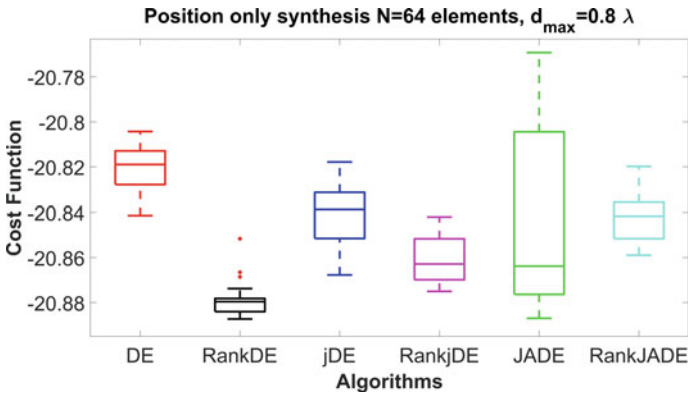


Fig. 2 Position only synthesis with 64-elements $d_{\max} = 0.8\lambda$. Boxplot of all algorithms results

Figure 2 shows the boxplot graph for this case. We notice that clearly the ranking versions of all algorithms obtain results with smaller dispersion of values than the corresponding original versions. The corresponding convergence rate graph is shown in Fig. 3. We notice that the ranking versions seem to converge at similar speed faster the corresponding original version. Moreover, the radiation pattern of the best array found by RankDE is plotted in Fig. 4 with sidelobe level (SLL) equal to −20.89 dB. Table 3 holds the elements position of this best pattern.

Next, we study another array case where we have set the maximum interelement distance to $\delta_{\max} = \lambda$. Thus, the search space is now larger than in the previous case and all decision variables are within the interval [0.5, 1]. Table 4 lists the algorithms comparative results. Again, we notice that in terms of mean values RankDE and RankjDE obtain better results than the original versions. RankJADE obtains slightly worst results than the ones of the initial JADE. Additionally, the best values obtained by RankDE and RankjDE are lower than those of the original versions, which is also the case for the worst values as well.

The boxplot graph for this case is depicted in Fig. 5. In this case the ranking versions except RankJADE obtained results with higher dispersion of values than the original versions. However, their results are mostly at lower SLL values. Figure 6

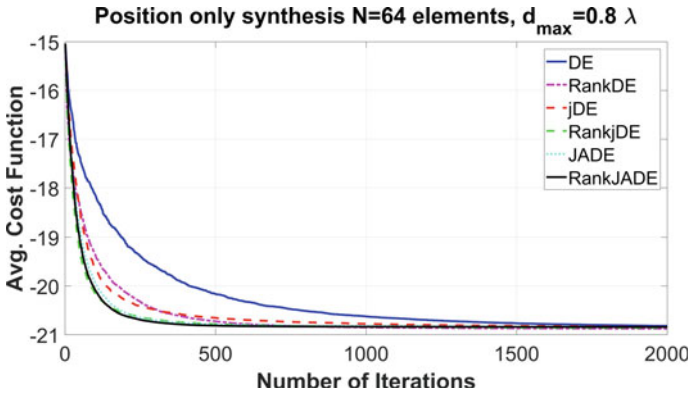


Fig. 3 Position only synthesis with 64-elements $d_{\max} = 0.8\lambda$. Convergence rate graph

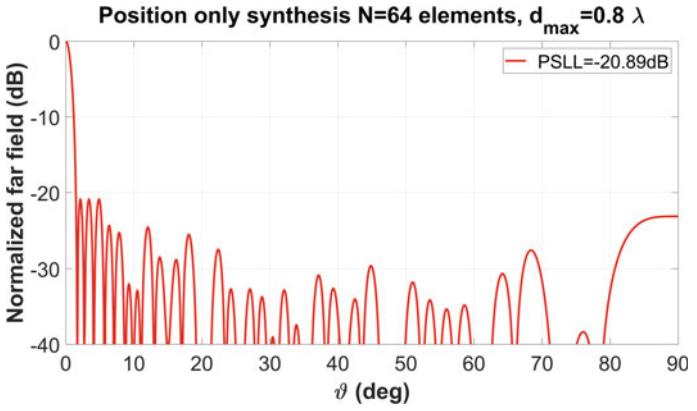


Fig. 4 Position only synthesis with 64-elements $d_{\max} = 0.8\lambda$. Best radiation pattern found

Table 3 Position only synthesis with 64-elements $d_{\max} = 0.8\lambda$. Element Positions of the best obtained result

k	x_k	k	x_k	k	x_k	k	x_k
1	0.250	9	4.250	17	8.720	25	14.234
2	0.750	10	4.750	18	9.223	26	15.034
3	1.250	11	5.250	19	9.753	27	15.834
4	1.750	12	5.750	20	10.253	28	16.634
5	2.250	13	6.252	21	11.035	29	17.434
6	2.750	14	6.752	22	11.834	30	18.234
7	3.250	15	7.270	23	12.634	31	19.034
8	3.750	16	8.068	24	13.434	32	19.834

Table 4 Position only synthesis with 64-elements $d_{\max} = \lambda$. Statistical results

Algorithm	Best	Worst	Mean	Median	St.Dev
DE	−23.32	−22.59	−22.96	−23.03	1.98E-01
RankDE	−23.16	−22.63	−22.98	−23.10	2.06E-01
jDE	−23.24	−22.93	−23.04	−23.03	7.93E-02
RankjDE	−23.38	−22.68	−23.10	−23.05	1.91E-01
JADE	−23.13	−22.66	−22.88	−22.73	2.12E-01
RankJADE	−23.11	−22.58	−22.72	−22.65	1.62E-01

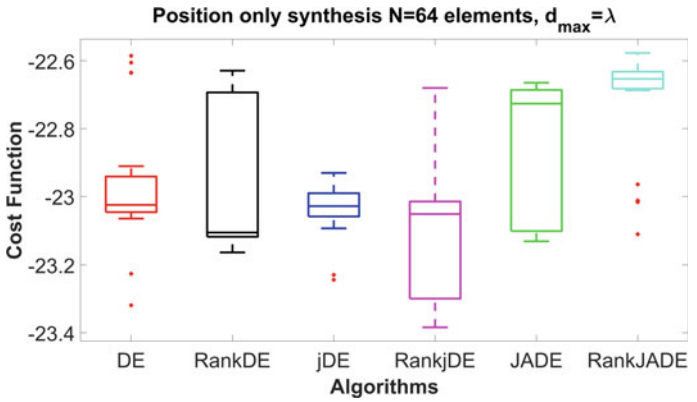


Fig. 5 Position only synthesis with 64-elements $d_{\max} = \lambda$. Boxplot of all algorithms results

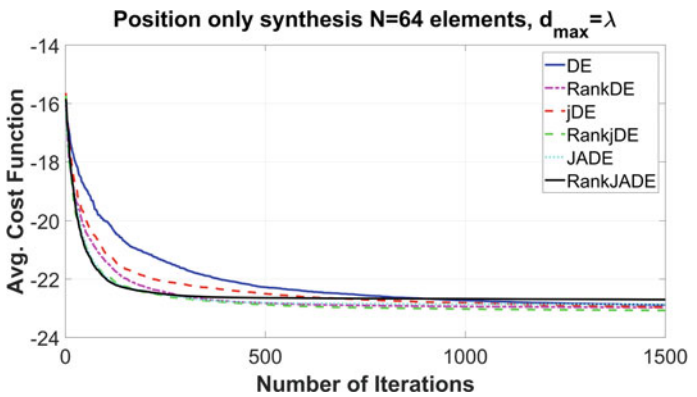


Fig. 6 Position only synthesis with 64-elements $d_{\max} = \lambda$. Convergence rate graph

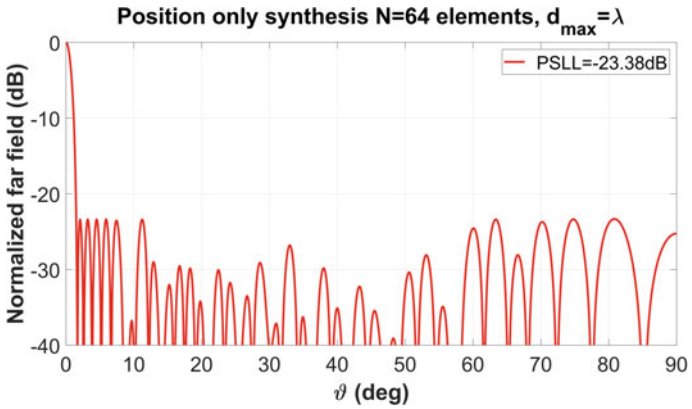


Fig. 7 Position only synthesis with 64-elements $d_{\max} = \lambda$. Best radiation pattern found

