Women in Engineering and Science

Vanita Garg
Kusum Deep
Valentina E. Balas *Editors*

Women in Soft Computing

Springer

Women in Engineering and Science

Series Editor

Jill S. Tietjen, Greenwood Village, CO, USA

The Springer Women in Engineering and Science series highlights women's accomplishments in these critical fields. The foundational volume in the series provides a broad overview of women's multi-faceted contributions to engineering over the last century. Each subsequent volume is dedicated to illuminating women's research and achievements in key, targeted areas of contemporary engineering and science endeavors.The goal for the series is to raise awareness of the pivotal work women are undertaking in areas of keen importance to our global community.

Vanita Garg • Kusum Deep • Valentina E. Balas
Editors

Women in Soft Computing

Springer

Editors
Vanita Garg
Mathematics, School of Basic Sciences
Galgotias University
Noida, Uttar Pradesh, India

Valentina E. Balas
Aurel Vlaicu University of Arad
Arad, Arad, Romania

Kusum Deep
Department of Mathematics, Joint Faculty
MF School of Data Science and Artificial Intelligence
Indian Institute of Technology Roorkee
Roorkee, India

ISSN 2509-6427 ISSN 2509-6435 (electronic)
Women in Engineering and Science
ISBN 978-3-031-44705-1 ISBN 978-3-031-44706-8 (eBook)
https://doi.org/10.1007/978-3-031-44706-8

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

Preface

Soft computing is a wide term which consists of many methods which includes fuzzy logic, neural networks, evolutionary algorithms, etc., then the algorithms which provides solution with a tolerant of imprecision or approximation.

Complex real-world problems are always difficult to cast into well-defined problems and then solve by analytical methods. This book aims to collect the works from real world which deals with difficult problems which are unsolved by traditional methods. Our book is an attempt to promote the women in the field of soft computing which is an integral part of science and engineering.

In Chap. 1, particle swarm optimization with different mutation strategies is applied to solve real-world problems, i.e., camera calibration problem. Chapter 2 describes differential evolution algorithm for solving non-linear optimization problem. In Chap. 3, a cooperative analysis of nature-inspired algorithms is given in context of load balancing in cloud environment.

Chapter 4 introduces a novel entropy TOPSIS approach for selection of sustainable logistics center location under Pythagorean fuzzy environment.

Chapter 5 presents an improved and novel Jaya algorithm with inertia weight factor. Chapter 6 is the literature survey on drone optimization. In Chap. 7, role of metaheuristic algorithms for search results clustering is given.

This book is useful for the researchers from academia and industry to get insight into soft computing for real-world problems.

Given the caliber of the authors' contributions, the editors' hopes for this volume have been much exceeded.

The editors would like to extend their sincere gratitude to all of the book's authors, reviewers, and Springer, without whose assistance they would not have been able to maintain the book's caliber and standards.

Noida, Uttar Pradesh, India — Vanita Garg
Roorkee, Uttarakhand, India — Kusum Deep
Arad, Romania — Valentina E. Balas

Acknowledgment

I want to express my sincere gratitude to everyone who helped to make this book possible. Throughout this process, their assistance, direction, and encouragement have been priceless.

Contents

Embedded Mutation Strategies in Particle Swarm Optimization to Solve Camera Calibration Problem

Vanita Garg, Aarti Singh, and Sagar Joshi

1 Introduction

Optimization is considered as the most studied subject in the field of real-life applications. Every decision-making criterion is based on optimization techniques. The optimization problem is often separated into categories termed local optimization and global optimization.

The objective of local optimization is to identify the greatest or lowest value within a limited area of the function value space. Finding the highest or lowest value across the entire region of the function value space is the goal of the global optimization.

Without loss of generality, a single-objective optimization problem is formulated as follows:

$$\min x \epsilon s \ f(x), x = L_i \leq x_i \leq U_i$$

$$s.t.g_i \ (x) \leq 0, j = 1, 2, \ldots, j$$

$$h_k \ (x) = 0, k = 1, 2 \ldots ., k$$

where

$f(x)$ is objective function

V. Garg (✉) · A. Singh · S. Joshi
Division of Mathematics, SBAS, Galgotias University, Greater Noida, Uttar Pradesh, India
e-mail: aarti_singh.sbasbsc@galgotiasuniversity.edu.in;
sagar_joshi.sbasbsc@galgotiasuniversity.edu.in

V. Garg et al. (eds.), *Women in Soft Computing*, Women in Engineering and Science,
https://doi.org/10.1007/978-3-031-44706-8_1

x is D-dimensional decision vector
j shows the number of inequality constraints
k indicates the number of equality constraints, and L_i, U_i are the lower and upper bound of the ith variables, respectively.

The objective and constraint functions might be explicit or implicit, and they can be linear or nonlinear. Any of the side constraints will never be violated by a good algorithm. There are no equality or inequality constraints in an unconstrained optimization problem, although there may be side constraints. Constrained optimization problem has one or more equality and or inequality constraints.

The technique requires determining the design variable values that produce the best objective function value while satisfying all equality, inequality, and side constraints. It is worth noting that for many issues, multiple optimums (also known as local or relative optima) may occur.

Over the time period, methods of optimization have evolved. There are many traditional and conventional method of optimization available in the literature. The traditional methods require the conditions of convexity and differentiability of the functions. However, in real-world problems, there is no well-behaved functions. Thus, it gives rise to nature-inspired optimization techniques.

Garg and Deep [4] have given a detailed review on biogeography-based optimization algorithm. Garg and Deep [3, 7] have proposed LX-BBO and tested on various benchmark functions. Garg and Deep [3, 7] have proposed LX-BBO and embedded mutation strategies. Garg and Deep [5] have extended LX-BBO for constrained optimization problems.

Moreover, Garg and Deep [6] have applied the constrained version of LX-BBO on portfolio optimization problem. Garg et al. (2022) have applied constrained LX-BBO for economic load dispatch problems.

The paper is organized as follows: Section 2 gives a brief introduction to PSO and proposed versions of PSO. Section 3 gives the introduction of the camera model for image formulation and the mathematics of the camera calibration problem for formulating the optimization problem. Section 4 gives the solution and numerical analysis of the camera calibration problem using all the six versions of PSO proposed and Blended BBO. Section 5 gives the conclusion and some future ideas are suggested.

2 Proposed Versions of Particle Swarm Optimization

Nature, which has evolved over a billion of years, provides a great source of inspiration and curiosity among academics from numerous fields to build a diverse spectrum of nature-inspired optimization algorithm. The majority of real-world application include optimization problem with a large number of nonlinear constraints.

Traditional optimization methods like gradient-based methods are not up to the task. Consequently, nature-inspired optimization methods are employed to handle the highly nonlinear issues. Nature-inspired algorithms have a high success rate because of their adaptability and capacity to solve NP-hard tasks. Birds, insects, fish, ants, and lions have all been used as inspiration for the development of these algorithms.

Nature-inspired algorithm can be classified into two broad categories: (1) evolutionary algorithm and (2) swarm-based algorithm.

PSO (particle swarm optimization) is considered to be one of the most popular among them. The details of the algorithm are given as follows.

2.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is a swarm foregoing behavior-based optimization technique. PSO is given by Kennedy and Eberhart in 1995. Fishes or birds collect their food using the social and individual skills. This phenomenon gives rise to a stochastic optimization algorithm which is popularly known as PSO. The natural tendency to find the best solution among the given choices is formulated mathematically using two update equations, which are given as follows.

Velocity Update Equation
Swarm of particles move in multidimensional search space to find the optimised solution. Each particle in swarm is affected by its neighbor and individual understanding. The velocity by which the particle moves from one place to another is given in the following equation:

$$v_{t+1} = v_t + C_1 * \text{rand}\,(\text{Pbest} - x_t) + C_2 * \text{rand} * (\text{Gbest} - x_t)$$

where

v_t = velocity of x particle at tth iteration
C_1 = the individual factor
C_1 = social factor
Pbest = the individual best particle
Gbest = the global best particle
Rand is the uniform random number between 0 and 1.

Particle Update Equation
The direction a particle is traveling in quest of an optimal path is determined by its velocity. The following is the particle update equation for PSO:

$$x_{t+1} = x_t + v_{t+1}$$

The solution will be updated as a result of these two equations, which will be repeated until a predetermined termination threshold is met.

PSO is a population-based strategy that includes multiple candidate solutions in the solution optimization process. Every solution is driven by the local and global top results throughout the whole search area. Thus, it strikes a perfect balance between exploration and exploitation for a robust algorithm.

2.2 Literature Review of PSO

A popular optimization method that has been altered throughout time is particle swarm optimization. PSO is also available in a variety of hybrid forms in the literature. PSO has also been used to solve a variety of actual optimization issues. To comprehend the mechanism of PSO, a brief examination of the relevant literature is required.

In Shi and Eberhart [23], a modified PSO is proposed in which a new parameter inertia weight is introduced. This parameter has proved to be a winner addition to PSO and has been used since then in many of the modified versions of PSO.

Kennedy and Eberhart [22] have proposed a discrete binary version of PSO. It is an initial attempt to solve discrete optimization problems. Many real-life problems have been solved using this version of PSO.

In Liang et al. [1], PSO is used with comprehensive learning to solve multimodal optimization. Multimodal problems most of the time have a drawback to trap into local optima. PSO for multimodal problems has solved these problems convincingly.

In Robinson and Rahmat-samii [2], PSO is used in electromagnetics. It is applied to solve antenna design problems as well.

In Trelea [8] convergence analysis and parameter selection of PSO has been studied where exploration and exploitation of the search space in PSO is also discussed. Convergence analysis gives the idea if the algorithm is approaching toward the best solution through iterations.

Eberhart and Kennedy [21] have given a detailed discussion about the comparison of GA and particle optimization. Both algorithms are population-based techniques and are considered to be the best in solving complex optimization problems. However, these techniques are different in structure. GA is known as an evolutionary algorithm and PSO is a swarm intelligence-based technique. Solutions tend to improve in PSO and in GA new solutions take place of dead or worse solution.

Parsopoulos and Vrahatis [9] have given PSO for various approaches like integer programming problems, tackling minimax problems, and finding multiple minimizers. A composite PSO is also given in which DE approach has been incorporated. Sun et al. [10] have studied individual particle of PSO using quantum behavior.

In Bonyadi and Michalewitz [11], a detailed review of PSO is given for single-objective continuous optimization problems. In this article, the limitations of PSO

in terms of local optimal solution and low convergence rate are discussed. These issues are very common in heuristic approaches which needs to be addressed so that PSO can be applied to a wide set of real optimization problems.

PSO is used to optimize the weights used in neural networks as a parameter. Kennedy [12] is an excellent work which gives detailed examples and applications of PSO and its limitations as well. Poli [13] has a brief study of applications of PSO.

Shi and Eberhart [23] analyzed the parameter inertia weight and its effect on PSO. They have given a detailed set of guidelines for how to select inertia weight and maximum velocity.

Eberhart and Shi [25] have given the comparative performance of PSO using inertia weight and constriction factor. The performance is tested on five benchmarks, and the comparison concludes that using constriction factor gives better results with a dynamic velocity range.

Carlisle and Dozier [14] have proposed an off the shelf PSO. Clerc and Kennedy [15] have analyzed particle's path in discrete time and then in continuous time. Trela [8] has discussed the performance of PSO in context to its convergence behavior and parameter selection.

Evers [16] has put light on a very important aspect of PSO. The stagnation in PSO is harmful to finding the global optimal solution. An automatic regrouping mechanism is proposed to overcome stagnation and to avoid local optimal solution.

Kennedy and Mendes [26] have discussed population structure in PSO. All nature-inspired optimization techniques deal with set of solutions instead of a single solution. Hence, the idea of the population in PSO is considered in this paper. Suganthan [24] discussed PSO with a neighbor operator.

Yin et al. [17] have introduced a complementary cyber swarm algorithm.

Liu has produced another analysis of PSO on order 2 stability. It brings the theoretical aspects of PSO which proves that PSO is not only a metaheuristic algorithm but also works well in theoretical aspect of a robust algorithm.

Kennedy and Eberhart [22] have used PSO for discrete value problems using binary coding. In Chen et al., PSO is used for discrete set of problems. Jarboui et al. [18] have allied PSO to combinatorial problems. Clerc [19] has given PSO for discrete PSO.

Xinchao [20] has proposed a perturbed PSO to find the global optima. Lovbjerg and Krink [27] have extended PSO with self-organizing criticality.

2.3 Version of PSO Using Mutation Strategies

One of the fundamental and initial optimization techniques inspired by nature is the genetic algorithm. The mutation operator is regarded as the primary operator in genetic algorithms to improve the algorithm's exploratory nature. In this study, we looked into the incorporation of the GA mutation operator into PSO. Real programmed mutation operators were successfully applied by Garg and Deep considering the earlier work's improved performance of mutation operators. The

mutation operators Cauchy, Gaussian, polynomial, random, and power are being embedded into the particle swarm optimization algorithm. The following are the five different mutation operators.

PSO with Power Mutation (PM-PSO)
PSO is imbedded with the power mutation reported by Garg and Deep [3]. Referred to as power mutation, its distribution function is given by:

$$f(x) = px^{p-1}, 0 \leq x \leq 1 \tag{1}$$

And the density function is presented by:

$$F(x) = x^{p}, 0 \leq x \leq 1 \tag{2}$$

The index of the distribution is denoted by p. The PM is used to generate a solution y near a parent solution z that follows the previously mentioned distribution. The mutated solution is then created using the following formula:

$$y = \begin{cases} x - z\,(x - x_l) & \text{if } r < t \\ x - z\,(x_u - x) & \text{if } r \geq t \end{cases} \tag{3}$$

where $t = \frac{x - x_l}{x_u - x_l}$ and $x_i$and x_u are lower bound value and upper bound value of the decision variable respectively and r is a random number uniformly distributed between zero and one. The power distribution function $p = 0.25$ & $p = 0.50$. The probability of mutation used in PM-PSO is 0.5.

PSO with Polynomial Mutation (Poly-PSO)
A new version of PSO proposed the name Poly-PSO that incorporates in PSO by Garg and Deep [3]. Using power mutation on the provided solution $x \epsilon [x_l, x_u]$, mutated solution x is constructed.

$$x' = \begin{cases} x + \delta_l\,(x - x_l) & \text{if } u \leq 0.5 \\ x + \delta_r\,(x_u - x) & \text{if } u > 0.5 \end{cases},$$

where $u \epsilon [0, 1]$ is a random number. The values δ_l and δ_r are computed as given by the formula:

$$\delta_l = (2u)^{\frac{1}{1+\rho m}} - 1, u \leq 0.5$$

$$\delta_r = 1 - (2\,(1 - u))^{\frac{1}{1+\rho m}}, u > 0.5$$

where $\rho_m \in$ [20,100] is a user-defined parameter.

PSO with Random Mutation (Rand-PSO)
A new version is suggested called Rand PSO which is proposed by integrated Rand mutation in PSO. Suppose x is any given solution than a rand mutation operator is used as $x \in [x_l, x_u]$, and a random solution h is created using a neighbourhood of the replaced solution.

$$h = x_l + (x_u - x_l) * \text{rand}$$

where rand$\in$[0, 1] represents a uniform distribution

PSO with Gaussian Mutation (G-PSO)
Gaussian mutation causes a small random change in the population. A random number from Gaussian distribution N (0,1) with parameter 0 as a mean and 1 as std dev. is generated.

PSO with Cauchy Mutation(C-PSO)
Cauchy mutation is defined in SCA as the same way as G-SCA. Suppose a random number is generated from Cauchy distribution and defined by $\delta_i(t)$. The scale parameter is represented by t, where $t > 0$. Consider the value $t = 1$ as used in LX C-BBO in this paper.

3 Mathematical Formulation of Stereo Camera Calibration Problem

In order to formulate the camera calibration problem mathematically, there are two procedures to follow. First, the camera model for image generation is defined. Second, camera calibration problem is defined as a nonlinear optimization problem.

3.1 Camera Model for Image Formation

Using a pinhole camera model for binocular vision, the geometry of image generation is defined in this study (stereo camera system).

We consider world reference frame (X, Y, Z), the camera frame (X_i, Y_i, Z_i), and the image frame (u_i, v_i) for $i = 1$ and 2 the five coordinate frames. O1 and O2 are the cameras' origins in terms of coordinate systems, and they coincide with their respective optical centers, while their Z coordinate axes are parallel to and intersect their respective picture planes' respective optical axes at their respective principal points. Each camera's picture plane and optical center are spaced apart by a factor of $i = 1, 2$ called fi (the focal length) (see Fig. 1).

An object point P, whose 3D world reference coordinates are (X, Y, Z). Four steps are involved in the conversion of its world coordinates into image coordinates:

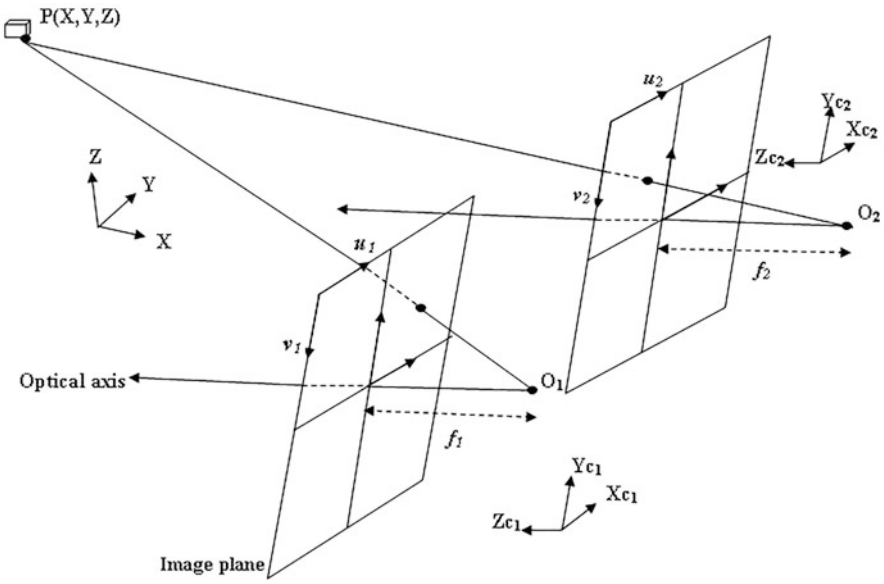


Fig. 1 Image formation by two cameras with pinhole model (Reproduced from Garg and Deep [3])

1. Applying the following relation, the three-dimensional world coordinates are first converted to camera coordinates:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = R \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + T \tag{4}$$

A 3×3 rotation matrix, R, and a translation vector, T, represent the camera's position in relation to the world coordinate system.

R and T can also be written as:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad T = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} \tag{5}$$

It is also possible to represent the matrix R's elements r_{ij} in terms of the angles of swing, tilt, and pan (α, β, γ) as:

$$R = \begin{bmatrix} \cos\alpha\cos\beta & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma \\ -\sin\alpha\cos\beta & \cos\alpha\cos\gamma - \cos\alpha\sin\beta\sin\gamma & \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma \\ \sin\beta & -\cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{6}$$

2. Perspective projection equations are used to convert the 3D camera coordinates (X_c, Y_c, Z_c) into the ideal retinal coordinates (X_u, Y_u) of the camera:

$$X_u = f\frac{X_c}{Z_c}, \quad Y_u = f\frac{Y_c}{Z_c} \tag{7}$$

3. It's therefore possible to turn these ideal retinal coordinates into actual retinal coordinates (X_d, Y_c) by taking into account lens distortion coefficient:

$$X_d = X_u\left(1 + kr^2\right)^{-1} \text{ and } Y_d = Y_u\left(1 + kr^2\right)^{-1} \tag{8}$$

4. As a last step, the retinal coordinates are converted into pixel coordinates (u, v) using the following relation:

$$u = X_d N_x + u_0 \quad v = Y_d N_x + v_0 \tag{9}$$

N_x and N_y represent the number of pixels that are contained in a unit distance of the image plane along the X and Y axes, respectively, in (u_0, v_0), the image center. Images of a 3D point can be formed by following the four processes outlined above:

$$u = \frac{f N_x}{\left(1 + kr^2\right)}\left(\frac{r_{11}X + r_{12}Y + r_{13}Z + T_x}{r_{31}X + r_{32}Y + r_{33}Z + T_z}\right) + u_0 \tag{10}$$

$$v = \frac{f N_y}{\left(1 + kr^2\right)}\left(\frac{r_{21}X + r_{22}Y + r_{23}Z + T_y}{r_{31}X + r_{32}Y + r_{33}Z + T_z}\right) + v_0 \tag{11}$$

Equations (10) and (11) represent the conversion of a single camera's 3D world coordinates (X, Y, and Z) to pixel coordinates (u, v). Point P's 3D position cannot be reconstructed using only its pixel data since this procedure is irreversible (u, v). It is possible to discover the exact location of a point P(X, Y, Z) in 3D space by comparing the stereo images $p_l(u_l, v_l)$ and $p_r(u_r, v_r)$ captured by two distinct cameras (one on each side) or by a single camera in two different locations.

3.2 Camera Calibration Problem as an Optimization Problem

This is the 3D world coordinates (X, Y, and Z) to pixel coordinates conversion for a single camera (u, v). Using only pixel data, it is impossible to rebuild the 3D position of point P because this method is irreversible (u, v). The precise location of a point $P(X, Y, Z)$ in 3D space can be determined by comparing the stereo pictures $p_l(u_l, v_l)$ and $p_r(u_r, v_r)$ taken by two separate cameras (one on each side) or by a single camera at two different places.

The problem is to find the best values for the 24 camera parameters (12 parameters ($f, N_x, N_y, u_0, v_0, k, T_x, T_y, \alpha, \beta, \gamma$ for each camera) so that the objective function given by the following equation is minimized:

$$F = \sqrt{\frac{1}{N}\sum_{i}^{N}\left[\left(u_{il}-u'_{il}\right)^2+\left(u_{ir}-u'_{ir}\right)^2+\left(v_{il}-v'_{il}\right)^2+\left(v_{ir}-v'_{ir}\right)^2\right]} \tag{12}$$

where $p'_{lj}(u'_{lj}, v'_{lj})$ and $p'_{rj}(u'_{rj}, v'_{rj})$ are using the predicted parameters; the corresponding pixel coordinates were determined. Based on the camera's understanding, any relevant search range for variables can be selected. For instance, it would be a difficult task to find the global minimum of the objective function with all 24 parameters changing.

4 Numerical Analysis of the Solution by Proposed Method

To solve the camera calibration problem, a vector S is defined which consists of unknown intrinsic and extrinsic parameters of left and right cameras respectively.

$$\begin{aligned} S = \big(&u_{0l}, v_{0l}, N_{xl}, N_{yl}, f_l, k_l, \alpha_l, \beta_l, \gamma_l, T_{xl}, T_{yl}, T_{zl}, u_{0r}, v_{0r}, N_{xr}, N_{yr}, \\ &f_r, k_r, \alpha_r, \beta_r, \gamma_r, T_{xr}, T_{yr}, T_{zr}\big) \end{aligned} \tag{13}$$

where T_x, T_y, T_z, α, β, and γ are the extrinsic parameters which denote the location and direction of the camera in a world coordinate system (WCS). (u_0, v_0) Image center, (f) focal length, (k) radial lens distortion, and (N_x and N_y) off-axis lens distortion are intrinsic parameters which represent camera optical and geometry features. Extrinsic parameters give the relationship between world coordinate system (WCS) and camera coordinate system (CCS). Then internal parameters can be used to set the relationship between the coordinate systems, i.e., camera coordinate system (CCS) and ideal coordinate system.

Camera calibration is an unconstrained, nonlinear optimization problem, where objective function is given by Eq. (12) and decision variables or independent variables are given by Eq. (13).

Camera calibration problem is solved by all the variants of Laplacian biogeography-based optimization (LX-BBO) proposed in Chaps. 2 and 3 with varying numbers of control points.

Camera mistakes are used to measure calibration accuracy. It is defined as the Euclidean distance between ground truth and the estimated value of a parameter by the camera. "Ground truth values," or the values utilized to generate data, are the intrinsic and extrinsic parameters of the left and right cameras that are used to calculate 2D picture coordinates from 3D grid points. Ground truth values and parameter bounds are shown in Table 1.

4.1 Generation of Synthetic Data

A calibration chart with 8 × 8 grids in the X and Y axes created the synthetic data. This generates 64 grid points. This graph has been shifted eight times in the Z direction to produce a 3D space with 512 points $P_i(X_i, Y_i, Z_i)$.

4.2 Parameter Settings

The parameters for all the algorithms are set as in Garg and Deep [3]. However, the termination criterion for this problem is set according to two conditions: when the maximum iterations (5000) are reached and when the absolute error value is less than 10^{-6}. Here absolute value is referred to the pixel error which is considered as the objective function itself. Out of these two conditions, whichever is attained earlier become the required termination criteria.

Table 1 Ground truth value of left and right camera

Parameter	Left camera		Right camera	
	Ground truth value	Bounds	Ground truth value	bounds
f	10	[5,15]	25	[20,40]
N_x	200	[170,230]	144	[100,200]
N_y	200	[170,230]	144	[100,200]
u_0	20	[15,25]	256	[200,300]
v_0	19	[15,25]	192	[150,250]
k	0.15152	[0,0.5]	0.15152	[0,0.5]
T_x	60	[20,80]	−38	[−100,80]
T_y	35	[25,45]	35	[25,45]
T_z	1210	[1000,1400]	1210	[1000,1400]
α	0	[−π/2,π/2]	−1.90146	[−π,π]
β	0	[−π/2,π/2]	0.20916	[−π/2,π/2]
γ	0	[−π/2,π/2]	0.15152	[−π/2,π/2]

4.3 Experimental Results

The performance of all six versions of PSO on camera calibration problem is compared with the performance of Blended BBO and PSO. The results of minimum camera errors of 50 independent runs for different number of control points are shown in Tables 2 and 3, respectively.

Simulations are performed on varying number of control points: 10, 50, 100, 200, and 500. When the number of control point is 10, PSO gives value of six camera parameters which are nearer to their ground truth value. For 50 control points, LX-L PSO gives four camera parameters nearer to ground truth value than the other variants of LX-PSO. LX-L PSO gives better camera error values for six camera parameters than the other variants of LX-PSO.

LX-Rand BBO gives better camera error than the other variants in three parameters when the number of control points is 100. Whereas when the control points are 100, LX-L PSO gives minimum camera error in six camera parameters. When the number of control points is 200, then five camera parameters are showing better camera error using LX-G PSO. LX-L PSO behaves better than all other variants of LX PSO when the number of control parameters is 500.

Tables 2 and 3 give the analysis based on results obtained by all the variants of LX-PSO. It gives the number of parameters which have optimal value obtained by all the variants of LX-PSO.

PSO has obtained the minimum camera error than all the variants of LX-PSO for all control points. LX-L PSO has proved to be the better algorithm than the other version of LX-PSO for camera calibration problem. LX-PSO with random mutation which is proved to be the best algorithm in Chap. 3 is not able to produce good results for this problem.

Rand PSO and C PSO have the best results, but this doesn't mean that the other types of PSO aren't also important. Depending on the difficulty of the optimization problem, any version can be shown to be useful.

In Fig. 2, convergence graphs of all the variants of PSO and Blended BBO are shown for camera calibration problem. Horizontal axis is the number of generations and vertical axis is the objective function value. Here, objective function value of camera calibration problem is the pixel error obtained. All the algorithms are performed with same initial population to make a fair comparison. It can be observed that Blended BBO has the poorest convergence as compared to other variants of PSO. L PSO has better convergence than all the other variants of PSO.

Table 2 Camera errors for left camera for varying numbers of control points

Parameter	Algorithm	10	50	100	200	500
f	Rand PSO	0.076868	0.024334	0.157513	0.124538	0.024437
N_x	PM PSO	0.0344981	0.434759	0.121133	0.046615	0.047882
	Poly PSO	0.04324238	0.008042	0.028207	0.01648	0.07599
	G PSO	0.048427	0.103863	0.144257	0.069504	0.006694
	C PSO	0.0030938	0.092509	0.007664	0.187033	0.139281
	L-PSO	0.0279277	0.04823	0.069935	0.022762	0.052066
	Blended BBO	0.0322	0.02618	0.00495	0.00671	0.02101
	Rand PSO	1.67	0.385597	0.382618	0.548318	0.730356
N_y	PM PSO	0.29898	1.159758	0.059637	0.998098	0.31041
	Poly PSO	0.08727	1.39171	0.071564	1.197718	0.372492
	G PSO	0.87745	1.670052	0.085877	1.437261	0.44699
	C PSO	0.5623	2.004062	0.103053	1.724713	0.536388
	L-PSO	0.07651	2.404874	0.123663	2.069656	0.643666
	Blended BBO	0.3934	2.885849	0.148396	2.483587	0.772399
	Rand PSO	2.3093973	11.82527	1.319943	1.9135	0.369863
u_0	PM PSO	0.387762	30.39693	8.714069	19.66042	22.65397
	Poly PSO	4.893	0.975087	18.41438	1.575908	38.95181
	G PSO	1.838734	0.683595	2.605886	1.110187	2.891485
	C PSO	10.3938	8.050961	28.01932	6.066603	0.534862
	L-PSO	0.873	0.490529	6.037532	5.533563	4.518829
	Blended BBO	0.40837	0.17199	0.23465	0.16406	0.16835
	LX-Rand BBO	0.0292	0.031997	0.256192	0.18951	0.073873
v_0	LX-PM BBO	0.04847	0.2431	0.061802	0.177463	0.337853
	LX-Poly BBO	0.1983	0.21491	0.404932	0.479892	0.068281
	LX-G BBO	0.09837	0.128807	0.052503	0.354757	0.940733
	LX-C BBO	0.112938	0.024072	0.040034	0.05118	0.582759
	LX-l BBO	0.0178398	0.000705	0.014607	0.429354	0.280561
	Blended BBO	0.08989	0.06981	0.00988	0.02119	0.00104
	Rand PSO	1.34265	0.741057	0.682209	1.015719	4.175725
k	PM PSO	0.3357	4.119933	1.733164	0.481672	0.086142
	Poly PSO	0.4598	0.805775	0.251563	2.922266	3.639468
	G PSO	0.30877	0.065042	0.004492	0.007795	0.040118
	C PSO	0.6988	0.392656	1.841579	0.570943	0.191385
	L-PSO	0.9766	0.017161	0.03813	0.64016	0.021408
	Blended BBO	0.018927	0.13884	0.02613	0.17277	0.07241
	Rand PSO	0.0012892	0.007537	0.005751	0.022836	0.003475
	PM PSO	0.00824	0.001756	0.002176	0.007588	0.012832
	Poly PSO	0.002872	0.00354	0.001167	0.001895	0.008437
	G PSO	0.002988	0.001249	0.022905	0.005643	0.000103
	C PSO	0.00092761	0.005876	0.004667	0.03083	0.033955
	L-PSO	0.00792	0.006664	1.17E-05	0.029757	0.003078
	Blended BBO	0.004982	0.01027	0.01677	0.01729	0.00156

(continued)

Table 2 (continued)

Parameter	Algorithm	10	50	100	200	500
T_x	Rand PSO	0.11972	0.026052	0.297564	0.864591	0.066409
T_y	PM PSO	0.59272	0.065632	0.63135	0.598079	0.020628
	Poly PSO	0.735091	1.010329	0.371582	0.435924	0.032583
	G PSO	0.073639	1.21052	0.646933	0.530257	0.029168
	C PSO	0.1938	0.221803	0.260677	0.222334	0.104469
	L-PSO	0.31838	1.00999	0.167359	0.351686	0.112323
	Blended BBO	0.6583	0.66404	1.00126	0.89232	0.23205
	Rand PSO	1.23983	5.142888	0.790665	0.912582	2.035849
T_z	PM PSO	7.40292	9.213382	2.426164	2.357414	0.510078
	Poly PSO	1.89202	8.029741	1.62026	4.407579	0.419929
	G PSO	0.076939	0.72936	0.941844	1.703283	0.935087
	C PSO	6.9802	5.560166	7.603574	4.731641	0.104048
	L-PSO	0.2639839	0.394265	0.576217	0.488848	0.505852
	Blended BBO	0.1527	1.312987	0.217567	0.457192	0.14602
	Rand PSO	0.67292	0.863252	1.301638	1.160016	0.301665
alpha	PM PSO	1.40292	6.685754	1.027865	1.186357	2.646604
	Poly PSO	0.0919187	11.9774	3.154013	3.064638	0.663101
	G PSO	3.9282	10.43866	2.106338	5.729853	0.545908
	C PSO	6.3938	0.948168	1.224397	2.214268	1.215613
	L-PSO	0.18484	7.228216	9.884646	6.151133	0.135262
	Blended BBO	0.093983	0.512545	0.749082	0.635502	0.657608
	Rand PSO	0.01383	1.706883	0.282837	0.594349	0.189826
beta	PM PSO	0.012928	1.122228	1.692129	1.508021	0.392165
	Poly PSO	0.000084749	8.691481	1.336224	1.542264	3.440585
	G PSO	0.000017	15.57062	4.100217	3.98403	0.862032
	C PSO	0.030038	13.57026	2.738239	7.448809	0.70968
	L-PSO	0.024849	1.232618	1.591716	2.878548	1.580297
	Blended BBO	0.000298	9.396681	12.85004	7.996473	0.175841
	Rand PSO	0.073332	0.666308	0.973807	0.826153	0.85489
gamma	PM PSO	0.047565	2.218948	0.367688	0.772654	0.246774
	Poly PSO	0.007292	1.458896	2.199768	1.960427	0.509814
	G PSO	0.045428	11.29892	1.737091	2.004943	4.47276
	C PSO	0.0375675	20.2418	5.330282	5.179239	1.120641
	L-PSO	0.01284575	1.458896	2.199768	1.960427	0.509814
	Blended BBO	0.0059	11.29892	1.737091	2.004943	4.47276
	Rand PSO	0.245493	20.2418	5.330282	5.179239	1.120641
	PM PSO	0.0158373	17.64134	3.559711	9.683451	0.922584
	Poly PSO	0.0013847	1.602404	2.069231	3.742113	2.054386
	G PSO	0.00001987	12.21568	16.70505	10.39542	0.228593
	C PSO	0.008387	0.8662	1.265949	1.073999	1.111357
	L-PSO	0.03629874	2.884632	0.477994	1.00445	0.320806
	Blended BBO	0.00598	1.896565	2.859699	2.548555	0.662758