Table 5 Position only synthesis with 64-elements $d_{\max} = \lambda$. Element Positions of the best obtained result

k	x_k	k	x_k	k	x_k	k	x_k
1	0.250	9	4.287	17	8.780	25	14.446
2	0.751	10	4.787	18	9.543	26	15.374
3	1.257	11	5.287	19	10.043	27	16.307
4	1.764	12	5.787	20	10.543	28	17.260
5	2.264	13	6.303	21	11.069	29	18.158
6	2.766	14	6.826	22	11.719	30	19.150
7	3.271	15	7.575	23	12.531	31	20.141
8	3.783	16	8.199	24	13.520	32	21.076

plots the convergence rate graph. The graph is similar to that of the previous case. Both RankDE and RankjDE converge faster than the original versions. RankJADE and JADE seem to converge at similar speeds. Furthermore, the radiation pattern of the best solution found by RankjDE is plotted in Fig. 7. The peak SLL is −23.38 dB. Table 5 lists the decision variables for this pattern case.

5.2 Position Phase Synthesis

Next, we evaluate the algorithms performance in two more complex cases. We have both position and phase optimization. We consider a 70-element array, which results to a total of 70 problem dimensions. The element phase are within the interval $[0°, 359°]$, while the interelement distances are as in previous case within the interval $\delta_{\min}, \delta_{\max}$.

Table 6 Position phase synthesis with 70-elements $d_{\max} = 0.8\lambda$. Statistical results

Algorithm	Best	Worst	Mean	Median	St.Dev
DE	−23.43	−22.26	−22.98	−23.02	4.30E-01
RankDE	−24.09	−23.18	−23.66	−23.70	3.46E-01
jDE	−23.63	−21.80	−22.98	−23.18	5.73E-01
RankjDE	−23.62	−22.08	−22.93	−22.88	4.80E-01
JADE	−23.25	−20.71	−21.36	−21.24	7.26E-01
RankJADE	−23.53	−21.03	−22.43	−22.50	8.20E-01

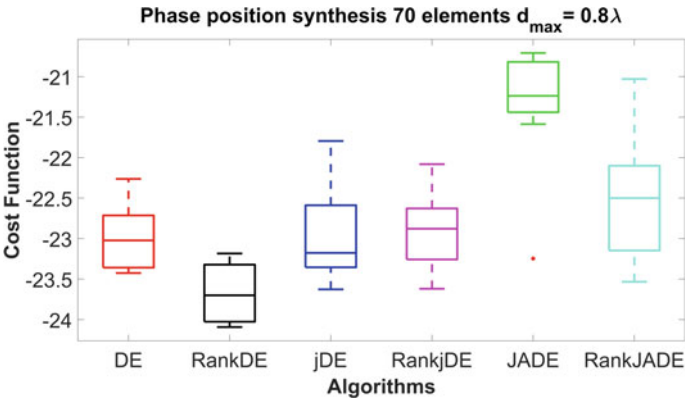


Fig. 8 Phase-Position synthesis with 70-elements $d_{\max} = 0.8\lambda$. Boxplot of all algorithms results

We evaluate the algorithms first at a case with $\delta_{\max} = 0.8\lambda$. Table 6 reports the comparative results. We notice for the case of RankDE and RankJADE the ranking versions clearly outperform the corresponding original versions. In this case the improvement in terms of mean values is high up to about 1 dB. RankjDE performs slightly worse than the original algorithm. Figure 8 presents the boxplot graph for this case. We notice that the distribution of values is quite similar for both the ranking and the original version, except for the RankJADE case. However, RankJADE obtained better SLL values in general. The convergence speed graph of Fig. 9 shows both the ranking and the original versions converge at the same speed. No convergence advantage is shown for any algorithm. Moreover, Fig. 10 draws the radiation pattern of two best solutions found by RankDE and RankjDE with peak SLL values −24.09 dB and −23.62 dB, respectively. The uniform array of 70 elements is also plotted in the graph. We notice that the optimization process improves the peak SLL by more than 10 dB compared with the uniform array. Table 7 holds the element positions and phases of the best array found by RankDE.

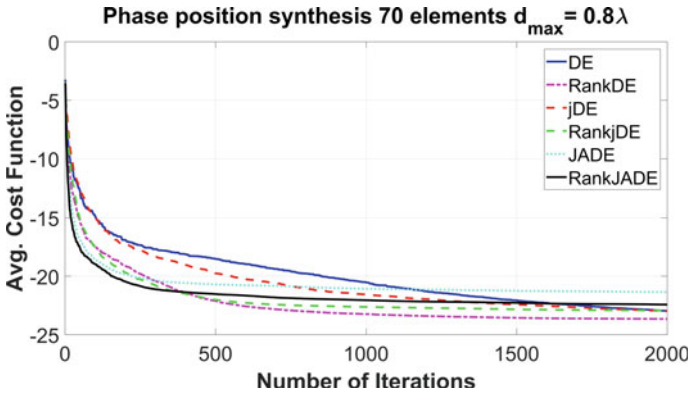


Fig. 9 Phase-Position synthesis with 70-elements $d_{\max} = 0.8\lambda$. Convergence rate graph

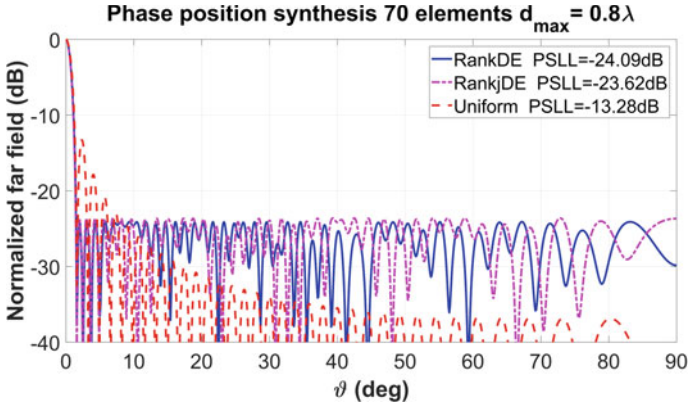


Fig. 10 Phase-Position synthesis with 70-elements $d_{\max} = 0.8\lambda$. Best radiation patterns found

The last design case is that of an 70 element array with $\delta_{\max} = \lambda$. This means that the search space is larger compared with the previous case. The comparative results are listed in Table 8. We notice in terms of mean values only RankDE outperforms the original version. However, both RankDE and RankjDE obtained lower best solution, which can be lower than about 0.5 dB. JADE obtained better results than the ranking version. Moreover, both JADE versions obtained worst results than DE and jDE versions.

Figure 11 draws the boxplot graph for the $\delta_{\max} = \lambda$ case. It seems that in most cases the distribution of values is larger in the ranking cases. The only exception is the RankJADE case. RankDE and RankJADE obtain results with a smaller median value. Furthermore, Fig. 12 shows the convergence rate graph. Apparently, the original

Table 7 Position phase synthesis with 70-elements $d_{\max} = 0.8\lambda$. Element Positions of the best obtained result

k	x_k	$\varphi_k{}^\circ$	k	x_k	$\varphi_k{}^\circ$
1	0.274	197.147	21	11.676	193.504
2	0.776	198.576	22	12.422	199.393
3	1.291	196.625	23	13.214	187.461
4	1.812	199.044	24	13.982	218.487
5	2.323	197.764	25	14.762	168.163
6	2.841	199.130	26	15.552	194.972
7	3.389	193.484	27	16.350	204.429
8	3.894	201.686	28	17.149	184.504
9	4.441	195.465	29	17.650	34.310
10	4.944	196.973	30	18.150	213.768
11	5.486	198.723	31	18.935	209.904
12	6.047	196.785	32	19.731	243.543
13	6.633	198.696	33	20.510	237.503
14	7.240	195.995	34	21.197	153.696
15	7.761	195.375	35	21.957	171.693
16	8.373	204.729			
17	9.002	195.895			
18	9.583	201.044			
19	10.321	195.395			
20	11.050	208.318			

Table 8 Position phase synthesis with 70-elements $d_{\max} = \lambda$. Statistical results

Algorithm	Best	Worst	Mean	Median	St.Dev
DE	−24.71	−23.50	−24.20	−24.23	4.32E-01
RankDE	−25.73	−24.20	−25.03	−25.31	5.43E-01
jDE	−25.01	−23.55	−24.52	−24.61	4.36E-01
RankjDE	−25.50	−21.88	−24.01	−24.13	1.14E+00
JADE	−24.48	−22.09	−23.03	−22.70	8.60E-01
RankJADE	−23.74	−21.40	−22.74	−22.85	6.72E-01

algorithms converge at similar speed with the ranking versions. The radiation patterns of two best results found by RankDE and RankjDE are depicted in Fig. 13 as well as the corresponding uniform array pattern. Table 9 lists the array configuration of the best solution found by RankDE.

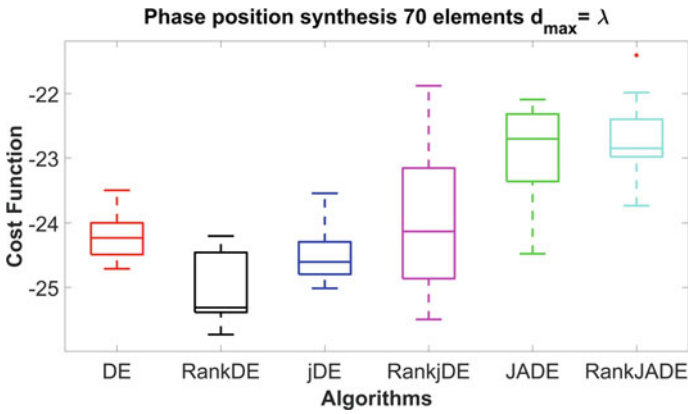


Fig. 11 Phase-Position synthesis with 70-elements $d_{\max} = \lambda$. Boxplot of all algorithms results

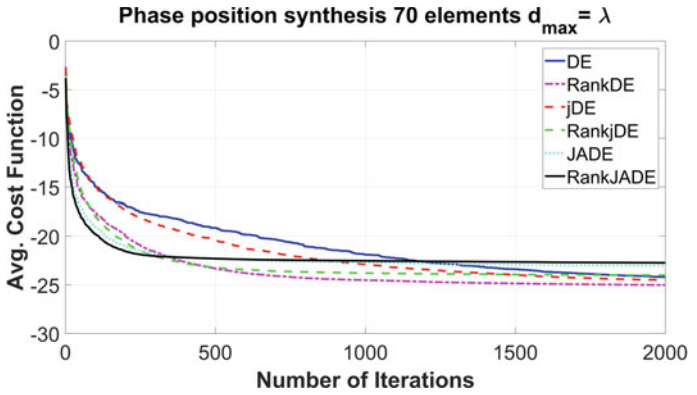


Fig. 12 Phase-Position synthesis with 70-elements $d_{\max} = \lambda$. Convergence rate graph

6 Conclusion

In this book chapter, we have evaluated the performance of DE with ranking-based mutation operators on different linear array design cases. The results show that ranking-based mutation operators perform better or similarly with the original DE versions. More tests with different population sizes and iterations are required in order to provide a final conclusion. However, these preliminary results show that in terms of the original DE algorithm ranking-based mutation operators may improve significantly the results. Thus, RankDE is a potential useful algorithm for linear array synthesis.

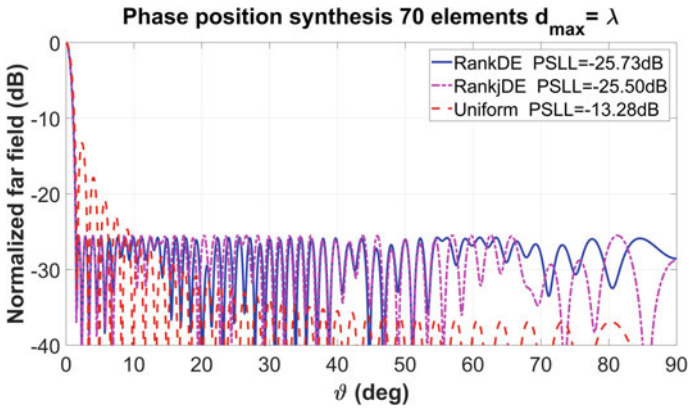


Fig. 13 Phase-Position synthesis with 70-elements $d_{\max} = \lambda$. Best radiation patterns found

Table 9 Position phase synthesis with 70-elements $d_{\max} = \lambda$. Element Positions of the best obtained result

k	x_k	$\varphi_k{}^\circ$	k	x_k	$\varphi_k{}^\circ$
1	0.258	165.262	21	11.481	165.532
2	0.760	165.110	22	12.167	164.010
3	1.266	167.873	23	12.863	163.318
4	1.776	168.917	24	13.633	158.377
5	2.301	164.579	25	14.532	162.351
6	2.821	165.592	26	15.390	161.003
7	3.346	163.519	27	16.321	165.734
8	3.859	167.540	28	17.270	177.719
9	4.412	169.118	29	18.202	154.973
10	4.933	163.228	30	19.170	182.996
11	5.515	164.436	31	20.112	212.070
12	6.041	163.743	32	21.033	90.425
13	6.616	168.818	33	22.004	155.871
14	7.177	166.046	34	22.951	148.243
15	7.738	168.526	35	23.860	174.738
16	8.384	162.443			
17	8.933	165.411			
18	9.539	165.990			
19	10.111	169.187			
20	10.755	167.385			

References

1. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**(6), 646–657 (2006). https://doi.org/10.1109/TEVC.2006.872133
2. Caorsi, S., Massa, A., Pastorino, M., Randazzo, A.: Optimization of the difference patterns for monopulse antennas by a hybrid real/integer-coded differential evolution method. IEEE Trans. Antennas Propag. **53**(1), 372–376 (2005)
3. Chen, Y., Yang, S., Nie, Z.: The application of a modified differential evolution strategy to some array pattern synthesis problems. IEEE Trans. Antennas Propag. **56**(7), 1919–1927 (2008). https://doi.org/10.1109/tap.2008.924713
4. Dib, N., Goudos, S., Muhsen, H.: Application of Taguchi's optimization method and self-adaptive differential evolution to the synthesis of linear antenna arrays. Prog. Electromagn. Res. PIER **102**, 159–180 (2010)
5. Dib, N., Goudos, S.K., Muhsen, H.: Application of Taguchi's optimization method and self-adaptive differential evolution to the synthesis of linear antenna arrays. Prog. Electromagn. Res. **102**, 159–180 (2010)
6. Doanis, P., Boursianis, A.D., Huillery, J., Bréard, A., Duroc, Y., Goudos, S.K.: Differential evolution in waveform design for wireless power transfer. Telecom **1**(2), 96–113 (2020). https://doi.org/10.3390/telecom1020008
7. Ferreira, J.A., Ares, F.: Pattern synthesis of conformal arrays by the simulated annealing technique. Electron. Lett. **33**(14), 1187–1189 (1997)
8. Gomez, N.G., Rodriguez, J.J., Melde, K.L., McNeill, K.M.: Design of low-sidelobe linear arrays with high aperture efficiency and interference nulls. IEEE Antennas Wirel. Propag. Lett. **8**, 607–610 (2009)
9. Gong, W., Cai, Z.: Differential evolution with ranking-based mutation operators. IEEE Trans. Cybern. **43**(6), 2066–2081 (2013). https://doi.org/10.1109/TCYB.2013.2239988
10. Goudos, S.: Shaped beam pattern synthesis of antenna arrays using composite differential evolution with eigenvector-based crossover operator. Int. J. Antennas Propag. **2015** (2015). https://doi.org/10.1155/2015/295012
11. Goudos, S.: Emerging evolutionary algorithms for antennas and wireless communications (2021). https://doi.org/10.1049/SBEW534E
12. Goudos, S., Siakavara, K., Samaras, T., Vafiadis, E., Sahalos, J.: Self-adaptive differential evolution applied to real-valued antenna and microwave design problems. IEEE Trans. Antennas Propag. **59**(4), 1286–1298 (2011). https://doi.org/10.1109/TAP.2011.2109678
13. Goudos, S., Siakavara, K., Samaras, T., Vafiadis, E., Sahalos, J.: Sparse linear array synthesis with multiple constraints using differential evolution with strategy adaptation. IEEE Antennas Wirel. Propag. Lett. **10**, 670–673 (2011). https://doi.org/10.1109/LAWP.2011.2161256
14. Goudos, S.K.: Design of microwave broadband absorbers using a self-adaptive differential evolution algorithm. Int. J. RF Microwave Comput. Aided Eng. **19**(3), 364–372 (2009). https://doi.org/10.1002/mmce.20357
15. Goudos, S.K.: Shaped beam pattern synthesis of antenna arrays using composite differential evolution with eigenvector-based crossover operator. Int. J. Antennas Propag. **2015** (2015). https://doi.org/10.1155/2015/295012, http://www.scopus.com/inward/record.url?eid=2-s2.0-84942346071&partnerID=40&md5=030cab52e32586d6f456f8ba23bee620
16. Goudos, S.K., Moysiadou, V., Samaras, T., Siakavara, K., Sahalos, J.N.: Application of a comprehensive learning particle swarm optimizer to unequally spaced linear array synthesis with sidelobe level suppression and null control. IEEE Antennas Wirel. Propag. Lett. **9**, 125–129 (2010)
17. Goudos, S., Zaharis, Z.D., Yioultsis, T.V.: Application of a differential evolution algorithm with strategy adaptation to the design of multi-band microwave filters for wireless communications. Prog. Electromagn. Res. **109**, 123–137 (2010)