Table 3 Camera errors for right camera for varying numbers of control points

Parameter	Algorithm	10	50	100	200	500	Average
f	Rand PSO	0.044406	0.18	0.11172	0.052761	0.043144	0.0864062
N_x	PM PSO	0.074132	0.079752	0.080253	0.106694	0.014889	0.071144
	Poly PSO	0.066342	0.159004	0.007878	0.149157	0.101876	0.0968514
	G PSO	0.05697	0.056588	0.025016	0.100576	0.022319	0.0522938
	C PSO	0.034158	0.139876	0.077253	0.18627	0.059315	0.0993744
	L-PSO	0.478059	0.045168	0.015363	0.001168	0.162348	0.1404212
	Blended BBO	0.31616	0.33453	0.133498	0.000769	0.271253	0.211242
	Rand PSO	1.373803	0.044301	0.197815	0.164958	0.129119	0.3819992
N_y	PM PSO	0.004764	0.084025	0.976119	0.815793	1.198443	0.6158288
	Poly PSO	0.477428	0.100776	0.172758	0.487176	0.06959	0.2615456
	G PSO	0.312556	0.41266	0.231014	0.113208	0.299791	0.2738458
	C PSO	0.262216	0.036426	0.088562	0.241455	1.205115	0.3667548
	L-PSO	2.27699	0.219725	1.106474	0.240857	0.25447	0.8197032
	Blended BBO	0.26576	0.00432	0.424428	0.518146	0.122522	0.2670352
	Rand PSO	0.193095	1.06913	0.317453	1.553951	0.387519	0.7042296
u_0	PM PSO	0.42701	0.883951	0.844218	1.545705	0.079553	0.7560874
	Poly PSO	0.520066	1.239188	1.031169	0.595897	0.628988	0.8030616
	G PSO	0.020741	0.352441	0.316507	0.672235	0.914089	0.4552026
	C PSO	0.188418	0.128667	1.190323	0.398805	0.923642	0.565971
	L-PSO	1.417966	0.475952	0.121807	0.99797	1.566333	0.9160056
	Blended BBO	0.19728	0.18135	0.416902	0.346858	0.05602	0.239682
	LX-Rand BBO	0.967718	0.477842	0.065884	2.642091	3.782402	1.5871874
v_0	LX-PM BBO	0.020769	0.066174	0.429793	0.173863	1.01068	0.3402558
	LX-Poly BBO	0.382444	0.56381	0.460645	2.242422	0.881862	0.9062366
	LX-G BBO	0.151233	1.192916	0.674603	1.118233	0.143079	0.6560128
	LX-C BBO	0.353092	1.189723	0.197002	0.611511	0.321765	0.5346186
	LX-l BBO	1.684685	0.113715	1.654228	0.499089	1.230732	1.0364898
	Blended BBO	0.26704	0.32139	0.213977	0.30217	0.494239	0.3197632
	Rand PSO	0.189348	0.243143	0.283194	0.667557	0.206054	0.3178592
k	PM PSO	0.711386	0.960224	0.126378	0.057336	0.228718	0.4168084
	Poly PSO	0.311976	0.07163	0.354237	1.536039	0.121138	0.479004
	G PSO	1.030991	0.158625	143.2231	0.861066	0.315927	29.117942
	C PSO	0.306065	0.619149	0.774449	0.278219	1.236844	0.6429452
	L-PSO	0.535554	0.719699	0.148611	0.021479	0.34391	0.3538506
	Blended BBO	0.15408	0.15939	0.044477	0.524075	0.821402	0.3406848
	Rand PSO	0.001876	0.000481	0.000738	0.006913	0.001189	0.0022394
	PM PSO	0.004312	0.001092	0.005227	0.012322	0.000652	0.004721
	Poly PSO	0.002155	0.006408	0.001389	0.000578	0.005187	0.0031434
	G PSO	0.000886	0.00783	0.00995	0.000677	0.003373	0.0045432
	C PSO	0.010874	0.000252	0.007142	0.00085	0.002382	0.0043
	L-PSO	0.004796	0.013398	0.001453	0.008632	0.002341	0.006124
	Blended BBO	0.01096	0.00108	0.003254	0.019105	0.000546	0.006989

(continued)

Table 3 (continued)

Parameter	Algorithm	10	50	100	200	500	Average
T_x	Rand PSO	0.655016	0.921915	0.385354	0.776072	0.484104	0.6444922
T_y	PM PSO	0.502832	0.023805	0.159177	0.440791	0.136339	0.2525888
	Poly PSO	0.220064	0.033263	0.162125	0.06153	0.207657	0.1369278
	G PSO	0.486602	0.186402	0.200331	0.075301	0.443852	0.2784976
	C PSO	0.376899	0.403617	0.282945	0.57978	0.195036	0.3676554
	L-PSO	0.009539	0.163548	0.223046	0.270093	0.089128	0.1510708
	Blended BBO	0.72184	0.03222	0.113701	0.067198	0.237947	0.2345812
	Rand PSO	0.348973	0.112831	0.121461	0.464107	0.214681	0.2524106
T_z	PM PSO	0.058383	0.602893	0.101501	0.016955	0.094817	0.1749098
	Poly PSO	0.404024	0.052901	0.133641	0.708794	0.410576	0.3419872
	G PSO	0.032359	0.123519	0.154569	0.276816	0.138895	0.1452316
	C PSO	0.077124	0.042046	0.062864	0.63851	0.120051	0.188119
	L-PSO	0.027562	0.058642	0.096728	0.240211	0.165829	0.1177944
	Blended BBO	0.00296	0.00702	0.035731	0.013615	0.010156	0.0138964
	Rand PSO	4.558761	1.423436	4.806374	3.697145	0.909133	3.0789698
alpha	PM PSO	1.558562	0.160899	0.263311	6.910535	3.17732	2.4141254
	Poly PSO	0.906546	7.254022	1.185153	3.026315	2.322818	2.9389708
	G PSO	0.637114	3.056054	6.319311	4.290712	3.641515	3.5889412
	C PSO	2.812785	8.606612	0.132218	1.825775	5.913478	3.8581736
	L-PSO	0.519471	1.312794	3.318024	0.965552	2.608121	1.7447924
	Blended BBO	0.14104	0.29457	1.555332	1.102941	0.027628	0.6243022
	Rand PSO	0.264531	0.297598	0.226021	0.366034	0.361085	0.3030538
beta	PM PSO	0.264531	0.297598	0.226021	0.363069	0.361085	0.3024608
	Poly PSO	0.264531	0.297598	0.226021	0.366034	0.361085	0.3030538
	G PSO	0.264531	0.297598	0.226021	0.366034	0.361085	0.3030538
	C PSO	0.264531	0.297598	0.226021	0.366034	0.364033	0.3036434
	L-PSO	0.264531	0.297598	0.226021	0.366034	0.361085	0.3030538
	Blended BBO	0.1748	0.03852	0.018035	0.021191	0.00819	0.0521472
	Rand PSO	0.217726	0.253488	0.115886	0.1006	0.483546	0.2342492
gamma	PM PSO	0.213613	0.511743	0.194694	0.775676	0.23811	0.3867672
	Poly PSO	0.24171	0.380763	0.170667	0.347232	0.147447	0.2575638
	G PSO	0.246247	0.527518	0.474766	0.747659	0.780586	0.5553552
	C PSO	0.238898	0.648911	0.268385	0.104427	0.219637	0.2960516
	L-PSO	0.225727	0.640835	0.48925	0.867623	0.822768	0.6092406
	Blended BBO	0.01392	0.00072	0.004543	0.024705	0.008408	0.0104592
	Rand PSO	0.000946	0.00022	0.000057	0.000121	0.010253	0.0023194
	PM PSO	0.010277	0.001793	0.000175	0.00033	0.012286	0.0049722
	Poly PSO	0.002064	0.001163	5.76E-05	0.000182	0.011406	0.0029745
	G PSO	0.009871	0.013056	0.000168	0.000272	0.010915	0.0068564
	C PSO	0.002473	0.000184	4.14E-05	0.000386	0.011693	0.0029555
	L-PSO	0.006225	0.013733	1.42E-05	0.000586	0.010614	0.0062344
	Blended BBO	0.00176	0.00288	0.00061	0.004502	0.005023	0.002955

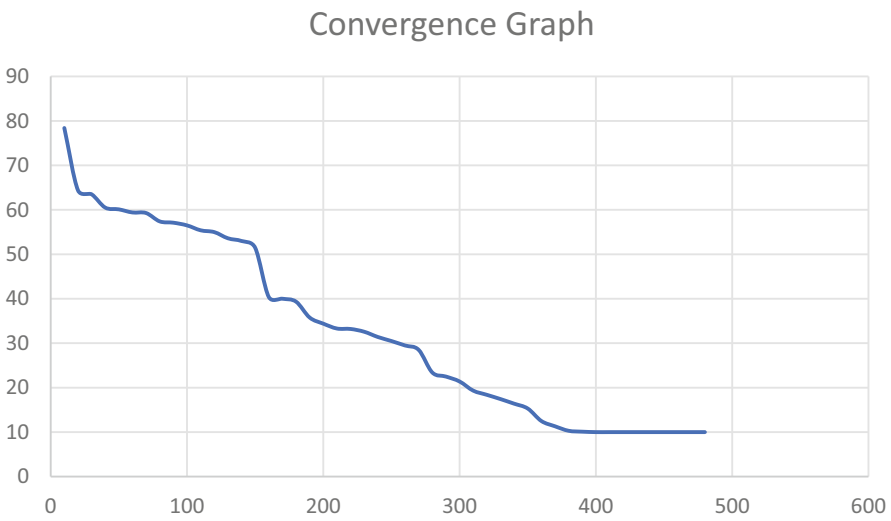


Fig. 2 Convergence behavior for stereo camera calibration problem solved by PSO

5 Conclusion and Future Scope

On the basis of above analysis, it can be concluded that all the six versions of PSO can be used to solve any unconstrained optimization problem. Camera calibration problem is considered as a complex problem in computer vision study. Evolutionary algorithms have not been quite successful in solving camera calibration problem. However, due to their wide applicability on optimization problems, the present study is an attempt to solve such kind of problems using PSO. The problem has been solved using Rand PSO, Poly PSO, PM PSO, C PSO, G PSO, and L PSO. L PSO is proved to be a better version among all the other variants considered. In future, other unconstrained optimization problems can also be solved by using the variants of PSO.

Declarations

Funding: NA
Conflicts of interest/competing interests: All the authors declare no conflict of interest.
Data availability: Authors can confirm that all relevant data are included in the article.
Code availability: Codes are available on request.
Ethics approval: NA
Consent to participate: NA
Consent for publication: All the authors agreed for publication.

References

1. Liang, B., Lehmann, J., Solomon, D., Kinyangi, J., Grossman, J., O'Neill, B., Skjemstad, J. O., Thies, J., Luizão, F. J., Petersen, J., & Neves, E. G. (2006). Black carbon increases cation exchange capacity in soils. *Soil Science Society of America Journal, 70*(5), 1719–1730.
2. Robinson, J., & Rahmat-Samii, Y. (2004). Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation, 52*(2), 397–407.
3. Garg, V., & Deep, K. (2016a). Efficient mutation strategies embedded in Laplacian-biogeography-based optimization algorithm for unconstrained function minimization. *Swarm and Evolutionary Computation, 7*, Issue-2.
4. Garg, V., & Deep, K. (2015). A state-of-the-art review of biogeography-based optimization. In *Proceedings of fourth international conference on soft computing for problem solving* (pp. 533–549). Springer.
5. Garg, V., & Deep, K. (2017). Constrained Laplacian biogeography-based optimization algorithm. *International Journal of System Assurance Engineering and Management, 8*(2), 867–885.
6. Garg, V., & Deep, K. (2019). Portfolio optimization using Laplacian biogeography-based optimization. *OPSEARCH, 56*(4), 1117–1141.
7. Garg, V., & Deep, K. (2016b). Performance of Laplacian biogeography-based optimization algorithm on CEC 2014 continuous optimization benchmarks and camera calibration problem. *Swarm and Evolutionary Computation, 27*, 132–144.
8. Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters, 85*(6), 317–325.
9. Parsopoulos, K. E., & Vrahatis, M. N. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing, 1*(2), 235–306.
10. Sun, S., Zeng, H., Robinson, D. B., Raoux, S., Rice, P. M., Wang, S. X., & Li, G. (2004). Monodisperse mfe2o4 (m = fe, co, mn) nanoparticles. *Journal of the American Chemical Society, 126*(1), 273–279.
11. Bonyadi, M. R., & Michalewicz, Z. (2017). Particle swarm optimization for single objective continuous space problems: a review. *Evolutionary Computation, 25*(1), 1–54.
12. Kennedy, D. (2006). *Writing and using learning outcomes: A practical guide*. University College Cork.
13. Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications, 2008*.
14. Carlisle, A., & Dozier, G. (2001). Tracking changing extrema with particle swarm optimizer. *Auburn University, Auburn, AL, Technical Report CSSE01-08*.
15. Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation, 6*(1), 58–73.
16. Evers, S., Afra, J., Frese, A., Goadsby, P. J., Linde, M., May, A., & Sándor, P. S. (2009). EFNS guideline on the drug treatment of migraine–revised report of an EFNS task force. *European Journal of Neurology, 16*(9), 968–981.
17. Yin, H., Gao, Y., & Fan, C. (2011). Distribution, sources and ecological risk assessment of heavy metals in surface sediments from Lake Taihu. *China. Environmental Research Letters, 6*(4), 044012.
18. Jarboui, B., Damak, N., Siarry, P., & Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation, 195*(1), 299–308.
19. Clerc, M. (2004). Discrete particle swarm optimization, illustrated by the traveling salesman problem. In *New optimization techniques in engineering* (pp. 219–239). Springer.
20. Xinchao, Z. (2010). A perturbed particle swarm algorithm for numerical optimization. *Applied Soft Computing, 10*(1), 119–124.

21. Eberhart, R., & Kennedy, J. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (Vol. 4, pp. 1942–1948).
22. Kennedy, J., & Eberhart, R. C. (1997, October). A discrete binary version of the particle swarm algorithm. In *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation* (Vol. 5, pp. 4104–4108). IEEE.
23. Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)* (pp. 69–73). IEEE.
24. Suganthan, P. N. (1999, July). Particle swarm optimiser with neighbourhood operator. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)* (Vol. 3, pp. 1958–1962). IEEE.
25. Eberhart, R. C., & Shi, Y. (2000, July). Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)* (Vol. 1, pp. 84–88). IEEE.
26. Kennedy, J., & Mendes, R. (2002, May). Population structure and particle swarm performance. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* (Vol. 2, pp. 1671–1676). IEEE.
27. Lovbjerg, M., & Krink, T. (2002, May). Extending particle swarm optimisers with self-organized criticality. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* (Vol. 2, pp. 1588–1593). IEEE.