18. Goudos, S.K., Siakavara, K., Samaras, T., Vafiadis, E.E., Sahalos, J.N.: Self-adaptive differential evolution applied to real-valued antenna and microwave design problems. IEEE Trans. Antennas Propag. **59**(4), 1286–1298 (2011)
19. Goudos, S.K., Siakavara, K., Samaras, T., Vafiadis, E.E., Sahalos, J.N.: Sparse linear array synthesis with multiple constraints using differential evolution with strategy adaptation. IEEE Antennas Wirel. Propag. Lett. **10**, 670–673 (2011)
20. Goudos, S.K., Gotsis, K.A., Siakavara, K., Vafiadis, E.E., Sahalos, J.N.: A multi-objective approach to subarrayed linear antenna arrays design based on memetic differential evolution. IEEE Trans. Antennas Propag. **61**(6), 3042–3052 (2013)
21. Goudos, S.K., Tsiflikiotis, A., Babas, D., Siakavara, K., Kalialakis, C., Karagiannidis, G.K.: Evolutionary design of a dual band e-shaped patch antenna for 5G mobile communications. In: 2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST), pp. 1–4 (2017)
22. Guo, J.L., Li, J.Y.: Pattern synthesis of conformal array antenna in the presence of platform using differential evolution algorithm. IEEE Trans. Antennas Propag. **57**(9), 2615–2621 (2009). https://doi.org/10.1109/tap.2009.2027046
23. Haupt, R.L.: Thinned arrays using genetic algorithms. IEEE Trans. Antennas Propag. **42**(7), 993–999 (1994)
24. Haupt, R.L.: Optimized weighting of uniform subarrays of unequal sizes. IEEE Trans. Antennas Propag. **55**(4), 1207–1210 (2007). https://doi.org/10.1109/tap.2007.893406
25. Hooker, J.W., Arora, R.K.: Optimal thinning levels in linear arrays. IEEE Antennas Wirel. Propag. Lett. **9**, 771–774 (2010)
26. Huang, V., Qin, A., Suganthan, P.: Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In: IEEE Congress on Evolutionary Computation. CEC 2006, pp. 17–24 (2006). https://doi.org/10.1109/CEC.2006.1688285
27. Isernia, T., Ares Pena, F.J., Bucci, O.M., D'Urso, M., Gomez, J.F., Rodriguez, J.A.: A hybrid approach for the optimal synthesis of pencil beams through array antennas. IEEE Trans. Antennas Propag. **52**(11), 2912–2918 (2004)
28. Kennedy, J.: Bare bones particle swarms. In: 2003 IEEE Swarm Intelligence Symposium, SIS 2003 - Proceedings, pp. 80–87 (2013). https://doi.org/10.1109/SIS.2003.1202251
29. Khodier, M.M., Christodoulou, C.G.: Linear array geometry synthesis with minimum sidelobe level and null control using particle swarm optimization. IEEE Trans. Antennas Propag. **53**(8 II), 2674–2679 (2005). https://doi.org/10.1109/tap.2005.851762
30. Kurup, D.G., Himdi, M., Rydberg, A.: Synthesis of uniform amplitude unequally spaced antenna arrays using the differential evolution algorithm. IEEE Trans. Antennas Propag. **51**(9), 2210–2217 (2003)
31. Li, G., Yang, S., Chen, Y., Nie, Z.: A novel electronic beam steering technique in time modulated antenna arrays. Prog. Electromagn. Res. **97**, 391–405 (2009)
32. Li, R., Xu, L., Shi, X.W., Zhang, N., Lv, Z.Q.: Improved differential evolution strategy for antenna array pattern synthesis problems. Prog. Electromagn. Res. **113**, 429–441 (2011)
33. Lin, C., Qing, A., Feng, Q.: Synthesis of unequally spaced antenna arrays by using differential evolution. IEEE Trans. Antennas Propag. **58**(8), 2553–2561 (2010). https://doi.org/10.1109/TAP.2010.2048864
34. Massa, A., Pastorino, M., Randazzo, A.: Optimization of the directivity of a monopulse antenna with a subarray weighting by a hybrid differential evolution method. IEEE Antennas Wirel. Propag. Lett. **5**(1), 155–158 (2006). https://doi.org/10.1109/lawp.2006.872435
35. Mezura-Montes, E., Velazquez-Reyes, J., Coello Coello, C.A.: A comparative study of differential evolution variants for global optimization. In: GECCO 2006 - Genetic and Evolutionary Computation Conference, 8th Annual Genetic and Evolutionary Computation Conference 2006, vol. 1, pp. 485–492 (2006)
36. Oliveri, G., Donelli, M., Massa, A.: Linear array thinning exploiting almost difference sets. IEEE Trans. Antennas Propag. **57**(12), 3800–3812 (2009)
37. Oliveri, G., Caramanica, F., Fontanari, C., Massa, A.: Rectangular thinned arrays based on McFarland difference sets. IEEE Trans. Antennas Propag. **59**(5), 1546–1552 (2011)

38. Omran, M.G.H., Engelbrecht, A.P., Salman, A.: Bare bones differential evolution. Eur. J. Oper. Res. **196**(1), 128–139 (2009)
39. Pal, S., Qu, B., Das, S., Suganthan, P.N.: Linear antenna array synthesis with constrained multi-objective differential evolution **21**, 87–111 (2010)
40. Qin, A.K., Suganthan, P.: Self-adaptive differential evolution algorithm for numerical optimization. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791 (2005). https://doi.org/10.1109/CEC.2005.1554904
41. Qin, A.K., Huang, V.L., Suganthan, P.: Differential evolution algorithm with strategy adaptation for global numerical optimization. Evol. Comput. **13**(2), 398–417 (2009). https://doi.org/10.1109/TEVC.2008.927706
42. Rocca, P., Oliveri, G., Massa, A.: Differential evolution as applied to electromagnetics. IEEE Antennas Propag. Mag. **53**(1), 38–49 (2011)
43. Storn, R.: Differential evolution research - trends and open questions. Stud. Comput. Intell. **143**, 1–31 (2008)
44. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces (1995)
45. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997). https://doi.org/10.1023/A:1008202821328
46. Wang, H., Rahnamayan, S., Sun, H., Omran, M.G.H.: Gaussian bare-bones differential evolution. IEEE Trans. Cybern. **43**(2), 634–647 (2013)
47. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evol. Comput. **15**(1), 55–66 (2011)
48. Yang, S., Gan, Y.B., Qing, A.: Sideband suppression in time-modulated linear arrays by the differential evolution algorithm. IEEE Antennas Wirel. Propag. Lett. **1**, 173–175 (2002). https://doi.org/10.1109/lawp.2002.807789
49. Yang, S., Gan, Y.B., Tan, P.K.: A new technique for power-pattern synthesis in time-modulated linear arrays. IEEE Antennas Wirel. Propag. Lett. **2**, 285–287 (2003). https://doi.org/10.1109/lawp.2003.821556
50. Yang, S., Gan, Y.B., Qing, A.: Antenna-array pattern nulling using a differential evolution algorithm. Int. J. RF Microw. Comput. Aided Eng. **14**(1), 57–63 (2004). https://doi.org/10.1002/mmce.10118
51. Zhang, J., Sanderson, A.C.: JADE: Adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)

Battle Royale Optimizer with a New Movement Strategy

Sara Akan and Taymaz Akan

Abstract Gamed-based is a new stochastic metaheuristics optimization category that is inspired by traditional or digital game genres. Unlike SI-based algorithms, individuals do not work together with the goal of defeating other individuals and winning the game. Battle royale optimizer (BRO) is a Gamed-based metaheuristic optimization algorithm that has been recently proposed for the task of continuous problems. This paper proposes a modified BRO (M-BRO) in order to improve balance between exploration and exploitation. For this matter, an additional movement operator has been used in the movement strategy. Moreover, no extra parameters are required for the proposed approach. Furthermore, the complexity of this modified algorithm is the same as the original one. Experiments are performed on a set of 19 (unimodal and multimodal) benchmark functions (CEC 2010). The proposed method has been compared with the original BRO alongside six well-known/recently proposed optimization algorithms. The results show that BRO with additional movement operator performs well to solve complex numerical optimization problems compared to the original BRO and other competitors.

Keywords Battle royale optimizer · Metaheuristic · Game-based optimization

S. Akan (✉)
Department of Computer Programing, Topkapi University, Istanbul, Turkey
e-mail: saraakan@topkapi.edu.tr

T. Akan
Department of Medicine, Louisiana State University Health Sciences, Shreveport, United States
e-mail: Taymazakan@ayvansaray.edu.tr

S. Akan · T. Akan
Faculty of Engineering, Topkapi University, Istanbul, Turkey

A. Mohamed et al. (eds.), *Handbook of Nature-Inspired Optimization Algorithms: The State of the Art*, Studies in Systems, Decision and Control 212,
https://doi.org/10.1007/978-3-031-07512-4_10

1 Introduction

Metaheuristic is a multipurpose algorithm that solves problems in an iterative manner, trying to find the best feasible option among a number of applicant solutions with a quality measure [1]. Metaheuristic optimization methods have attracted the attention of scientists and engineers from a number of disciplines, as they serve a major role both in scientific and industrial use in many practical challenges [2]. Over the last three decades, various optimization techniques have been proposed on various sources of inspiration. Battle royale optimization (BRO) [3] is a recently proposed optimization algorithm that is motivated by a digital games genre called the "royal battle". BRO is a population-based optimization algorithm that simulates the deathmatch play mode of Player Unknown's Battlegrounds (PUBG) [4]. BRO aims at solving single-objective optimization in continuous problem spaces [5]. Although several optimization algorithms have been introduced in recent years, the BRO algorithm is of particular importance among these algorithms. This is because this algorithm has opened a new horizon for metaheuristic algorithms. Before this algorithm was developed, metaheuristic algorithms consisted of three main categories: Evolutionary algorithms (EA); Swarm Intelligence (SI); and physical phenomena algorithms. But BRO introduced a new metaheuristic category called game-based.

EA-based algorithms are inspired by Darwin's theory of biological Evolution, mimicking concepts of reproduction, mutation, and selection to achieve the best possible solution. Genetic Algorithm (GA) [6], Evolution Strategies (ES) [7], Tabu Search (TS) [8], Simulated-Annealing (SA) [9], Differential Evolution (DE) [10], Biogeography-Based Optimizer (BBO) [11], Forest Optimization Algorithm (FOA) [12], etc. can be mentioned as the most well-known EA paradigms.

SI-based algorithms mimic the collective behavior of various kinds of the living being, such as humans, animals, insects, single-cells, etc., in the population. Although not every individual is intelligent alone, their collective behavior leads to an intelligent search scheme. Some of well-known and recently developed SI-based algorithms are Particle Swarm Optimization [13], Ant colony optimization [14], Cat swarm optimization [15], Artificial Bee Colony (ABC) [16], Cuckoo Search (CS) [17], Firefly Algorithm (FA) [18], Animal migration optimization [19], Dolphin echolocation (DE) [20], Chimp optimization algorithm (COA) [21], Human mental search (HMS) [22], and Selfish Herd Optimizer (SHO) [23], Group Search Optimizer (GSO) [24] as well as others.

To reach the optimal solution physics-based algorithms make use of the laws of physics such as gravitational force, inertia force, electromagnetic force, etc. The movement and communication between populations are conducted by means of these rules [25]. Examples of physics-based methods are Quantum stochastic optimization (QSO) [26], Central Force Optimization (CFO) [27], Gravitational Search Algorithm (GSA) [28], Charged System Search (CSS) [29], Black Hole (BH) [30], Optics

inspired optimization (OIO) [31], Water Evaporation Optimization (WEO) [32], Yin–Yang-Pair Optimization (YYPO) [33], Thermal exchange optimization [34], Electromagnetic Field Optimization (EFO) [35], Billiards-inspired optimization algorithm (BOA) [36], and others.

Exploration and exploitation are two critical factors in the success of an optimization algorithm. Every optimization algorithms have their own strategy for performing exploration and exploitation mechanism [28]. In order to deliver a successful outcome in an optimization algorithm, the balance between exploration and exploitation must be maintained. Exploration aims to escape from local optima by exploring among new solution-candidates in the unvisited areas of problem space. On the other hand, exploitation focuses on the area where exploitation focuses on the best solution to search more accurately. E.g., in PSO p_{best} provides exploitation alongside c_1 coefficient parameter while g_{best} provides exploration alongside c_2 coefficient parameter. Moreover, in some cases, an additional local search step is applied to enhance exploitation [37].

According to the famous mathematician Leonhard Euler, everything in nature is ultimately either minimized or maximized. Whether plants or animals, living beings have been able to adapt to the environment over time, and they found a way to survive. Nature has always seen and will continue to witness far-reaching changes; as a result, nature can be exposed to new changes and problems that must find new solutions to adapt to them. In fact, the scientific and engineering community is not unlike nature: every day, new issues may arise that at times require a new method of solution. Hence, it makes sense to be inspired by nature to develop optimization algorithms to minimize or maximize the problems. This claim could be a confirmation of the No Free Lunch (NFL) theorem [38]. NFL is a response to critics who claim that there are many optimization algorithms and that there is no need for new algorithms. The NFL proves that an optimization algorithm may not outperform others in meeting all optimization challenges. There is no heuristic algorithm yet that solves all optimization problems by providing superior performance. Therefore, it is necessary to propose new approaches as well as to improve existing ones.

BRO is a recently-proposed population-based metaheuristic optimization algorithm for continuous optimization problems that is inspired by strategies of multiplayer video game genre (Battle royale games). There are various kinds of Battle Royale games, the most important of which are Player Unknown's Battlegrounds (PUBG) [4], Call of Duty: Warzone [39], Apex Legends [40], Counter-Strike: Global Offensive [41], and Ring of Elysium [42]. However, these games are not very different from each other in game strategy. In fact, BRO mimics the solo-deathmatch mode of PUBG.