Analysis of Nonlinear Optimization Problems Using Differential Evolution Algorithm

K. Ramalakshmi, J. Roscia Jeya Shiney, L. Krishna Kumari, and R. Rajalakshmi

1 Preamble

In 1996, Storn and Price developed the differential evolution algorithm (DEA). This method is a stochastic population-based hunt practice that illustrates admirable potential in resolving an extensive collection of optimization issues from numerous areas and a lot of real-world application problems [1]. It is a class of evolutionary computation technique that endures the processes of mutations, crossovers, and operators selection during every generation to attain the overall best possible solution. The current generation's parent vector is also named the objective vector. To generate a mutant vector through the differential mutation, operation mutation is used which is named as the contributor vector. The contributor vector of every objective vector is obtained from the contemporary population by random selection of three distinct parameter vectors from the current population. The weighted difference between two population *vectors* to a third vector is added using *the* mutation process, to generate a *mutant vector.* The crossover process creates the test vector through merging specifications of the mutation vector to the specifications of a target vector chosen out of population. The appropriateness level and assortment process decide the vectors chosen for the next generation, which depend on the engendered test vectors as well as the consequent parent's vectors. This mutation approach and the crossover process have the most influence on the DEA's efficacy. The most significant intrinsic regulatory attributes that regulate the population's diversity and the algorithm's growing of convergence are the scaling

K. Ramalakshmi (✉) · J. R. J. Shiney
Department of Electronics and Communication Engineering, P.S.R Engineering College, Sivakasi, Tamil Nadu, India

L. K. Kumari · R. Rajalakshmi
Department of Electronics and Communication Engineering, Ramco Institute of Technology, Rajapalayam, Tamil Nadu, India

V. Garg et al. (eds.), *Women in Soft Computing*, Women in Engineering and Science,
https://doi.org/10.1007/978-3-031-44706-8_2

factor, population dimension, and crossover rate. The advantages of this algorithm are ease of implementation, reliability, fast operation, and robustness [2–3]. DEA is found to be efficient on a huge range of standard optimization problems. In studies [1], it was found that DEA performed better than simulated annealing as well as genetic algorithms. This method is also more accurate than the controlled random search strategy.

In [4], it is proven that DEA is more accurate than several other optimization methods like simulated annealing and evolutionary programming. The trial vector generation strategies should be properly chosen along with their associated control parameter values to acquire the best possible outcomes for a particular problem. In this approach, several experimental vector generation strategies are available, among which only some are appropriate in working out a specific problem. Control parameters like crossover rate, population size, and scaling factor are important to improve the efficacy of the algorithm. To work out a precise optimization issue, it is necessary to adjust the relevant variables and conduct a time-conserving tryout and miscalculation-incisive process. In the process of evolution, the population of DE moves throughout various areas in the exploration space. In the searching process, definite approaches connected to precise parameter settings possibly will be further efficient than other methods. Hence, an adaptive strategy along with its allied constraint values should be used at diverse phases of the progression process. DEA has emerged as one among the most essential metaheuristics nowadays owing to its ability to resolve various industrial optimization issues. The concert of this method is limited by the following factors: it can easily converge to a local optimum and is dependent on the control parameters. The parameters used for control vary for numerous areas of the space being explored and depend on the situation at hand. It also entails a time-consuming process of trial and error in finding the best parameters for an issue that exists.

Optimization approach is essential in various domains such as statistics, finance, and engineering. Most of the realistic systems have fitness functions with features called noncontinuity, nonlinearity, multidimensionality, and nondifferentiability. They also exhibit problems such as convergence within local minima and stochasticity. Analyzing and solving problems are difficult. DEA belongs to the class of genetic algorithms, and evolutionary programming. Metaheuristic search algorithms (MSAs) are considered to resolve complex contemporary optimization issues through adapting their survey methods based on inspiration from various natural phenomena. MSAs conduct searches through various stages of investigation and development throughout the optimization procedure to obtain the overall best results. Exploration engages the progression of determining varied clarifications within the exploration space. Utilization focuses on the exploration process contained by the neighborhood for the best results. An appropriate balance of the investigation and utilization phases is essential to succeeding with numerous categories of optimization issues. MSAs locate the near-global optimum solution without significant modifications to the algorithmic frameworks. The stochastic environment of MSAs facilitates enhanced sturdiness against confined fit-up challenges.

Metaheuristics [5] are potential methods which are exploited when conventional optimization approaches cannot crack multifaceted issues efficiently. Metaheuristics may resolve problems with different characteristics and arrive at an acceptable solution, in a reasonable time when matched up to conventional approaches. Metaheuristics for optimization were able to classify trajectory-related approaches as well as population-related methods. Trajectory-related approaches formulate alterations from a potential elucidation to an assorted one, like simulated annealing, hill climbing, gradient descent, simplex method, and tabu search. Some of the multiple trajectory-based algorithms are the iterative local hunt, variable neighborhood search, as well as greedy adaptive exploration strategy.

In population-related methods like evolutionary algorithms (EAs), groups of findings, also named the population, are used on the similar occasion, by means of a latest population being breed from an elderly one. They obtain a good solution to a nonlinear programming problem in a feasible time. Some of the important evolutionary algorithms are particle swarm optimization, genetic algorithms, and differential evolution. Amalgamation of EAs with local search approaches [6] are capable of constraining the exploration into an enhanced region more from confined optima. They boost the utilization of a little potential region of the exploration space consequently to increase the rapidity of convergence in the searching process.

2 Literature Survey and Associated Works

Several researchers have taken efforts to review the substantial progress made in the differential evolution algorithm for various applications since its introduction in 1997. In [7], the progress of differential evolution-based research is discussed. It focuses on the detailed analysis of various modified DE structures and their performance evaluation. The application of DE to solving rotated problems with different dimensional sizes is discussed in [8]. The authors did not concentrate on DEA in engineering applications, current trends, or future research directions of differential evolution method. The key topics covered in papers [8, 9] are the changes made to the available DE variants to resolve optimization problems related to engineering applications. The theoretical analysis of the differential evolution algorithm, along with future research directions for DE, was elaborated. The study in paper [10] is concentrated on the purpose of differential evolution to resolve fixed and variable cost-effective dispatch issues. Here, problem formulation with equality and inequality constraints considering the objective function for various economic dispatch problems is described along with an outline of the conventional DE alternatives for problem-solving.

Theoretical studies on differential evolution algorithms, like convergence characteristics, computational complication, and population assortment, were substantially discussed in [11]. In [12], the DE alternatives and its functions in energy administration issues were discussed. In most of the literature studies [10–12], the descriptions of the methodology and databases used for data collection are not clearly discussed.

The impact of DE variant's parameter settings to obtain the best possible solution in engineering applications is discussed. Recent research on the modification of DE algorithms were comprehensively described and analyzed in [13]. An elaborated study of the best possible control specifications for DE in various classes of optimization issues to boost the overall seek-out strategy of DE is also elaborated. Open research issues from various viewpoints on the DE variants are also discussed. In addition, upcoming research areas and the ability to deal with open investigation challenges are addressed. In order to address high-dimensional problems, some modifications are required in the conventional DE for performance enhancement. The concert of this approach is dependent on an assortment of control variables such as stagnation, premature junction, and sensitivity. Stagnation occurs in a condition where the assortment of population is large but still not able to congregate to a suboptimal solution [14].

If the control specifications turn into ineffective in support of a particular issue in the assessment space, then stagnation takes place. Many of the researchers have adopted various techniques like differential mutations to perturbations and mutations to assortment pressure, as well as operator amendment techniques toward advancing the concert of the conventional DE approach [15–16]. The performance of differential evolution and the optimal solution attained were based on either binomial or exponential techniques. Adjustable differential evolution algorithm using fuzzy logic approach is proposed in [17]. This technique adapts the control variables like crossover rate and mutation factor of this method with inputs such as the individuals of successive generations and objective function values. A novel approach with an extension of individuals in the population with constraint values is studied in [18]. The key factor in this algorithm is that the adjustment of control parameters by evolution results in individuals of higher quality having a higher probability of propagating them.

An effectual evolutionary approach for global optimization design includes a local search (LS) heuristic. In [19], it is suggested to combine this method with chaos local search (CLS) technique. This method uses local search based on chaos to identify the top players in the process of evolution. Merely the top performers in this iterative process will be taken into account for the following generation. The other individuals are dismissed shortly after which a random generation of them is generated. The proposed work in [20] uses a similar CLS approach after each production for the finest individual. The perturbation is first done in one direction, and based on the result obtained, the disruption will be accomplished in the converse track. The current step size will be shortened to half the size of the prior iteration if the ultimate solution fails to arrive to a universal optimal level. Since this search strategy utilizes a huge number of utility estimations, a neighboring constraint has to be finished to prevent the overuse of assessing resources. Two other alternative techniques to progress the concert of the DE approach are illustrated in [21]. DETLS approach utilizes a trigonometric local seek-out technique. Here, a weighted mean of the three points, including the preeminent as well as two different arbitrary individuals from the population, is considered the latest individual. In the quadratic interpolation approach named DEILS, the same three individuals as in the DETLS

method are selected. Additionally, the DEILS method generates a parabolic curve before locating its minimum point. Both algorithms apply a local search after each generation and keep doing so until the best outcome is found.

A differential evolution through Taguchi local hunt methods (THDE) is suggested in [22]. In this method, a halfway point among two arbitrarily chosen people from the population is utilized.

In this technique, each parent is split into four pieces to produce nine new people. The portion of the parent producing good results is replicated to the children based on the consequence of three parents being estimated for every factor. The offspring produced at the same moment replaces a random member of the population. This type of approach is distinct from the conventional local search method, which substitutes the weakest descendent of the population for the worst descendent in it. Commonly used techniques apply local search strategies to the best individuals. There are several other techniques, as in [19, 22], where LS is utilized to a portion of the preeminent peoples in the population or to individuals with higher fitness values. There are also other methods where LS is utilized with a possibility to every individual in the population. New techniques for the process of mutation in a differential evolution algorithm have a good impact over the concert of the DE approach. For instance, a novel alternate of DE named DE/current to p best is proposed in [23], where $p \in (0, 1)$. In this method, the highest percentage of individuals is considered the best individuals that afford a steadiness among both investigation and utilization phases. In another method for DE/current to gr-best, as in [24], the set of descendants providing good results such as $q\%$ of total population are selected randomly. As a substitute of tuning the constraint parameters and adopting new methods for mutation, researchers have also suggested integrating ensemble strategies and parameters into evolutionary algorithms.

A hybrid mutation method has been proposed [25] as a new alternate of DE. In a hybrid mutation technique, two kinds of mutation and self-adapting control parameters are incorporated. In many of the research works, impression of compound mutation approaches and binomial crossover for generating experimental vectors are utilized for finding solution in embarrassed optimization issues. The adaptive method, when combined with the conventional differential evolution algorithm, improves system performance but has the problem of a slow convergence rate [26]. The self-adaptive and adjustable variable controls in a DEA were able to increase the resilience and rate of divergence with proper design so that the system spontaneously adjusts to the parameters. The parameter adaptation is an approach named the adaptive DE method (ADE), and it is essential to be attuned. The self-adaptive variable adjusts them in a dynamic manner and controls the value assignments over processing according to certain predefined rubrics. Thus, in an optimization problem, the parameter can be dynamically adjusted by utilizing self-adaptive and adaptive controls, or a mixture of both [27].

The adaptive DE assists in the exploitation phase to prevent the convergence issue and attain the final global optimum value. Many research studies have been suggested by various authors for hybridizing the traditional evolutionary algorithm in order to resolve various optimization issues. The primary population of the

DE algorithm is produced through problem-oriented algorithms. The successive explanations from various iterations are obtained using a different evolutionary computation technique improved through a local hunt strategy. This kind of grouping is named a memetic approach. In addition, mutation operations can be modified to obtain solutions for individuals in the population with higher probabilities than those of others. There is no crossover or mutation operation in an estimation of distribution algorithm (EDA) [28], which is the special category of the existing progression computation techniques. The innovative outcomes are sampled based on the possibility of distribution. This process takes place by utilizing the prevailing descendants acquired in the preceding creations.

Many research works have progressed in EDA and its variants since it proved to be an effective algorithm in combined as well as perpetual realms. In this algorithm the main factor to be considered is the construction of outstanding probability distribution models to define the essential features of the issue. To define favorable regions in the exploration phase, the probability model is used. EDA-FL is the new EDA alternate proposed in [29]. In this work, Kalman filtering is used to improve the modeling accuracy, and a training approach is utilized to enhance the efficacy of sampling. A novel multifactorial variant of this method named PMFEA-EDA is proposed by means of introducing a novel sampling approach to handle huge combinatorial demands of Internet-based jobs. In an estimation-distribution differential evolution (EDDE), the control variables are sampled from the reduced Gaussian PDFs which adjust to the most favorable values in the optimization process.

3 The Differential Evolution Algorithm

This algorithm is different from other computation techniques in the way it combines responses utilizing a difference factor rate of specified single vectors, and the recombination of individuals takes place through a method based on probability. The differential mutation process is the key factor that differentiates DEA in comparison to various population-based approaches. The mutation rule is applied to every parameter within the vector space and enhances the exploration phase, depending on the other solutions. Thus, mutation strategies improve the capacity to find new potential descendant according to the spreading of solutions inside the vector space [30]. DEA belongs to the category of genetic algorithms. It is used for solving real-world problems in an effective manner. It is a convenient design tool that can be easily accessed for practical applications. DE is useful in finding the optimum solution for various science and engineering applications to deal with complex problems without the need for expert knowledge or complex algorithmic design. In this method, the process of mutation is used as the search strategy, which leads the investigate toward the probable province during the search. They are dependent on crossover mechanisms, which swap information among solutions to

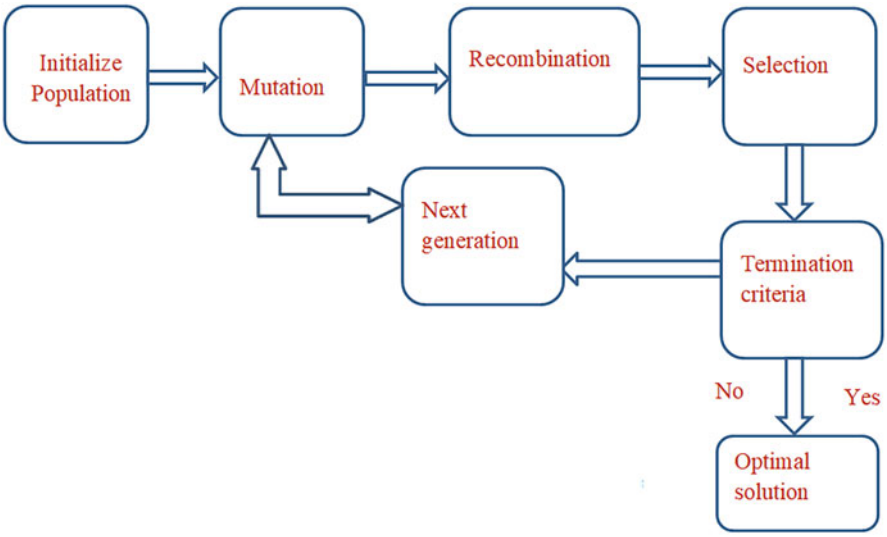


Fig. 1 Steps involved in the differential evolution algorithm

attain the optimum result, while other evolutionary computation techniques utilize the primary search method called mutation.

The performance of the DEA is based on the selected approach as well as the constraints. The DEA utilizes mutation, crossover, and selection operators in its activities to identify the global optimum resolution for every successive generation. The basic differential evolution technique is depicted in Fig. 1 alongside its various phases. The size of population, crossover rate, and scaling factor are parts of the constraints. The objective function is elected arbitrarily from the beginning population to begin the differential evolution method, which is population-based and initiated with a remedy to the issue at hand. In this initial population, the number of vectors population (N) is elected within the limits.

An arbitrary number creator for every vector inside the specified parameter limits are defined by the equation given as

$$x_i = \text{random}\,(0, 1)\,.\left(x_j^U - x_j^L\right) + x_j^L \tag{1}$$

The random function yields a constant arbitrary number in the sort (0, 1). A function is optimized using R real parameters; N is the population size, which should have a minimum value of at least 4. The parameter vector is of the form:

$$x_{i,G}=[x_{1,i,G} \quad x_{2,i,G} \quad x_{3,i,G} \qquad\qquad x_{D,i,G}]$$

where $i = 1, 2 \ldots N$ and G is the generated number.

The lower and higher limits of every variable is defined as

$$x_j^L \leq x_{j,i,1} \leq x_j^U \tag{2}$$

Initial parameter values are arbitrarily selected at regular gap. All the parameter vectors encounter the processes of recombination, mutation, and selection. The search space gets expanded by mutation. From this parameter vector, three vectors are arbitrarily elected with some indicator. In the next step, add the biased variation of these vectors to the third vector, resulting in the donor vector. The donor vector is defined as

$$v_{i,G+1}=x_{r_{1,}G+}F\left(x_{r_{2,}G} \quad -x_{r_{3,}G}\right) \tag{3}$$

The range of mutation factor is [0, 2]. The recombination process includes the dominant outcomes from the earlier creation. In the recombination process, the experimental vector $u_{i,\,G+1}$ is formed by the elements of the donor vector $v_{i,\,G+1}$ and the variables of the target vector $x_{i,\,G}$.Then the variables of the donor vector form the experimental vector through likelihood CR. In order to regulate the differential mutation exploration method, the algorithm follows a uniform crossover method to create the experimental vectors. An experimental vector is obtained from the values of two diverse vectors. The crossover probability is denoted as CR, and fraction of the CR value is duplicated from mutant vector. The crossover probability value is evaluated with an arbitrary number $\text{rand}_{j,\,i}$ whose value lies within [0, 1]. If an arbitrary number is a smaller amount or identical to the experimental variable is innate from the mutant vector the variable is obtained from $V_{j,\,i,\,G+1}$ or else the variable is obtained from the vector $X_{i,\,j.}$

$$u_{j,i,G+1}=\begin{cases} v_{j,i,G+1} & \text{if } \text{rand}_{j,i} \leq \text{CR or } j = I_{\text{rand}} \\ x_{j,i,G} & \text{if } \text{rand}_{j,i} > \text{CR or } j \neq I_{\text{rand}} \end{cases} \tag{4}$$

where $\text{rand}_{j,\,i} \sim$ U [0, 1], I_{rand} is an arbitrary integer commencing $[1, 2 \ldots, D]$ also I_{rand} assure that $v_{j,\,i,\,G+1} \neq x_{j,\,i,\,G}$. In the next step, the intention vector $x_{i,\,G}$ is contrasted with the experimental vector $v_{i,\,G+1}$, and the one with the least possible function assessment is passed on to the upcoming evolution. In the selection stage, there is a process to find out when the trial vector $u_{j,\,i,\,G+1}$ has a significance of the fitness function less than or equal to that of its intention vector$x_{j,\,i,\,G}$. In the next iteration, by contrasting each of the experimental vectors with the objective vector from that the variables are duplicated, the algorithm either switches the objective vector in or retains the same position in the population:

$$x_{i,G+1}=\begin{cases} u_{i,G+1} & if \quad f\left(u_{i,G+1}\right) \leq f\left(x_{i,G}\right) \quad i = 1, 2 \ldots N \\ x_{i,G} & \text{otherwise} \end{cases} \tag{5}$$

After updating the population, selection, recombination, and mutation prolong in anticipation of certain end clause is accomplished. To attain the best possible solutions with this method, the above steps are to be followed. The concert of the DEA is inclined through the constraints like crossover rate as well as mutation. There are three types of parameter control: deterministic parameter control, self-adaptive parameter control, and adaptive parameter control. Self-adaptive parameter control and adaptive parameter control can vary the control parameters without previous knowledge about the problem behavior in the searching phase. Some of the recent applications of the DE approach include:

- DE algorithm is used in electricity market applications to find best possible offers based on day by day bidding action.
- It is used in the optimization of strategies for checkers.
- It is used in the multicast direction-finding approach for the subsequent generation of internet to improve the superiority of service.
- Power plant control applications.
- Digital filter design.
- The DE algorithm occupies an eminent part in network system configuration of distributing systems.
- Maximization of profit in an industrial process.
- It is used in the controller design of an aircraft stability control system.
- Optimization of the process involved in alcohol fermentation
- DEA is used in color image quantization to lessen the amount of image colors.

The DE approach has the features such as simple structure, speed, robustness, and ease of use. The effectiveness of the traditional DE approach is greatly relying on the experimental vector creation procedure and the connected variable values are utilized. Thus, an improper selection of these would result in premature convergence or stagnation.

4 New Techniques and Algorithm to Improve the Standard DE Algorithm

For a successful optimization procedure, the differential evolution method links numerous control factors, such as mutation and crossing rates. In the adaptive DE algorithm (ADE), the parameters are adjusted, which relies on a feedback factor that is dynamically adapted. In self-adaptive parameter control, specific parameters are altered according to a set of predefined rules. Diverse characteristics, including the crossover percentage, variation rate, and size of the population, are fine-tuned through differential evolution in a population that is self-adapting. The fuzzy adaptive differential evolution (FADE) method is a type of DE strategy that uses fuzzy logic controllers to control the variables involved in the crossover as well as mutation processes. In this algorithm, the amount of the population is supposed to

be attuned and is fixed all through the progression process. The parameters required for both mutation and crossover operations are determined for each generation using the current generation and the fitness value acquired in the adaptive DE algorithm provided in [31]. For every parent of a creation, the children are built using the mutation as well as crossover rates. In an algorithmic framework proposed in [32], various mutations as well as crossover approaches are used. In this method, a sole mutation strategy is adapted by choosing one from the available mutation strategies. In this work, the concert of the mutation approach is based on the progress of development during the searching process.

When the DE algorithm is used for optimization, there is no specific value for the crossover rate and scaling factor that is effectual in all the seek-out phases. In practice, numerous possible groupings of cross over rate as well as scaling factor were able to be efficient all through various phases of the investigate progression. One of the most effective DE variants is differential evolution with a multipopulation-based ensemble of mutation methods (MPEDE). The control parameters in a traditional DE algorithm are fixed, and only one mutation approach is used. As a result, the performance of DE might vary substantially depending on the optimization problem [33]. Many research works have addressed issues related to automatic parameter tuning and the construction of exceptional ensembles of mutation strategies. The SaDE algorithm, which includes a combined set of two transformation strategies and self-adapted factors; the DE [34] algorithm with self-adaptive tuning parameters; the CODE algorithm, which includes an arrangement of transformation approaches with their own factors; and the MMRDE algorithm, which encompasses multiple transformation strategies determined by the roulette wheel selection, are some of the improved DE variants.

MPEDE [35] is a variation on the traditional differential evolution algorithm that is based on the multipopulation notion. It separates the entire population into four subgroups and employs various tactics. This approach employs three distinct mutation mechanisms. There are three equal indicator subpopulations of small size and a reward subpopulation of considerable size in this system. A novel sequence is used to adjust the mutation probability and crossover rate throughout time. After a certain number of generations, the optimal mutation strategy is assigned to the reward subpopulation. The control parameters and mutation strategy used are significant in improving the performance of the DE algorithm. Tracking the state of convergence during the procedure allows for improvement. In a DE algorithm, a low recombination rate (F) assists in the exploitation phase, whereas a high F improves the exploration process.

The crossover rate (CR) is important for preserving population variety. During the initial phase of the process of iteration, a higher CR value can be used, which is then steadily decreased using a controlled sequence. This is done to maintain diversity so that system performance is not reduced. Linear decreasing mode and linear increasing mode are the two different control sequences of the crossover rate used. The variation in the mean fitness value is calculated, and if the fitness value does not change between iterations, the sequence will switch from declining to growing or growing to declining mode. If the fitness score of the objective

function remains stable for a given procedure, there is a chance of diversity issues, exploration issues, or trapping in the local optimum. To conquer this issue, the control parameter value must be tuned accordingly. Thus, the tracking convergence over time based on the mathematically adjusted version of the DEA enhances the system's performance. The concert of the new DE can be improved through varying the mutation operator by means of adjustable mutation factors and mixed mutation approaches. In the adjustable mutation scheme, the value of the mutation factor is adjusted from a larger significance to a smaller significance based on the convergence characteristics to pass up early union during the initial investigating phase and to improve convergence to the best possible solution during the future phase of the investigating process.

An integrated substitution strategy (IMS) combines three distinct mutation approaches. IMS increases the assortment of arbitrary exploration and prevents early convergence to a confined bare minimum. Various local search methods can be integrated with the DEA to increase the rate of convergence. In restart differential evolution (RDE) approaches with a restricted investigate mechanism, a novel proposed local mutation rule is based on the location of the best and worst individuals in the population at large. In this method, each vector learns from the position of the greatest and weakest people in a given generation's total population. The location of the best and worst vectors of a specific generation determines each mutant vector's new location, which tracks the same direction as the best vector. RDE supports global exploration at the start of the search process based on the likelihood of employing the local mutation strategy. In [36], a crossover-based adaptive local search (LS) strategy is developed to improve the performance of the basic DE approach. This algorithm employs a hill-climbing heuristic, which modifies the length of the search as needed. In [37], new variants of the DE algorithms are implemented using two alternative local search techniques called trigonometric local search (TLS) and interpolated local search (ILS). The system's performance is improved by using this strategy without modifying the convergence characteristics. The system's performance is improved by using this strategy without modifying the convergence characteristics.

An adjustable local hunt strategy balances the scale of global and local hunt involved. It compares the performance of both global and local search methods to determine its preference for optimization. The local hunt operation based on the Hadamard matrix is named the Hadamard local strategy (HLS). HLS searches can enhance the likelihood of investigating the best results in the search area by creating large number of children. This step is useful to balance the investigation and development phases. The parameter adaptation mechanism incorporated into HLS is used to resolve huge optimization issues. The hybridization of DEA with the inclusion of local search techniques would increase system efficiency [38]. In this technique, the local exploration method is adapted to the most excellent character in every creation to explore the major favorable areas all through the stages of evolution. This step increases the speed of convergence and reduces the chance of trapping within the local minimum. An important factor to be considered in the local

search strategy is to find out which individuals in the population must go through the local search technique.

Recent literature works reveal that local search techniques can be utilized to the finest descendants in the population or the proportion of descendants producing best possible solutions. It can also be related to descendants with higher suitability values than the average or with a possibility to every descendants of the entire population. The purpose of developing a new crossbreed DE approach is to enhance the optimization method to resolve various issues in the production fields and machining operations. There are many approaches to hybridize a traditional differential evolutionary approach to resolve optimization issues. To conduct parameter optimization in industrial process, various evolutionary computation techniques have been modified with other optimization techniques. A parallel genetic simulated annealing (PGSA) algorithm [39] is used to find the optimal machining parameters for milling operations. The results of this algorithm reveal its effectiveness in a milling operation. The integration of the differential evolution and estimation of distribution algorithm (DE/EDA) is used to resolve the transmission network expansion planning (TNEP) problem. The TNEP problem aims to find the new transmission circuits to be included in the electrical system with the minimum investment cost and meet future power demand. TNEP takes into account various methods like mathematical programming and metaheuristic techniques. In every generation, the probability vector is updated and then sampled to produce the next population. The first step involves initializing the probabilistic model. The probabilistic model is characterized by p which is depended on a normal distribution function p (μ, σ). The initial value of the mean μ is created arbitrarily in the search space [$X_{\min}$,$X_{\max}$]:

$$\mu_j^{G=0} = \text{round}\left(X_{\min,j} + \left(X_{\max,j} - X_{\min,j}\right) * \text{rand}\right) \tag{6}$$

The initial standard deviation σ is set to a higher value to attain the diversity of individuals. Let n be the dimension of the problem. For the entire set of iterations, the values are updated as follows:

$$\mu_j^{G+1} = (1-\eta) * \mu_j^G + \eta * \left(x_{1,j}^G + x_{2,j}^G - x_{\text{worst},j}^G \right. \tag{7}$$

where ŋ is called the learning rate ∈ (0, 1), $x_{1,j}^G$ and$x_{2,j}^G$ represents the first two individuals with the best solutions, and $x_{\text{worst},j}^G$ represents the individual with the worst solution of the current population.

$$\sigma_j^{G+1} = (1-\eta) * \sigma_j^G + \eta * \sqrt{\sum_{i=1}^{m} \left(x_{1,j}^G - \mu_{\text{best}}^G\right)^2 \Big/ N_{\text{best}}^G} \tag{8}$$

where N_{best}^G is the amount of descendants with the preeminent results from the up to date generation and μ_{best}^G is the mean of N_{best}^G individuals. Once the probability

vector p is updated, it is utilized to create the subsequent generation through a random normal distribution N (μ, σ). *Then the* new population is generated by the sampling process. Then the population is evaluated to find the optimal power flow for the TNEP problem. The results tested on an IEEE 118 bus system and a Garver 6 bus system illustrate that hybridization of DE-EDA is proficient and improves the efficiency and robustness of metaheuristics. It is also found that hybridization is an efficient way to solve complex optimization problems. A hybrid differential evolution algorithm based on the gaining-sharing knowledge algorithm and Harris hawk optimization (DEGH) [40] accomplishes brilliant performance even with a fixed scaling factor. Harris hawk optimization (HHO) is an innovative swarm-based algorithm that replicates the cooperative behavior and chase pattern of Harris hawks in the hunting process. It involves three phases known as exploration, transition from exploration to exploitation, and exploitation. DEGH includes a dual-insurance mechanism in the mutation operation with four mutation strategies to realize a balance between the exploration and exploitation phases. The crossover probability has a positive effect on the choice of the mutation operator for each individual. The internal phases of the differential evolution algorithm are closely interrelated, such that the crossover probability is closely associated with mutation and selection operations. The crossover probability self-adaption strategy used in this algorithm strengthens the interconnection between the mutation, crossover, and selection stages of the algorithm. The crossover probability and scaling factor adapt the evolution strategy of each individual to make the algorithm more suitable for numerous applications. Recent research trends in the differential evolution algorithm use substantial heuristic techniques to enhance their search strategy and parameter adaptation.

The smart sampling (SS) method is proposed in [41] to detect the favorable areas of the exploration space anywhere the best possible solution exists. In this method, depended on the appropriate value, the preliminary population is filtered to choose the most promising solutions based on a trained classifier. The better solutions are appended to the population, and the classifier is used to recognize the downsized population in order to preserve a predetermined cardinality via discarding the other members. This progression is continual until a convergence norm is met and DE is initialized with members from the favorable region in the search space. This method is suitable for multimodal and high-dimensional problems.

The rotation matrix for this is constructed up of the eigenvectors of the population's covariance matrix. By combining the target and contributor vectors with the eigenvector matrix, the vectors are rotated. The conjugate transpose of the rotation matrix is multiplied once the trials have been created. The individuals go through a crossover procedure once every generation with a predetermined likelihood; the likelihood is referred to as the control variable [42].

Parallel computing is a form of high-performance computing. In parallel computing, simultaneous solving is depended on the rule that a great issue is split into reduced ones and solved concurrently. Parallel DE algorithms are used to improve the speed and accuracy of exclusive optimization problems. Differential evolution algorithms are used on modern parallel computing platforms based

on super parallel SIMD (single-instruction, multiple-data) devices like graphical processing units. The essential conditions needed for convergence, the variations of the DE population, and the computational complication of the DE approach are interlinked and considered important aspects of the algorithm. Hence, a combined method of addressing these three issues concurrently is needed to walk around their intercorrelation. DEA offers the users a strong set of offspring generation methods and is thus a robust optimizer in solving difficult optimization scenarios.

5 Methods to Enhance DE by Controlling the Population Diversity

DE improves a candidate solution depended on an evolutionary progression iteratively in order to optimize an issue. Such algorithms are capable of rapidly navigating very large design spaces and make little to no assumptions about the underlying optimization problem. By carefully selecting a subclass of the strategy pursuit, contemporary optimization techniques can efficiently explore the collection of all potential solutions or plan pursuit space. However, time taken for computation increases substantially toward the size of the proposed exploration space simply for an adequate expanse of deliberate appraisals which are required to stay convinced in the caliber of the solutions that are identified. This makes impossible for these recent approaches to take massive design issues into account at once. Developing engineers use their expertise, experience, and creativity to solve problems related to building design in the real world; however, these concepts are tough and terrible to replicate and device in systematized optimization approaches [44].

The concert of DE approach named the linear population size reduction (LPSR) control approach can be significantly enhanced by the advancement of more sophisticated population size based techniques. During the iterative process, the LPSR method controls the population size using a linear reduction scheme. Conversely, when the algorithm enters a local optimum, the population declines and the population diversity decreases, making it challenging for LPSR to assist the approach in exiting restricted best solution. Increasing the population appropriately all through the iterative process is a clear solution to this issue. However, in order to boost the population, the population's size must be under control, and an expansion plan must be created.

This study makes an effort to balance DE's capacity for both exploration and exploitation while also regularly increasing the population through the iterative progression, adjusting the population assortment adaptively. This study does not take into consideration creating population control approaches dependent on population assortment because doing so is a complex task. As an alternative, we seek to regulate the population using a straightforward periodic function. A sawtooth-linear population size adaptive (SLPSA) approach is suggested to achieve this. In general, the size of population displays a sliding trend throughout the iterative process. The

local trend can, however, change to the positive side by sporadic population growth. The additional subjects are drawn from outside records that contain experimental vectors that were previously abandoned. The ability of DE to explore and exploit is improved by this novel population control technique. Relying on the distance-based factor adaptations for DISH, the weight upgrade formulas for F and CR are also enhanced. The algorithm's capability for exploring will be strengthened by increasing the weights given to *F* and CR in proportion to the divergence between the parent and trial vectors. Based on SLPSA and amended apprise formulas for *F* and CR, this paper suggests a brand-new variation approach dubbed SLDE.

A distinctive population adapted technique called SLPSA incorporates the traits of sawtooth and linear functions to govern alterations in population size. SLPSA utilizes the sawtooth-linear function's modifying properties across the iterative procedure to mitigate the likelihood that the approach will reach an optimal location and to increase the algorithm's ability to perform a global search. In DISH, the apprise algorithms for *F* and CR weights have been updated. The enhanced apprise formulas substitute the squared Euclidean distance between the parent and experiment vectors for the Euclidean distance between the parent and trial vectors. The technique is superior to DISH at exploring high-dimensional environments because of the detachment between the parent and experimental vectors.