Like other stochastic algorithms, search agents randomly distributed over the problem space using a uniform distribution. BRO generates randomly N solutions using Eq. (1).

$$x_{i,d} = r(ub_d - lb_d) + lb_d \tag{1}$$

where $i = 1 \ldots N$, $d = 1 \ldots D$. N is the number of possible solutions in the population. D is the dimension of the problem space. lb_d and ub_d are the lower and the upper bounds of dth dimension, respectively. In the following, the fitness value of each solution is compared to the fitness value of the nearest neighbor with regard to the Euclidean distance. And then, the solution with the better fitness value will be declared the winner and the other the loser. Next, the loser one will be damaged, and the damage count of it will be increased by one. On the contrary, the damage of the winner will be reset to zero. The operator is expressed as: $x_{dam}.damage = x_{dam}.damage + 1$ and $x_{vic}.damage = 0$, where dam and vic are the indexes of the loser and winner solutions, respectively. Meanwhile, loser one attempts to move toward the best solution to locate the better position. The new position vector of the loser solution can be calculated by Eq. (2).

$$x_{dam,d} = x_{dam,d} + r\left(x_{best,d} - x_{dam,d}\right), \tag{2}$$

where r is a randomly-generated number from a uniform distribution [0,1] and $x_{dam,d}$ is the position of the damaged solution in dth dimension. Moreover, If a solution has a worse fitness value than his neighbor through a pre-determined number of times in a row, it will not be a solution candidate. For this purpose, the solution will be eliminated from the set of the solutions and will be replaced by a random one according to Eq. (1). It is worth noting that to provide exploitation, the problem space shrinks down towards the best solution in every Δ iteration. The initial value for Δ is calculated per $\Delta = MaxCicle/log_{10}(MaxCicle)$. Then, it will be updated through $\Delta = \Delta + round\left(\frac{\Delta}{2}\right)$. $MaxCicle$ is the maximum number of iteration herein. To put this concept lower and upper bound of each problem dimension will shrink towards the best solution as follows:

$$\begin{aligned} lb_d &= x_{best,d} - SD(\overline{x_d}) \\ ub_d &= x_{best,d} + SD(\overline{x_d}) \end{aligned} \tag{3}$$

where $SD(\overline{x_d})$ indicates the standard deviation of all solutions of the population in dth dimension and $x_{best,d}$ refers to the position of the best solution ever found.

2 A New Movement Strategy

The new movement strategy is characterized by keeping the individual motion and the expansion of the search area. In doing so, it can reflect changes in individual speed. In this case, by using an additional parameter, the individual in motion updates and maintains another parameter to use in the next iteration. This parameter is randomly generated at first and is later updated. Subsequently, its value would be closely associated with an individual search capability. The greater the value of the movement operator, the greater the individual speed will be. To do a global search, the individuals

would have a larger step. Their global capacity for optimization is strong, and their capacity for local search is weak. To do a global search, the individuals would have a larger step. Their global capacity for optimization is strong, and their capacity for local search is weak. The smaller the value of the movement operator, the smaller the size of the individual step will be. If the movement step is too large, an algorithm can fail to achieve or converge to the optimal solution, resulting in the algorithm not converging.

For the moving an individual towards the best one the Eq. (2) has been reformulated as follows:

$$\begin{aligned} \lambda_{dam,d} &= r_1 x_{best,d} + r_2\left(\lambda_{dam,d} - x_{dam,d}\right), \\ x_{dam,d} &= \lambda_{dam,d} + x_{dam,d} \end{aligned} \tag{4}$$

where r_1 and r_2 are two randomly-generated numbers from a uniform distribution [0,1] and $\lambda_{dam,d}$ is also a combination randomly-generated numbers from a uniform distribution [0,1] that will be updated through the iterations. This modified moving strategy provides more randomness in both exploration and exploitation.

3 Experimental Results and Performance Evaluation

A comprehensive experimental evaluation and comparison were performed to evaluate the performance of the BRO with a random inertia weight. Like the original BRO well-known PSO (Particle Swarm Optimization) algorithm and five recent proposed optimization algorithms: ALO (The Ant Lion Optimizer) [43], MFO (Moth-flame optimization algorithm) [44], MVO (Multi-Verse Optimizer) [45], SCA (A Sine Cosine Algorithm) [46], and WOA (The Whale Optimization Algorithm) [47] have been selected as competitors. Moreover, the original BRO has also been used in comparisons. The parameters for the control of all algorithms have been set in the initial papers according to recommendations. In addition, all algorithms were carried out and implemented in MATLAB and Conducted on a Core i7-7700 HQ 2.81 Processor with 32 GB of RAM. This paper also uses 19 well-known benchmark functions for comparison that include unimodal and multi-modal properties, like the original study. All these functions have been listed in Tables 1, 2 and 3. For every algorithm, the maximum number of iteration and population size is 500 and 100, respectively. Also, the threshold value for maximum damage is 3, and both r_1 and r_2 are randomly generated number uniformly distributed in the range [0,1]. All of the following test results are achieved by taking an average of 25 independent runs each. In addition, the median and standard variation of the fitness evaluation values are used as performance measures in 25 independent runs. Tables 4, 5 and 6 for all unimodal and multimodal test functions, have provided the experimental results.

Table 1 Unimodal benchmark test functions

Function	Name	Range	Shift position
$f_1(\mathrm{x}) = \sum_{i=1}^{n} x_i^2$	Sphere	[−100, 100]	[−30, −30,…, −30]
$f_2(\mathrm{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	Schwefel 2.20	[−10, 10]	[−3, −3,…, −3]
$f_3(\mathrm{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	Rotated hyper-ellipsoids	[−100, 100]	[−30, −30,…, −30]
$f_4(\mathrm{x}) = \max_{i=1,\ldots,n} \lvert x_i \rvert$	Schwefel 2.21	[−100, 100]	[−30, −30,…, −30]
$f_5(\mathrm{x}) = \sum_{i=1}^{n-1} \begin{bmatrix} 100\left(x_{i+1} - x_i^2\right)^2 \\ +(x_i - 1)^2 \end{bmatrix}$	Rosenbrock	[−30, 30]	[−15, −15,…, −15]
$f_6(\mathrm{x}) = \sum_{i=1}^{n} ([x_i + 00.5])^2$	Step	[−100, 100]	[−750, −750,…, −750]
$f_7(\mathrm{x}) = \sum_{i=1}^{n} i x_i^4 + rand[0, 1)$	Quartic	[−128, 128]	[−25, −25,…, −25]

Table 2 Multimodal benchmark test functions

Function	Name	Range	Shift position
$f_8(\mathrm{x}) = \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i \rvert})$	Schwefel	[−500, 500]	[−300,…, −300]
$f_9(\mathrm{x}) = 10n + \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i))$	Rstrigin	[−5.12, 5.12]	[−2,…, −2]
$f_{10}(\mathrm{x}) = -20.exp(-0,2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + exp(1)$	Ackley	[−32, 32]	
$f_{11}(\mathrm{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	Griewank	[−600, 600]	[−400,…, −400]

(continued)

Table 2 (continued)

Function	Name	Range	Shift position
$f_{12}(\mathrm{x}) = \frac{\pi}{n} \times \{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2$ $\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\} +$ $\sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \le x_i \le a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases}$	penalized	$[-50, 50]$	$[-30, \ldots, -30]$
$f_{13}(\mathrm{x}) = 0.1\{sin^2(3\pi x)$ $+ (x_i - 1)^2(1 + sin^2(3\pi y))$ $+ (x_n - 1)^2(1 + sin^2(2\pi x_n))\}$ $+ \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	Levi	$[-50, 50]$	$[-100, 100]$

Table 3 Multimodal benchmark test functions with fixed-dimension

Function	Dim	Range
$f_{14} = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right)^{-1}$	2	[−65, 65]
$f_{15} = \sum_{i=1}^{11} \left[a_i - \frac{x_1 (b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	[−5, 5]
$f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 + 4x_2^2 + 4x_2^4$	2	[−5, 5]
$f_{17} = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10\left(1 - \frac{1}{8\pi}\right) \cos(x_1) + 10$	2	[−5, 5]
$f_{18} = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2\right)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 + 36x_1 x_2 + 27x_2^2\right)\right]$	2	[−2, 2]
$f_{19} = -\sum_{i=1}^{4} c_i e^{\left(-\sum_{i=1}^{3} a_{ij} (x_j - p_{ij})^2 \right)}$	3	[−1, −3]

Table 4 proves that the modified BRO provides the best results in six out of seven unimodal functions with respect to mean and std. It is clear that there is a great improvement in the results. As it is clear, GOA performs worst among the others. Also, Table 5 shows that in five of seven multimodal functions, BRO outperforms others in terms of mean and outperforms five out of seven respect to std. As is evident from these tables, there is a great improvement in results. Such that, for F1, the mean value of the proposed method is 1.20128e-26, while for the original BRO is 3.0353e-09. Coming to the fixed-dimension multimodal functions, the modified BRO has not performed better in just one case (F14). In the rest of the function, the proposed method provides competitive results.