6 DE for Applications Relating Single and Multi-objective Optimization Model

An approach to stochastic optimization is the multi-objective evolutionary algorithm [50, 51]. MOEAs help to predict the best Pareto outcomes for particular issues, much like other optimization algorithms. However, they are not the same as population-based methods. The widespread of the current MOEAs employs the idea of dominance in its operations, though some do not. As a result, this article focuses on the MOEA class, which is based on dominance. With the exception of the use of superiority relationships, the MOEA's optimization mechanism is very comparable to evolutionary algorithms. To select the best option for the creation of the inherited peoples, the impartial value is specifically premeditated for every descendant at each iteration and utilized to find the bond of governance among peoples. It is possible to combine this population with its parent to create populations for the following peer group. Objective space may also allow MOEA the freedom to use some established sustenance procedures, like niching.

Finding the ideal solution values for multiple desired goals is discussed to as multi-objective optimization (MOO). The MOO is chosen because it simplifies problems by avoiding complex equations, which is one of the reasons for its use in optimization. A compromise (trade-off) on some incompatible issues is possible due to the decision-making problem in MOOs. Vilfredo Pareto first popularized MOO. An MOO has a vector representing the objective function. Every vector of

the fitness function is a vector representative outcome. In MOO, numerous solutions are available rather than a single one that serves all purposes best.

The optimization procedure can be used to identify the best value or solution. The optimization issues can involve using a single objective or multiple objectives, incisive for the large and small value. MOO describe issues with multiple objectives, which occurs in many places such as mathematics, economics, social studies, aviation, engineering, automotive industry, and agriculture.

The fisheries bioeconomic model can be improved using the MOO in the economics field [52]. It can be used as evaluation tool for resource manipulation and organizing plan efficacy. The theory of public property, depending on the population evolution logistics approach, helps as the groundwork for this model. The case that the author is analyzing encompasses fashioning a model for North Sea fisheries with four goals in mind: maximizing profits, keeping historical quota shares among countries, continuing industry hire, and minimizing surplus. This approach is contrasted with four others, including goal programming (GA1, GA2, and GA3) and genetic algorithms (GA). Here, the gain objective stood out as the most significant difference because it did so in the GP model. This objective's weight in the GA approach was 20 times higher than it was in the GP approach. In addition, all undesirable GA deviations received a 100-fold weighting increase. This was accomplished to make the population's people distinct and to make it harder to determine the most effective people through a particular practice. When creating a GA model, a number of things are to be considered, including the variable specification, tight bounds among variables, weighting, and constrictions. There are sundry standard constraints that can be changed to mark how well the optimization performs. Unconstrained problems are especially well suited for GA deliberation because restriction call for the organization of potential impossibility, which can significantly slow down the optimization process. For the specific development of the problem under investigation, a standard genetic algorithm is typically used, where the modeler should take gain of model arrangement for efficient execution.

Although the GP and GA models have a similar basic structure, the weights vary in an effort to produce comparable outputs and, consequently, valid analogous significance. This results from the fundamental differences between the methodologies: GA is a probabilistic search approach, whereas GP is an exact optimization methodology, and the final solution that is found may be the global optimum. In light of this, it is similar to that the GA will choose an elucidation path that leads to the most "better" solutions, even though it may initially favor less significant goals.

Metaheuristics Multi-objective Optimization: Because of their success in mono-objective optimization, metaheuristics have been extensively modified to handle multi-objective combinatorial problems, exclusively for issues wherever knowledge of the issue is frequently reachable. Contrary to proactive techniques, metaheuristics focus on the answer and allow for an accessible trade-off between the solution's quality and time taken for execution. Metaheuristics are also projected to offer an adaptable framework for a variety of issues. In this study, the vector assessed genetic procedure [43], which modified the genetic algorithm for the multi-objective optimization scenario, was the first multi-objective metaheuristic to be described.

The multi-objective Tabu search [44], the multiple ant colony approach for routing of vehicle issues with time windows [45], the MOSA [46], and the MOSS [47] are only a few of the additional multi-objective metaheuristics that have been developed since then.

One run and multi-runs are the two main metaheuristic paradigms that have been developed. The idea behind one-run-based approaches is to modify the unique metaheuristic so that it returns a batch of outcomes after just one accomplishment. One of this archetype's key benefits is that it adheres to the fundamental idea of metaheuristics, which is to deal with a solution to a certain extent than a resolution constituent at each iteration. In the context of multi-objective optimization, this refers to a set of resolutions with a particular level of performance. However, it is difficult to control how many solutions are returned. It is true that every single solution that gets offered should go through an assessment stage in which it gets contrasted to the formerly conserved likely near-optimal solutions in order to delete any dominated ones. This is valid if the potential near-optimal outcomes are archived in increments. If, for population-based meta-heuristics, namely, genetic algorithms, the near-optimal outcomes get returned unpredictably at algorithm departure, the near-optimal solutions are assessed against one another at this point.

Multi-runs-based techniques, which are based on the a priori multi-objective approach, involve utilizing the creative metaheuristic to a particular grouping of the fitness functions being taken into account. The objective of the metaheuristic is to invent one problem outcome that satisfies a specific preference structure at the end of each run. It is assumed that the algorithm will run as many times as there are near-optimal solutions needed. The existence of two distinct near-optimal solutions under two different preference structures is, regrettably, not guaranteed. Once more, the collected solutions are contrasted with one another to weed out any favored ones. The near-optimal solutions should, nevertheless, be varied or distributed across the Pareto front [48] which demonstrated that still with a uniformly disseminated set of weights, evenly distributed solutions are not always possible. Determining appropriate and effective preference structures beforehand is therefore not simple, at least for a widespread of multi-objective issues, despite the fact that the idea that multi-runs-based approaches go behind is moderately uncomplicated.

MOSA method: It is a subclass of SA extensions to multi-objective optimization that physiques an assessed Pareto anterior by compiling nondominated outcomes revealed while scouring the reasonable province. One who maintains those effective solutions is thought of as maintaining an archive.

A MOSA method has been designed by [46] that uses a list of potentially effective solutions to identify nondominated solutions. This list includes all the generated solutions that have not yet been conquered via some additional built solutions. This approach measures the effectiveness of transitions using weighted functions. Every scalarizing function would prompt a preferred investigate path toward a suboptimal outcome. If the ideal resolution is not found, some nearly ideal solutions would be taken into consideration. With the intention of covering every effective front, the use of a broad assortment of weights has been suggested. Every time a new solution is accepted, the list of potentially efficient solutions is presumed

to be updated, at which point any dominated solutions are taken off the list. The list of potentially effective solutions gets longer if no dominated solution is found [49].

A Pareto simulated annealing, a population-based metaheuristic inspired by genetic algorithms, considers a set of sample solutions produced at each temperature as potentially improving. Each result of the trial is modified in such a way that the accepted innovative result ought be far from the previous outcome's adjoining match. In this method, weights are increased for objectives where the closest solution outperforms the current one, while weights for objectives where the closest solution is superior to the current one are decreased. In the following iteration, both the evaluation and probabilistic acceptance steps will use the new weights combination.

7 Improved Differential Evolution Using Whale Algorithm

The combination of differential evolution algorithm and the whale algorithm called the improved DE whale optimization algorithm (IDEWOA) is used to overcome the issue of trapping within local minimum. The significant features of the improved DE approach are nonlinear convergence, balance among global hunt and local manipulation, crossover mechanism, and enhanced accuracy for solving the optimization problem. The whale optimization algorithm (WOA) has unique strategy of attaining at an optimal solution. The disadvantages of this algorithm are less accuracy and search speed. In a DE algorithm, the initial population selection is very important to prevent limited feebleness at the initial phase. However, the advantages of these two algorithms are used in the DEWOA [53]. The population information is updated based on predation and bubble net techniques in the IDEWOA. The exploration phase is developed by amending the convergence factor. The population diversity is improved by using the crossover as well as selection mechanisms of the DE algorithm. The efficiency of this approach is upgraded by using the elimination mechanism in the crossover strategy to show its superiority in solving various capacity optimization issues. The process of IDECWOA algorithm is described as follows.

The initialization process involves fixing the size of the population (N), dimension of the solution (D), and count of iterations Tmax with the present count of iteration (t). Chaotic mappings are used to generate ergodic and stochastic-natured arbitrary systems from deterministic structures. Logistic and sine mappings are some examples for one-dimensional chaotic mapping with high speed of computation. It is shown in [54] that sine chaos exhibit noticeable chaotic possessions than logistic chaos. Therefore, in the population initialization method of IDECWOA, sine chaos self-mapping is used. The sine mapping is used to initialize the whale population as $\{X_i, \quad i = 1, 2, \cdots, N\}$. Self-mapping sine chaotic is revealed in expression below as

$$x_{n+1} = \sin\left(2/x_n\right), n = 0, 1, \cdots, N \tag{9}$$

where the base value x_n is not zero to prevent fixity as well as zero point in range $[-1, 1]$.

Three mutually dissimilar target vectors $\{X_{i1}, X_{i2}, X_{i3}\}$ are selected randomly from the people. An innovative variant vector is engendered via the discrepancy aspect as per the expression given below:

$$V_i = X_{i1} + F.(X_{i2} - X_{i3}) \tag{10}$$

where variance factor (F) which occupies the range [0, 1].

Further, the crossover operation between the original target vector and the variation vector is used to generate the test vector. In this algorithm the binomial crossover method is more commonly used as per the equation

$$U_{i,j} = \begin{cases} V_{i.j}, & \text{rand i, j } [0, 1] \leq \text{CR} \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{11}$$

where $V_{i.j}$ is the jth dimension of the ith distinct created in the former phase r and i, j is the random number generated among [0, 1]. The crossover aspect is denoted by CR which is also a random number within [0, 1].

The fitness value of the generated test vectors is related to target vectors, and those with higher value of appropriateness are chosen for the succeeding group. The mathematical equation of the appropriateness role is given as

$$X_i(t+1) = \begin{cases} U_i(t), f_{\text{fit}}(U_i(t)) < f_{\text{fit}}(X_i(t)) \\ X_i(t), \quad \text{otherwise} \end{cases}, \tag{12}$$

where f_{fit} is the fitness function. The asynchronous method of selection approach is used in this algorithm. Here, the best experimental vector is exchanged via the corresponding target vector in each step after comparing the target vector with the newly generated experimental vector. The selected best test vector is used in updating the available population. This makes the algorithm to have a faster convergence speed. Then adaptive inertia weight ω value is calculated as per the expression in Eq. (13). The optimum inertia weight strategy ensures strong global search capability of the algorithm.

$$\omega = 0.5 + \exp\left(-\frac{f_{\text{fit}}(x)^t}{u}\right) \tag{13}$$

where f_{fit} is the objective value of the whale x and u is the top aptness value in the first iteration of the population. The adaptive inertia weight has a nonlinear property

which has control over the position. The smaller values of the inertia weight will lead to larger adaptation values with global optimization performance. The proper whale site is taken as the best outcome. It is denoted by X_{best} with its alike global objective value taken as F_{best}. This approach is encouraged through the predatory behavior of humpback whales. The selection is dependent on an arbitrary probability factor $p \in r$ and [0,1] and a factor |A|. An arbitrarily nominated specific whale is chosen as the target in the food pursuit point of the whale algorithm that is illustrated through the equation

$$X(t+1) = X_{\text{rand}}(t) - A.D_1 \tag{14}$$

In the Eq. 14, the value of D_1 is given by

$$D_1 = |C.X_{\text{rand}}(t) - X(t)| \tag{15}$$

In Eq. (15) $X_{\text{rand}}(t)$ is the whale appointed arbitrarily using the entire people. The instantaneous position of the contemporary individual whale is taken as $X(t)$. C is a vector in random distribution between [0,2]. The coefficient |A| is denoted by the expression

$$\text{A} = 2a.r - a \tag{16}$$

$$a = 2 - \frac{2t}{T_{\max}} \tag{17}$$

In the above equations, r is an arbitrary number ranging [0, 1] and the control parameter a.

After finding the random probability factor, the position of distinct whales is updated as per expressions 14–17. Then the fitted value is evaluated for all whale descendants. If the finest appropriateness in a current generation t is greater than the previous value, it is taken as the optimal solution.

8 Novel Clustering-Based Mutation Operator-Based Enhanced Differential Evolution Algorithm

The clustering-based DE approach (CDEA) uses a new operator dependent on the human mental search (HMS) optimization approach [55]. In this enhanced differential evolution algorithm, the existing population is categorized as groups. The potential region which has preeminent mean objective function value has been elected as the cluster. The initial vector of mutation operator is chosen as the robust outcome of convergence area. Here innovative descendent outcomes are included in the present population based on a novel method. Clustering is an unsupervised

pattern recognition technique in which the samples are split into various groups such that the descendants of a similar group show huge similarity when match up to the descendants of other groups. K-means clustering algorithm [56] is more commonly used in many of the applications based on Euclidean distance as the similarity measure. The enhanced mutation operator used in the CDEA is utilized to find a potential area in exploration space as in the HMS algorithm. It is encouraged from the investigation techniques in online auction. It uses a mental seek-out that finds the space about all promising outcomes dependent on Levy flight along with a grouping in CDEA. Then the conventional K-means approach is used to group the existing population. In this algorithm, the cluster number is taken between 2 and $\sqrt{\text{population size}}$ [57]. After the clustering process, the average fitness function of the cluster is determined, and the one which has the robust outcome is used to categorize the field of interest in the area of exploration. Here, the individual which produces the eminent results in the succeeding cluster cannot be the robust solution in the present population. Therefore, grouping of mutation is conducted till M using good mutation and crossover strategies. The number of new candidate solutions is taken as M.

The clustering-based mutation generates M new offspring and with a generic population-based algorithm (GPBA) is used to update this algorithm [58]. The population updation is included in this algorithm to add latest descendent outcomes in the existing population. The v_{cluster} consists of M number of new descendants:

$$v_{\text{cluster}} = w + F\ (u_{r1} - u_{r2}) \tag{18}$$

To assess this approach, it duplicates the searching approaches of the proffer space in online auctions. This approach comprises the following: (1) the exploration of each solution should be dependent on Levy flight, (2) clustering finds the eminent region, and (3) make the solution as robust.

In above equation

u_{r1} and $u_{r2}$are arbitrarily elected candidate outcomes
w is the best candidate solution in the promising region

In the replacement phase of the algorithm, the M descendent outcomes arbitrarily chosen in present population are taken as B. In the set $v_{\text{cluster}} \cup B$, the best M individuals are taken as $\overline{\text{B}}$. The new population is then updated as $(P - B) \cup B$. This process is iterated to attain the optimal solution. The DE-based clustering algorithm is used in unsupervised categorization and segmentation of different image modalities. In such applications the approach is used to reduce quantization error and intra-cluster distances and increase inter-cluster distances concurrently. The exploration of differential evolution approach with other approaches helps to proceed from one outcome to other in finding the best. Recent research work in this approach is focused on combining the mutation and fitness functions proportionately and lacking crossover.

9 Recent Advances in Differential Evolution in Real-Time Applications

A new heuristic approach called the differential evolution (DE) algorithm has three foremost benefits: it discovers the real overall least amount of a multimodal investigation space despite of the preliminary parameter significances; it converges quickly; and it only utilizes a small number of control constraints. It is frequently employed for resolving multidimensional global optimization issues in continuous spaces and has been effective in resolving a variety of issues. In recent decades, it has undoubtedly emerged as one of the majority potent as well as adaptable latest optimizers for constant parameter spaces. DE is employed in many different scientific and technological fields. After providing a thorough overview of the fundamental DE categories, we move on to discuss current proposals regarding parameter variation of DE; single-objective DE-dependent global optimizers; DE used in a variety of optimization situations, together with embarrassed, extensive, multimodal, multi-objective, and variable optimization; fusion of DE with erstwhile optimizers; as well as extensive study on relevance of DE.

Each iteration of DE proceeds all the way through the identical computational procedure as used in a typical evolutionary algorithm, opening through a consistently arbitrary set of candidate findings sampled as of the sufficient investigate dimensions. DE, however, differs significantly from other well-known EAs like evolutionary programming (EP) and evolutionary strategies (ESs) in which it modifies the support vectors using scaled differences of the diverse affiliates of the present population. These variations typically adjust with the objective landscape's base scales as iterations progress. The sampled difference vectors will be lesser in the first variable while higher in the second, for instance, the population becomes condensed in one constraint although stay on as distributed in erstwhile. The search moves of the algorithm are significantly improved by this automatic adaptation. Another name for this characteristic is the self-referential mutation. To put it another way, whereas EP, ES, and erstwhile genuine-coded genetic algorithms (GAs) demand the requirement or adjustment of the unconditional step dimension for every variable on iterations, the canonical DE needs the requirement of a comparative scale factor F for each and every variables. Basic DE stands out as being a very straightforward algorithm with a minimal amount of code needed to implement it in the general coding, contrasting several erstwhile approaches. The canonical DE also has a feature that makes it simple for the professionals to use: it only needs a small number of control specification. However, DE performs admirably when optimizing an ample range of fitness functions in requisites of concluding accuracy, robustness, and working-out speed.

10 Conclusion

This chapter reviewed and used a new multi-objective optimization problem to illustrate how to develop several performance evaluation approaches. This chapter is devoted to differential evolution like a category of metaheuristic processes intended for finding appropriate findings for optimization problems of genuine parameters. Latest approaches and techniques are explored toward improvement of the typical DE algorithm to solve various complex problems. This chapter discusses classification that aided DE by incorporating classification rather than ranking or regression for pairwise comparisons in DE to solve complex features. Various methods to improve the effectiveness of the conventional DE approach are studied, like diversity organizing of the DE population through inserting novel constraints to calculate the assortment throughout the progression. The DE is discussed for specific applications related to single and multiple objective optimization model. This chapter studies the effectiveness of an adaptation scheme for DE parameters using standard techniques to improve overall performance over algorithms. Recent advances in differential evolution algorithm along with real-time application case studies are also analyzed in this chapter. Therefore, in this chapter, the DE approach for working out nonlinear optimization issues is studied elaborately.

References

1. Price, K., Storn, R., & Lampinen, J. (2005). *Differential evolution: A practical approach to global optimization*. Springer.
2. Li, X., & Yin, M. (2014). Modified differential evolution with self-adaptive parameters method. *Journal of Combinatorial Optimization, 31*(2), 546–576.
3. Wang, Y., Xiong, H. L., Huang, T., & Long, L. (2014). Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. *Applied Soft Computing, 18*(5), 232–247.
4. K Price, R Storn, and J Lampinen "Differential evolution – A practical approach to global optimization" Natural Computing series-NC (2005).
5. Russel, S., & Norvig, P. (2003). *Artificial intelligence: A modern approach* (pp. 111–114). Prentice Hall.
6. Jirong, G., & Guojun, G. (2010). Differential evolution with a local search operator. In *Proc. 2010 second international Asia conference on informatics in control, automation and robotics (CAR), Wuhan, China* (Vol. 2, pp. 480–483).
7. Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence, 33*(1), 61–106.
8. Das, S., & Suganthan, P. N. (2010). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computing, 15*(1), 4–31.
9. Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution – An updated survey. *Swarm and Evolutionary Computing, 26*(4), 1–30.
10. Jebaraj, L., Venkatesan, C., Soubache, I., & Rajan, C. C. A. (2017). Application of differential evolution algorithm in static and dynamic economic or emission dispatch problem: A review. *Renewable and Sustainable Energy Review, 77*(3), 1206–1220.

11. Opara, K. R., & Arabas, J. (2019). Differential evolution: A survey of theoretical analysis. *Swarm and Evolutionary Computing, 44*(1), 546–558.
12. Javaid, N. (2018). *Differential evolution: An updated survey. Conference on complex, intelligent, and software intensive systems* (pp. 681–691). Springer.
13. Hong Limb, W., Mohamad Faiz Ahmad, A., NorAshidi Mat Isa, A., & MengAng, K. (2022). Differential evolution: A recent review based on state-of-the-art works. *Alexandria Engineering Journal, 61*(12), 11835–11858.
14. Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review, 33*(1), 61–106.
15. Bergey, P. K., & Ragsdale, C. (2005). Modified differential evolution: A greedy random strategy for genetic recombination. *Omega Journal, 33*(3), 255–265.
16. Qing, A. (2009). *Differential evolution: Fundamentals and applications in electrical engineering*. Wiley.
17. Liu, J., & Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Computing, 9*(6), 448–462.
18. Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transaction on Evolutionary Computation, 10*(6), 646–657.
19. Pei-Chong, W., Xu, Q., & Xiao-Hong, H. (2009). A novel differential evolution algorithm based on chaos local search. In *International conference on information engineering and computer science* (pp. 1–4).
20. Jia, D., Zheng, G., & Khan, M. K. (2011). An effective memetic differential evolution algorithm based on chaotical search. *Information Sciences, 181*(15), 3175–3187.
21. Ali, M., Pant, M., & Nagar, A. (2010). Two local search strategies for differential evolution. *Bio-Inspired Computing: Theories and Applications (BIC-TA), 18*(13), 1429–1435.
22. Peng, H., & Wu, Z. (2015). Heterozygous differential evolution with Taguchi local search. *Soft Computing, 19*(10), 3273–3291.
23. Zhang, J., & Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transaction on Evolutionary Computing, 13*(5), 945–958.
24. Islam, S. M., Das, S., Ghosh, S., Roy, S., & Suganthan, P. N. (2011). An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics Part B (Cybernetics), 42*(2), 482–500.
25. Li Tong, Y., Tang, H. S., Huan, C. H., & Wang, Z. Y. (2023). An improved differential evolution by hybridizing with estimation-of-distribution algorithm. *Information Sciences, 619*(1), 439–456.
26. Liu, J., & Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Computing Journal, 9*(2), 448–462.
27. Brest, J., Bošković, B., Greiner, S., Žumer, V., & Maučec, M. S. (2007). Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing Journal, 11*(7), 617–629.
28. Armañanzas, R., Inza, I., Santana, R., Saeys, Y., Flores, J. L., Lozano, J. A., & Van de Peer, Y. (2008). A review of estimation of distribution algorithms in bioinformatics. *BioData Mining Journal, 1*(1), 6–15.
29. Li, Y., Han, T., Tang, S., & Huang, C. (2023). An improved differential evolution by hybridizing with estimation-of-distribution algorithm. *Information Sciences, 619*(3), 439–456.
30. Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review, 3*(2), 61–106.
31. Ling, M. X., Wang, F. Y., Ding, X., Hu, Y. H., Zhou, J. B., Yang, R. E., & Sun, X. Y. (2009). Cretaceous ridge subduction along the lower Yangtze River belt. *Economic Geology, 104*(2), 303–321.
32. Qian, W., Chai, J., Xu, Z., & Zhang, Z. (2018). Differential evolution algorithm with multiple mutation strategies based on roulette wheel selection. *Applied Intelligence, 48*(4), 3612–3629.

33. Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. *IEEE Transaction on Evolutionary Computing, 13*(5), 945–958.
34. Qin, A. K., & Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. *Proceedings of the IEEE Congress on Evolutionary Computation, 2*, 1785–1791.
35. Mallipeddi, R., Suganthan, P. N., Pan, Q., & Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing Journal, 11*(2), 1679–1696.
36. Simplicio Viana, M., Morandin Junior, O., & Colnago Contreras, R. (2020). A modified genetic algorithm with local search strategies and multi-crossover operator for job shop scheduling problem. *Sensors Journal, 20*(18), 5440–5454.
37. Ali, M., Pant, M., & Nagar, A. (2010). Two local search strategies for Differential Evolution. *IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), 2*, 1429–1435.
38. Durakbasa, M. N., Akdogan, A., SerdarVanli, A., & GunayBulutsuz, A. (2015). Optimization of end milling parameters and determination of the effects of edge profile for high surface quality of AISI H13 steel by using precise and fast measurements. *Measurement Journal, 68*(5), 92–99.
39. Wang, Z.-G., Wong, Y., & Rahman, M. (2010). Development of a parallel optimization method based on genetic simulated annealing algorithm. *Parallel Computing Journal, 31*(08), 839–857.
40. Birogul, S. (2019). Hybrid Harris Hawk Optimization Based on Differential Evolution (HHODE) Algorithm for optimal power flow problem. *IEEE Access, 99*(12), 1–13.
41. Melo, V., & Delbem, A. (2012). Investigating smart sampling as a population initialization method for differential evolution in continuous problems. *Information Sciences, 193*(06), 36–53.
42. Ras, M., Wilke, D., & Groenwold, A. (2014). On rotationally invariant continuous-parameter genetic algorithms. *Advances in Engineering Software journal, 78*(12), 52–59.
43. Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In J. J. Grefenstette (Ed.), *Proceedings of the 1st international conference on genetic algorithms* (pp. 93–100). Lawrence Erlbaum Associates, Inc.
44. Hansen, P. M. (1996). Tabu search for multiobjective optimization: MOTS. In *Proceedings of the 13th International Conference on Multiple Criteria Decision Making (MCDM 97), Cape Town, South Africa*.
45. Gambardella, L. M., Taillard, É., & Agazzi, G. (1999). MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, F. Glover, et al. (Eds.), *New Ideas in Optimization* (pp. 63–76). Maidenhead, UK.
46. Zeng, Z., Zhang, M., Zhang, H., & Hong, Z. (2022). *Improved differential evolution algorithm based on the sawtooth-linear population size adaptive method, 608*, 1045–1071.
47. Beausoleil, R. P. (2006). 'MOSS' multiobjective scatter search applied to non-linear multiple criteria optimization. *European Journal of Operational Research, 169*(2), 426–449.
48. Das, I., & Dennis, J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Journal of Structural Optimization, 14*(1), 63–69.
49. Czyżak, P., & Jaszkiewicz, A. (1998). Pareto simulated annealinga metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis, 7*(1), 34–47.
50. Ulungu, E. L., Teghem, J., & Fortemps, P. (1995). Heuristic for multi-objective combinatorial optimization problems by simulated annealing. In J. Gu, G. Chen, Q. Wei, & S. Wang (Eds.), *MCDM: Theory and applications* (pp. 229–238). Sci-Tech.
51. Lam, T. B., & Sameer, A. (2008). *Multi-objective optimization in computational intelligence: Theory and practice*. Information Science Reference.
52. Mardle, S., Pascoe, S., & Tamiz, M. (2000). An investigation of genetic algorithms for the optimization of multi-objective fisheries bioeconomic models. *Int Trans Oper Res, 7*(2000), 33–49.

53. Li, H., & Fu, C. (2022). An improved differential evolution whale algorithm for economic load distribution. *Journal of Computer and Communications, 10*(10), 132–143.
54. Liu, L., & Zhang, R. (2022). Multistrategy improved whale optimization algorithm and its application. *Computational Intelligence and Neuroscience Journal, 5*(2), 1–14.
55. Mousavirad, S. J., & Ebrahimpour-Komleh, H. (2017). Human mental search: a new population-based metaheuristic optimization algorithm. *Applied Intelligence, 47*(3), 850–887.
56. MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *5th Berkeley symposium on mathematical statistics and probability* (pp. 281–297).
57. Cai, Z., Gong, W., Ling, C. X., & Zhang, H. (2011). A clustering-based differential evolution for global optimization. *Applied Soft Computing, 11*(1), 1363–1379.
58. Deb, K. (2005). A population-based algorithm-generator for real-parameter optimization. *Soft Computing, 9*(4), 236–253.

A Comparative Analysis Report of Nature-Inspired Algorithms for Load Balancing in Cloud Environment

Yogita Yashveer Raghav and Vaibhav Vyas

1 Introduction

Nature-inspired algorithms have several advantages over traditional algorithms when it comes to load balancing. First, traditional algorithms may not be able to handle the complexity of large-scale load-balancing problems, where there are many resources to manage and many constraints to consider. Nature-inspired algorithms, unlike traditional ones, are tailored to address intricate problems and can unearth solutions that traditional algorithms may overlook. Second, nature-inspired algorithms are often more flexible than traditional algorithms, allowing them to adapt to changing conditions and to explore a wider range of solutions. This can be particularly useful in load balancing, where resource demands may fluctuate over time and where there may be multiple objectives to optimize (e.g., minimizing response time and maximizing throughput). Third, nature-inspired algorithms can often find better solutions than traditional algorithms, especially when the solution space is large and complex. For instance, particle swarm optimization has been shown to outperform traditional algorithms in finding optimal load-balancing solutions for large-scale data centers. Finally, nature-inspired algorithms are often more scalable than traditional algorithms, meaning that they can handle larger and more complex load-balancing problems with reasonable computational resources. The capability to scale is essential for load balancing in modern computing environments, where the number of resources and the complexity of applications are expanding rapidly [1, 2].

Y. Y. Raghav · V. Vyas
Department of Computer Science, Banasthali Vidyapith, Tonk, Rajasthan, India

Y. Y. Raghav (✉)
Department of Computer Science, K R Mangalam University, Gurugram, Haryana, India

V. Garg et al. (eds.), *Women in Soft Computing*, Women in Engineering and Science,
https://doi.org/10.1007/978-3-031-44706-8_3

1.1 Characteristics of Nature-Inspired Algorithms

Nature-inspired algorithms (NIAs) are a type of optimization algorithm that applies principles from natural and biological systems to solve complex problems that are challenging to address using traditional methods. Due to their effectiveness, NIAs have become popular in numerous fields such as engineering, finance, computer science, and medicine. Some of the key characteristics of NIAs include:

Search-based: NIAs are algorithms based on search, which means they attempt to find the best solution to a given optimization problem. They work by exploring the solution space and evaluating the fitness of potential solutions.

Population-based: NIAs often utilize a population-based approach, where a set of potential solutions is maintained and evolved over time. Various operators such as selection, crossover, and mutation are applied to the population to generate new candidate solutions.

Stochastic: NIAs are characterized as stochastic algorithms, utilizing randomization to direct the search process. The incorporation of randomness is employed to prevent premature convergence and introduce diversity in the population.

Parallelizable: NIAs are often parallelizable algorithms that can be executed simultaneously on multiple processors or nodes. This feature makes them suitable for optimization problems that require a significant amount of computation.

Robust: NIAs are known for their robustness and ability to handle noisy and uncertain optimization problems. They are capable of finding high-quality solutions even in complex fitness landscapes or with large search spaces.

Easy to implement: NIAs are relatively easy to implement compared to traditional optimization methods. They often require only a few lines of code and can be easily adapted to different optimization problems [3, 4].

In summary, nature-inspired algorithms have demonstrated their effectiveness and efficiency in optimizing a diverse range of problems. They provide a potent alternative to conventional optimization techniques and have the potential to transform many fields by facilitating the optimization of complex systems that were previously challenging or impractical to optimize.

2 Literature Review

Due to their effectiveness in addressing complex optimization problems, nature-inspired algorithms have become a popular class of optimization algorithms. These algorithms draw inspiration from biological and natural processes such as evolution, swarm behavior, and animal behavior. As a result, numerous researchers have conducted literature reviews on different nature-inspired algorithms.

In a research paper [1], a new algorithm has been introduced that uses ant colony optimization (ACO) to distribute workloads among nodes in a cloud network with

the goal of achieving load balancing. The modified ACO approach is customized to meet the specific requirements of cloud or grid networks, with the aim of optimizing node performance. In contrast to the original ACO method, where ants construct individual result sets that are subsequently merged into a complete solution, the modified algorithm continually updates a single result set. The proposed algorithm is designed to address the need for effective load distribution and sustained system functionality.

The utilization of ant colony optimization for load balancing in cloud networks is introduced in this paper [2] through a novel algorithm. The algorithm aims to achieve a balanced workload distribution by searching for underloaded nodes. The effectiveness of the proposed load-balancing strategy is evaluated using CloudAnalyst, and the experimental results demonstrate superior performance compared to conventional approaches like first come first serve (FCFS), soft computing techniques such as genetic algorithm (GA), local search algorithms like stochastic hill climbing (SHC), and existing ant colony-based strategies.

The paper [5] aims to improve the resource utilization of a cloud computing platform by accomplishing multidimensional load balancing of resources across all physical machines. To tackle the intricacy of this challenge, the authors suggest an ant colony optimization algorithm for allocating virtual machines. This algorithm employs a personalized ant colony optimization approach and introduces an improved physical machine selection strategy to avoid getting stuck in local optima and premature convergence. The authors conducted comprehensive simulations to display the efficacy of the suggested algorithm in load-balancing virtual machine allocation and enhancing resource utilization in cloud computing platforms.

The research paper [6] introduces a new approach for load balancing, named LBMPSO, which employs a personalized task scheduling technique based on particle swarm optimization (PSO) to distribute tasks across accessible cloud resources. The primary objective is to improve resource utilization and minimize makespan. The proposed algorithm ensures efficient communication and coordination between resources and tasks within the data center. The CloudSim simulator is employed for the implementation, and simulation findings demonstrate that the proposed technique outperforms current methods by reducing makespan and improving resource utilization.

In paper [7], a novel load-balancing method called LBMPSO is introduced. This method utilizes a customized task scheduling approach based on particle swarm optimization (PSO) to allocate tasks to the available cloud resources. The primary objective is to achieve optimal resource utilization while minimizing makespan by ensuring efficient communication and coordination among the resources and tasks in the data center. The proposed algorithm is implemented in the CloudSim simulator, and its effectiveness is demonstrated through simulations, showing its superiority over existing techniques in reducing makespan and improving resource utilization.