4 Conclusion

In this article, a modified BRO with a new movement operator has been proposed to provide a better balance between exploration and exploitation. Within the modified BRO, there are no user-set parameters, nor are kinds of other user-intervention required. At the same time, this algorithm maintains the complexity of the original one. Experiments were performed on 19 well-known (unimodal and multimodal) benchmark functions (CEC 2010). To verify the performance of the modified BRO, the original BRO, besides the well-known PSO algorithm and five recent proposed optimization algorithms ALO, MFO, MVO, SCA, and GOA, were also tested for comparison. Results indicate that the BRO with random inertia weight performs well to address complex numerical optimization problems, compared to the original BRO and other competitors.

Table 4 Results for unimodal benchmark test functions of 30 dimension

F		M-BRO	BRO	PSO	ALO	MFO	MVO	SCA	GOA
F1	Mean	**1.2012e-26**	3.0353e-09	8.131e-08	1.e-06	0.937433	0.24190	0.139829	1028.242
	Std	**1.6456e-26**	4.1348e-09	1.567e-07	1.e-06	0.547602	0.04815	0.213883	2921.282
	Time	3.598942	3.536958	1.1516	76.82223	0.5428634	0.7795	**0.213883**	1264.051
F2	Mean	**1.3350e-15**	0.000046	0.1282	20.52849	28.501319	0.29813	0.000979	77.97797
	Std	**9.8358e-16**	0.000024	0.2889	31.56666	19.846522	0.07011	0.001209	48.421113
	Time	3.290291	3.577353	1.1460	77.20241	**0.5510458**	0.71,363	0. 5,723,780	1332.949
F3	Mean	**4.2811e-08**	54.865255	103.5336	127.4139	12,432.852	13.2258	1524.3665	5095.518
	Std	**6.2916e-08**	16.117329	77.8071	43.86708	8927.4874	5.38360	1339.0129	5875.806
	Time	3.829758	3.710890	1.1226	77.78005	1.449506	1.62891	**1.436205**	1206.811
F4	Mean	**7.7175e-07**	0.518757	3.0470	8.659458	30.626058	0.52880	16.26705	1.775508
	Std	**1.4126e-06**	0.403657	1.9104	3.145536	10.680434	0.22757	9.442185	0.989782
	Time	3.834288	3.532575	1.1399	77.5049	**0.5653759**	0.74,207	0.5887966	1303.754
F5	Mean	**27.03186**	99.93684	59.0941	70.59842	1128.7042	329.447	782.05773	754.8616
	Std	**0.4024602**	82.862358	44.7770	71.72374	1327.9809	566.273	2102.6807	2748.332
	Time	5.096510	3.948993	1.2316	77.04758	**0.6677657**	0. 8423	0. 720,768	423.1110
F6	Mean	2.257257	**2.8731e-08**	9.332e-08	1.e-06	1601.2818	0.23379	4.097225	0.000003
	Std	0.3160390	**1.8423e-08**	2.255e-07	1.e-06	3741.6075	0.05323	0.599685	0.000002
	Time	4.221259e	3.551993	1.1685	77.13036	0.5411606	0.74088	5.861539	424.5775
F7	Mean	**7.8361e-04**	**0.000368**	0.0215	0.023584	2.211140	0.00798	0.021428	4.330057
	Std	**5.8031e-04**	**0.000094**	0.0097	0.00849	3.461601	0.00327	0.01812	9.45593
	Time	5.196062e	4.307858	1.5415	77.5254	**1.082169**	1.26860	1.102313	1242.990

Table 5 Results for multimodal benchmark test functions of 30 dimension

F		M-BRO	BRO	PSO	ALO	MFO	MVO	SCA	GOA
F8	Mean	**−4.5130e+03**	−7035.210	**−9.39e+31**	−5601.864	−9439.105	−7872.021	−4118.072	−6759.6244
	Std	5.3803e+02	712.3326	4.22e + 32	362.17167	933.4169	651.30106	**295.8680**	899.29158
	Time	3.871863	3.782519	1.1632	77.28404	**0. 705,079**	0.7452922	0. 732,390	1259.423
F9	Mean	**0**	48.27535	54.2471	60.851573	107.59104	108.04483	17.584379	174.76975
	Std	**0**	14.09458	**13.4287**	22.120831	24.913857	32.191667	20.95633	29.134588
	Time	3.811548	3.771521	1.2822	77.06642	**0. 621,141**	0. 8,954,284	0.635833	1270,283
F10	Mean	**3.3573e-14**	0.350724	0.9906	2.053675	6.363405	0.712974	10.13371	3.366269
	Std	**3.595e-14**	0.688712	0.8038	0.796278	8.534955	0.693826	9.537759	4.154672
	Time	3.646655	3.784324	1.2443	77.20121	**0.647817**	0.9,157,972	0.684157	1209.365
F11	Mean	**0**	**0.001373**	0.0089	0.010494	0.751211	0.479614	0.455533	7.231566
	Std	**0**	**0.010796**	0.0113	0.016548	0.179238	0.159060	0.275454	23.778806
	Time	3.84395	3.844791	1.2318	77.07877	0.735585	1.028005	0.775249	1213.085
F12	Mean	**0.130451**	0.369497	30.4105	6.933983	1.734171	0.553024	1.184349	4,323,306.37
	Std	**2.8956e-02**	0.601450	9.4299	3.807565	1.328050	0.624414	1.270975	1,616,521.88
	Time	5.694005	5.278253	1.3266	78.24491	**1.965486**	2.268537	2.036552	1214.762
F13	Mean	0.628109	**0.000004**	41.1520	0.006889	2.047761	0.032335	19.14310	18,404,721.8
	Std	0.110578	**0.000020**	9.6473	0.009962	1.909983	0.013044	71.92344	81,308,523.2
	Time	3.304954	3.303456	1.3337	78.24868	**1.986649**	2.291993	2.154423	1191.996

Table 6 Results for fixed-dimension multimodal benchmark test functions

F		M-BRO	**BRO**	**PSO**	ALO	MFO	MVO	SCA	GOA
F14	Mean	0.998004	**0.9980**	**0.9980**	1.157048	0.998004	0.998004	1.077373	1.791015
	Std	0.6285483	**0**	**0**	0.371931	**0**	4.4661e-12	0.396819	1.144130
	Time	3.565870	3.60161	2.7190	8.726093	**0.282815**	2.960769	0.289890	824.6873
F15	Mean	**04.5511e-04**	0.00047	0.0005105	0.001544	0.000748	0.000624	0.000845	0.007727
	Std	3.00125e-04	**0.00026**	0003.3323	0.003925	0.000270	0.000315	0.000398	0.008465
	Time	2.486470	2.36244	0.9858	1.140260	0.277466	0.3253819	**0. 272,637**	158.6677
F16	Mean	**−1.0316**	**−1.0316**	**−1.0316**	−1.03162	−1.03162	−1.031628	−1.031619	−1.03162
	Std	8.1149e-06	**5.995e-16**	6.798e-16	6.872e-16	6.79e-16	4.7670e-08	0.000007	7.194e-16
	Time	2.015800	1.950645	0.8377	6.266332	0. 25,199	0. 2,871,051	**0. 242,102**	79.82012
F17	Mean	**0.397887**	**0.397887**	0.3979	**0.397887**	**0.39788**	**0.397887**	0.398227	0.410236
	Std	**0**	**0**	**0**	**0**	7.42e-16	**0**	0.000260	0.064367
	Time	1.959311	1.975396	0.7929	5.986799	0.21526	0. 2,524,479	**0. 206,958**	117.7970
F18	Mean	**3.000000**	**3.000000**	**3.000000**	**3.000000**	**3.000000**	3.000001	3.000002	**3.000000**
	Std	**2.5895e-16**	**2.5895e-16**	1.344e-15	9.012e-14	**1.88e-15**	**3.3333e-07**	0.000003	0,000,101
	Time	1.979300	2.104709	0.7916	6.17052	0.20697	0.2461370	**0. 192,042**	84,81,287
F19	Mean	**−3.86208**	−3.86278	−3.8628	−3.86278	−3.8627	−3.862782	−3.856386	−3.8566
	Std	5.4290e-04	**2.006e-15**	2.266e-15	6.225e-15	2.266e-15	8.0539e-08	0.003206	0.0032
	Time	2.884435	4.461555	1.0814	9.011699	0.302513	0. 3,353,661	**0. 299,619**	167.7970

Conflict of Interest Authors declare that they have no conflict of interest.

References

1. Yu, J.J.Q., Li, V.O.K., Lam, A.Y.S.: Optimal V2G scheduling of electric vehicles and unit commitment using chemical reaction optimization. In: 2013 IEEE Congress on Evolutionary Computation, 20–23 June 2013, pp. 392–399 (2013). https://doi.org/10.1109/CEC.2013.6557596
2. Lazar, A.: Heuristic knowledge discovery for archaeological data using genetic algorithms and rough sets. In: Heuristic and Optimization for Knowledge Discovery. IGI Global, pp. 263–278 (2002)
3. Rahkar Farshi, T.: Battle royale optimization algorithm. Neural Comput. Appl. (2020). https://doi.org/10.1007/s00521-020-05004-4
4. Contributors, W.: PlayerUnknown's Battlegrounds—Wikipedia, The Free Encyclopedia (2020)
5. Agahian, S., Akan, T.: Battle royale optimizer for training multi-layer perceptron. Evol. Syst. 1–13 (2021)
6. Holland, J.: Adaptation in natural and artificial systems: an introductory analysis with application to biology, control and artificial intelligence (1975)
7. Schwefel, H.-P.: Evolution strategies: a family of non-linear optimization techniques based on imitating some principles of organic evolution. Ann. Oper. Res. **1**(2), 165–167 (1984)
8. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986). https://doi.org/10.1016/0305-0548(86)90048-1
9. Van Laarhoven, P.J., Aarts, E.H.: Simulated annealing. In: Simulated Annealing: Theory and Applications, pp. 7–15. Springer (1987)
10. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997). https://doi.org/10.1023/A:1008202821328
11. Simon, D.: Biogeography-based optimization. IEEE Trans. Evol. Comput. **12**(6), 702–713 (2008). https://doi.org/10.1109/TEVC.2008.919004
12. Ghaemi, M., Feizi-Derakhshi, M.-R.: Forest optimization algorithm. Expert Syst. Appl. **41**(15), 6676–6687 (2014). https://doi.org/10.1016/j.eswa.2014.05.009
13. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 4–6 Oct. 1995, pp. 39–43 (1995). https://doi.org/10.1109/MHS.1995.494215
14. Dorigo, M., Caro, G.D.: Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 6–9 July 1999, vol. 1472, pp. 1470–1477 (1999). https://doi.org/10.1109/CEC.1999.782657
15. Chu, S.-C., Tsai, P.-w., Pan, J.-S.: Cat swarm optimization. In: Yang, Q., Webb, G. (eds.) PRICAI 2006: Trends in Artificial Intelligence, Berlin, Heidelberg, 2006, pp. 854–858. Springer, Berlin, Heidelberg (2006)
16. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**(3), 459–471 (2007). https://doi.org/10.1007/s10898-007-9149-x
17. Yang, X.-S., Deb, S.: Cuckoo search via Lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), pp. 210–214. IEEE (2009)
18. Yang, X.-S.: Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms, pp. 169–178. Springer (2009)
19. Li, X., Zhang, J., Yin, M.: Animal migration optimization: an optimization algorithm inspired by animal migration behavior. Neural Comput. Appl. **24**(7), 1867–1877 (2014). https://doi.org/10.1007/s00521-013-1433-8

20. Kaveh, A., Farhoudi, N.: A new optimization method: dolphin echolocation. Adv. Eng. Softw. **59**, 53–70 (2013). https://doi.org/10.1016/j.advengsoft.2013.03.004
21. Khishe, M., Mosavi, M.R.: Chimp optimization algorithm. Expert Syst. Appl. **149**, 113338 (2020). https://doi.org/10.1016/j.eswa.2020.113338
22. Mousavirad, S.J., Ebrahimpour-Komleh, H.: Human mental search: a new population-based metaheuristic optimization algorithm. Appl. Intell. **47**(3), 850–887 (2017). https://doi.org/10.1007/s10489-017-0903-6
23. Fausto, F., Cuevas, E., Valdivia, A., González, A.: A global optimization algorithm inspired in the behavior of selfish herds. Biosystems **160**, 39–55 (2017). https://doi.org/10.1016/j.biosystems.2017.07.010
24. Abualigah, L.: Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. Neural Comput. Appl. (2020). https://doi.org/10.1007/s00521-020-05107-y
25. Tang, D., Dong, S., Jiang, Y., Li, H., Huang, Y.: ITGO: invasive tumor growth optimization algorithm. Appl. Soft Comput. **36**, 670–698 (2015). https://doi.org/10.1016/j.asoc.2015.07.045
26. Apolloni, B., Carvalho, C., de Falco, D.: Quantum stochastic optimization. Stoch. Process. Appl. **33**(2), 233–244 (1989). https://doi.org/10.1016/0304-4149(89)90040-9
27. Formato, R.A.: Central force optimization. Prog. Electromagn. Res. **77**, 425–491 (2007)
28. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: a gravitational search algorithm. Inf. Sci. **179**(13), 2232–2248 (2009). https://doi.org/10.1016/j.ins.2009.03.004
29. Kaveh, A., Talatahari, S.: A novel heuristic optimization method: charged system search. Acta Mech. **213**(3), 267–289 (2010). https://doi.org/10.1007/s00707-009-0270-4
30. Hatamlou, A.: Black hole: a new heuristic optimization approach for data clustering. Inf. Sci. **222**, 175–184 (2013). https://doi.org/10.1016/j.ins.2012.08.023
31. Husseinzadeh Kashan, A.: A new metaheuristic for optimization: optics inspired optimization (OIO). Comput. Oper. Res. **55**, 99–125 (2015). https://doi.org/10.1016/j.cor.2014.10.011
32. Kaveh, A., Bakhshpoori, T.: Water evaporation optimization: a novel physically inspired optimization algorithm. Comput. Struct. **167**, 69–85 (2016). https://doi.org/10.1016/j.compstruc.2016.01.008
33. Punnathanam, V., Kotecha, P.: Yin–Yang-pair optimization: a novel lightweight optimization algorithm. Eng. Appl. Artif. Intell. **54**, 62–79 (2016). https://doi.org/10.1016/j.engappai.2016.04.004
34. Kaveh, A., Dadras, A.: A novel meta-heuristic optimization algorithm: thermal exchange optimization. Adv. Eng. Softw. **110**, 69–84 (2017). https://doi.org/10.1016/j.advengsoft.2017.03.014
35. Abedinpourshotorban, H., Mariyam Shamsuddin, S., Beheshti, Z., Jawawi, D.N.A.: Electromagnetic field optimization: a physics-inspired metaheuristic optimization algorithm. Swarm Evol. Comput. **26**, 8–22 (2016). https://doi.org/10.1016/j.swevo.2015.07.002
36. Kaveh, A., Khanzadi, M., Rastegar Moghaddam, M.: Billiards-inspired optimization algorithm; a new meta-heuristic method. Structures **27**, 1722–1739 (2020). https://doi.org/10.1016/j.istruc.2020.07.058
37. Gan, C., Cao, W., Wu, M., Chen, X.: A new bat algorithm based on iterative local search and stochastic inertia weight. Expert Syst. Appl. **104**, 202–212 (2018). https://doi.org/10.1016/j.eswa.2018.03.015
38. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997). https://doi.org/10.1109/4235.585893
39. Contributors, W.: Call of duty: warzone—Wikipedia, The Free Encyclopedia (2020)
40. Contributors, W.: Apex legends—Wikipedia, The Free Encyclopedia (2020)
41. Contributors, W.: Counter-strike: global offensive—Wikipedia, The Free Encyclopedia (2020)
42. Contributors, W.: Ring of Elysium—Wikipedia, The Free Encyclopedia (2020)
43. Mirjalili, S.: The ant lion optimizer. Adv. Eng. Softw. **83**, 80–98 (2015). https://doi.org/10.1016/j.advengsoft.2015.01.010
44. Mirjalili, S.: Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl.-Based Syst. **89**, 228–249 (2015). https://doi.org/10.1016/j.knosys.2015.07.006

45. Mirjalili, S., Mirjalili, S.M., Hatamlou, A.: Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput. Appl. **27**(2), 495–513 (2016). https://doi.org/10.1007/s00521-015-1870-7
46. Mirjalili, S.: SCA: a sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. **96**, 120–133 (2016). https://doi.org/10.1016/j.knosys.2015.12.022
47. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016). https://doi.org/10.1016/j.advengsoft.2016.01.008