The author of paper [8] introduces an enhanced version of the particle swarm optimization (PSO) algorithm. The initial PSO algorithm was found to have the slowest algorithm speed based on the experimental results, which was somewhat alleviated by the improved PSO algorithm. In terms of task solving, the enhanced

PSO algorithm outperformed the original PSO, while the red-black and Naïve Bayes algorithms exhibited significantly slower performance. With regard to load balancing, both PSO and the improved PSO algorithms showed efficient performance, while the other two algorithms demonstrated poor performance.

In paper [9], a novel technique for load balancing in cloud infrastructure is introduced, which leverages genetic algorithm (GA) to distribute the workload. The primary aim is to achieve a well-balanced workload distribution while minimizing the task completion time. To evaluate the effectiveness of this approach, the study employs the cloud analyst simulator to simulate the proposed algorithm. The results demonstrate that the GA algorithm outperforms existing methods such as FCFS, round robin, and stochastic hill climbing, a local search algorithm. The comparison is based on simulating a typical sample application.

This research paper [10] compares the performance of a proposed algorithm, HMMGA, with other existing algorithms, including Max-Min, IBPSO-LBS, and TOPSIS-PSO. Experimental simulations are conducted, and the results demonstrate that HMMGA outperforms Max-Min and TOPSIS-PSO in terms of reducing makespan. For five VMs, it achieves an average of 1.63 and 3.88 seconds less makespan. Additionally, HMMGA improves resource utilization by 10–40% compared to Max-Min and TOPSIS-PSO. Another experiment shows that HMMGA significantly reduces the average waiting time compared to Max-Min and IBPSO-LBS by approximately 1.7–25.99 s.

This paper [11] review presents a comprehensive examination of load balancing in cloud computing utilizing the ABC algorithm. Additionally, it defines fundamental concepts of swarm intelligence and its properties.

The primary aim of this research paper [12] is to investigate the use of the grey wolf optimization (GWO) algorithm to enhance load balancing in cloud computing environments by considering resource reliability. The proposed approach involves detecting idle or overloaded nodes using the GWO algorithm and then determining the fitness function and threshold for each node. Simulations conducted with CloudSim indicate that this approach is highly effective in producing optimal solutions, as demonstrated by its ability to reduce costs and response times compared to other methods.

In this research paper [13], a new concept is proposed which involves equipping nodes with the ability to comprehend incoming workloads, evaluate various workload and node parameters, and then balance the workload accordingly. To determine the degree of balancing, a probabilistic balancing factor is computed. Additionally, fuzzy logic is used during the accumulation phase to account for any temporal uncertainties that may arise after the initial load-balancing phase. The study shows that incorporating this logic improves performance in terms of computational time, workload, task migration, and cost.

The objective of the study [14] is to introduce a load-balancing (LB) strategy for cloud computing that incorporates the evaluation of the capacity and workload of each virtual machine (VM). When the load of a VM surpasses a certain threshold, the LB algorithm assigns tasks to allocate workload. The proposed approach, referred to as the CS-FA algorithm, identifies the most suitable VMs for task

delegation and relocates overburdened VM tasks to less burdened ones to prevent imbalances in workload. Results show that CS-FA outperforms HDLB by migrating fewer tasks, with only two tasks migrated compared to seven tasks in HDLB. Additionally, in a scenario with 40 loads, CS-FA migrated only two tasks, while the existing method migrated six tasks.

3 Load Balancing

Load balancing is a strategy employed in distributed systems and computer networks to evenly distribute network traffic or workloads across a range of servers, devices, or other resources. Load balancing aims to optimize resource utilization, improve throughput, minimize response times, and ensure system reliability and availability. Load balancing can be applied at various levels of the system, including the network layer, transport layer, and application layer. For instance, a load balancer can be utilized to distribute incoming network traffic over multiple servers, or it can be utilized to distribute database queries over multiple database servers.

All in all, load balancing is a vital approach to guarantee the efficient and dependable functioning of computer networks and distributed systems, particularly in high-traffic environments where the workload can vary significantly over time. By distributing workloads across several resources, load balancing helps to enhance system performance, minimize response times, and ensure high availability and reliability of the system.

3.1 Various Nature-Inspired Algorithm for Load Balancing

Ant colony optimization, particle swarm optimization, and genetic algorithms, which are inspired by nature, can potentially enhance load balancing in computer systems. Load balancing entails distributing workloads among several servers or resources to enhance performance and avoid overloading. There are several ways nature-inspired algorithms can be utilized for load balancing. For instance, ant colony optimization uses pheromones to communicate and locate the optimal path for distributing workloads, where each ant represents a task or workload and the pheromones represent server load. As more tasks are assigned to a server, its load increases, making it less attractive to future tasks. Similarly, particle swarm optimization treats each server as a particle and adjusts its position to reduce overall load imbalance. Genetic algorithms replicate natural selection to find the best load-balancing strategies, evolve them into new generations, and combine them to create an optimal solution.

Table 1 provides a comparison of various nature-inspired algorithms, considering different parameters. Table 2 discusses the applications of nature-inspired algo-

rithms. The subsequent section delves into a detailed discussion of nature-inspired algorithms, including their attributes and equations.

3.1.1 Ant Colony Optimization

The ACO algorithm is a metaheuristic method that is capable of resolving combinatorial optimization problems. It imitates the actions of real ant colonies and utilizes synthetic ants to systematically explore the search space and gradually create a solution. The primary equations utilized in the ACO algorithm are as follows:

Pheromone trail update equation:

$$\text{au}\ (\text{i}, \text{j}) = (1 - \text{evaporationrate})^{*}\ \text{tau}\ (\text{i}, \text{j}) + \text{sum}\ (\text{deltatau}\ (\text{k}, \text{l})) \tag{1}$$

The variables in this equation have specific meanings: tau (i,j) refers to the quantity of pheromone present on the edge between nodes i and j. The constant evaporation_rate represents the rate at which pheromone evaporates. Meanwhile, delta_tau(k,l) refers to the amount of pheromone deposited on the edge between nodes k and l by the ant that discovered the best solution. This pheromone trail update equation modifies the pheromone quantity on the edges depending on the quality of the solutions obtained.

Probabilistic decision rule equation:

$$\begin{aligned} \text{p}\ (\text{i}, \text{j}) = &\left[\text{tau}\ (\text{i}, \text{j})\ \hat{}\text{alpha}\right]^{*}\ \left[\text{eta}\ (\text{i}, \text{j})\ \hat{}\text{beta}\right] \\ &/\text{sum}\left(\left[\text{tau}\ (\text{k}, \text{l})\ \hat{}\text{alpha}\right]^{*}\ \left[\text{eta}\ (\text{k}, \text{l})\ \hat{}\text{beta}\right]\right) \end{aligned} \tag{2}$$

The variables in this equation have specific meanings. The variable p(i,j) represents the likelihood that an ant will move from node i to node j. tau(i,j) indicates the quantity of pheromone present on the edge between nodes i and j, while eta(i,j) is a heuristic value that measures the desirability of moving from node i to node j. Additionally, alpha and beta are constants that control the impact of the pheromone trail and the heuristic value on the decision rule. The probabilistic decision rule equation determines the probability that an ant will opt for a particular path by considering both the pheromone trail and the heuristic value.

Ant solution construction equation:

$$\text{J} = \text{sum}\left(\text{c(i, j)}^{*}\ \text{x}\ (\text{i}, \text{j})\right) \tag{3}$$

The variables in this equation have specific meanings. J represents the objective function value of the solution constructed by the ant. c(i,j) refers to the cost of moving from node i to node j, while x(i,j) is a binary variable that has a value of 1 if the ant moves from node i to node j and 0 otherwise. The ant solution construction equation determines the objective function value of the solution created by the ant by considering the cost of movement and the paths chosen by the ant [15].

Table 1 Comparison of nature-inspired algorithms

Algorithm	Source of inspiration	Objective function	Basic operators	Common control parameters	Algorithm-specific control parameters	Features	Advantages	Disadvantages
Ant colony optimization (ACO) [1, 2]	Ant colony foraging behavior	Reducing the response time while maximizing the utilization of resources	Pheromone trail updates, local search	The variables that can be adjusted in the algorithm are the number of ants, pheromone decay rate, and the values of alpha and beta	The Q-value of the pheromone and the heuristic function are used to determine the next task to be selected	Good capacity to adapt to changing environments, this method can potentially be time-consuming due to its requirement for many iterations. Additionally, it may get trapped in local optima	This method has the ability to adapt to changing environments, handle a wide range of tasks, and optimize multiple objectives simultaneously	Time-consuming due to the need for much iteration. Can get stuck in local optima. Sensitive to parameter tuning
Particle swarm optimization (PSO) [7, 8]	Flocking behavior of birds or schooling behavior of fish	Minimizing response time, maximizing resource utilization	Velocity and position updates	Swarm size, inertia weight, cognitive and social learning rates	None	Good convergence speed and can handle a large number of tasks but may get stuck in local optima and has poor performance in high-dimensional search spaces	Good convergence speed, can handle a large number of tasks, robust to parameter tuning	Can get stuck in local optima, may require a larger swarm size to achieve good results, poor performance in high-dimensional search spaces

(continued)

Table 1 (continued)

Algorithm	Source of inspiration	Objective function	Basic operators	Common control parameters	Algorithm-specific control parameters	Features	Advantages	Disadvantages
Genetic algorithm (GA) [9, 10]	Principles of natural selection and genetic variation	Minimizing response time, maximizing resource utilization	Selection, crossover, mutation	Population size, crossover and mutation rates, selection method	Encoding scheme for solutions, fitness function	Good scalability and can optimize multiple objectives simultaneously, but has slow convergence speed and is sensitive to parameter tuning	Ability to optimize multiple objectives simultaneously, can handle complex search spaces, good scalability	Slow convergence speed, poor performance in high-dimensional search spaces, sensitive to parameter tuning
Artificial bee colony (ABC) [11]	Foraging behavior of honey bees	Minimizing response time, maximizing resource utilization	Employed and onlooker bee phase, local search	Number of bees, abandonment limit, scout bee phase limit	None	Straightforward implementation and effective exploration capability, but it may become trapped in local optima and exhibit slow convergence speed	Simple implementation, good exploration ability, robust to parameter tuning	Suffers from slow convergence speed, is prone to getting trapped in local optima, and necessitates a larger population size for achieving desirable outcomes

Grey wolf optimizer (GWO) [12]	Hunting behavior of gray wolves	Minimizing response time, maximizing resource utilization	Leader, beta, delta, and omega updates	Number of wolves, convergence curve coefficient, scaling factor	None	Good convergence speed and can handle complex search spaces, but is sensitive to initial population size and position and has poor performance in high-dimensional search spaces	Good convergence speed, can handle complex search spaces, robust to parameter tuning	Sensitive to initial population size and position, Poor performance in high-dimensional search spaces
Cuckoo search (CS) [14]	Brood parasitism behavior of cuckoo birds	Minimizing response time, maximizing resource utilization	Lévy flights and egg laying	Population size, discovery rate, Lévy flight exponent	None	Strong ability to explore complex search spaces, but may exhibit poor convergence speed and necessitate a larger population size to achieve satisfactory outcomes.	It exhibits strong exploration capabilities and can effectively handle intricate search spaces, and is robust when it comes to parameter tuning.	The method exhibits slow convergence speed, is highly sensitive to parameter tuning, and may necessitate a larger population size to attain satisfactory outcomes.
Firefly algorithm (FA) [13]	Attracting behavior of fireflies	Minimizing response time, maximizing resource utilization	Attraction and movement	Number of fireflies, attractiveness, absorption coefficient	None	Good convergence speed and can handle a large number of tasks, but may get stuck in local optima and has poor performance in high-dimensional search spaces	Good convergence speed, can handle a large number of tasks, robust to parameter tuning	Can get stuck in local optima, poor performance in high-dimensional search spaces, sensitive to parameter tuning

Table 2 Applications of various nature-inspired algorithms

Algorithm	Applications
Ant colony optimization	The ACO algorithm has demonstrated its effectiveness in numerous domains, including logistics, transportation, communication networks, and machine learning [2]
Particle swarm optimization	The PSO algorithm has found successful applications in various fields, including engineering, finance, image processing, and machine learning [7]
Genetic algorithm	Genetic algorithms are frequently utilized to tackle optimization problems that are challenging to solve using conventional techniques, particularly those with a vast search space or many local optima. These algorithms have been effectively utilized in numerous domains, such as engineering, finance, and computer science, among others [9]
Artificial bee colony	The ABC algorithm has proven to be an effective solution for various optimization problems, such as function optimization, parameter estimation, and feature selection. It is especially beneficial for problems that exhibit a large search space, nonlinearity, and multimodality. Compared to other optimization algorithms, the ABC algorithm offers the advantage of being simple, flexible, and efficient [11]
The grey wolf optimizer	The GWO algorithm has been effectively utilized in solving optimization problems such as function optimization, feature selection, and parameter tuning. It is particularly beneficial for problems with a large search space, nonlinearity, and multimodality. Compared to other optimization algorithms, the GWO algorithm's strengths lie in its simplicity, flexibility, and efficiency [12]
Cuckoo search optimization	The utilization of the cuckoo search algorithm has been observed in multiple optimization problems, such as function optimization, parameter estimation, and feature selection and has exhibited robust convergence characteristics. Furthermore, it has proven capable of identifying high-quality solutions with minimal function evaluations [14]
Firefly algorithm	The GWO algorithm has found successful applications across an array of optimization problems, encompassing function optimization, feature selection, and parameter tuning for machine learning models, among others [13]

3.1.2 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a robust metaheuristic algorithm that is widely utilized to solve optimization problems. The algorithm is influenced by the collaborative actions of bird flocks or fish schools, where individuals work collectively toward a shared objective based on their own and their peers' experiences. PSO employs a group of particles that represent possible solutions to the optimization problem, and each particle strives to discover the best solution by modifying its position in the search space. The PSO algorithm employs the following key equations:

Particle positions update equation:

$$\mathrm{xi}\,(\mathrm{t}+1) = \mathrm{xi}\,(\mathrm{t}) + \mathrm{vi}\,(\mathrm{t}+1) \tag{4}$$

Here, x_i(t) denotes the current position of particle i at time t, v_i(t+1) represents the velocity of particle i at time t+1, and x_i(t+1) is the updated position of particle i at time t+1. This equation is used to update the position of each particle based on its current position and velocity.

Particle velocity update equation:

The velocity of each particle is updated using Eq. (5), which takes into account the particle's current velocity (v_i(t)), personal best position (pbest_i), and global best position (gbest). The equation is expressed as:

$$\mathrm{vi}\,(\mathrm{t}+1) = \mathrm{w}^{*}\ \mathrm{vi}\,(\mathrm{t}) + \mathrm{c1}^{*}\ \mathrm{rand1}^{*}\ \ (\mathrm{pbesti} - \mathrm{xi}\,(\mathrm{t})) + \mathrm{c2}^{*}\ \mathrm{rand2}^{*}\ \ (\mathrm{gbest} - \mathrm{xi}\,(\mathrm{t})) \tag{5}$$

The inertia weight, w, determines the influence of the particle's previous velocity on its new velocity. The constants c1 and c2 determine the impact of the particle's personal best and global best positions, respectively. To introduce randomness into the algorithm, random numbers between 0 and 1, rand1 and rand2, are used.

Objective function evaluation equation:

$$\mathrm{f}\,(\mathrm{xi}\,(\mathrm{t})) \tag{6}$$

The fitness of a solution, represented by the particle's position x_i(t), is evaluated using the objective function f. This function is applied to determine the adequacy of each particle based on its present location in the search space.

Personal best update equation:

$$\begin{aligned} &\text{if } \mathrm{f}\,(\mathrm{xi}\,(\mathrm{t})) < \mathrm{f}\,(\mathrm{pbesti}) : \\ &\mathrm{pbesti} = \mathrm{xi}\,(\mathrm{t}) \end{aligned} \tag{7}$$

To update the personal best position of particle i, Eq. (7) checks if the fitness of its current position, f(x_i(t)), is less than its previous personal best. If this condition is met, pbest_i is updated to x_i(t).

Global best update equation:

$$\text{gbest} = \text{xj}\,(\text{t}) \quad \text{where j} = \text{argmin}\,(\text{f}\,(\text{xi}\,(\text{t}))) \tag{8}$$

To update the global best position based on the particle with the best fitness at time t, the equation gbest = x_j(t) is used, where j is the index of the particle with the minimum value of the objective function among all particles, obtained using the argmin() function [16].

3.1.3 Genetic Algorithm

Genetic algorithms are a type of optimization algorithm that draw inspiration from the principles of natural selection and evolution. To solve a problem, a population of candidate solutions is created and iteratively improved using selection, crossover, and mutation principles. The following steps are involved in the process.

Initialization: Randomly create an initial population of candidate solutions.
Evaluation: Evaluate the adequacy of each candidate solution within the population.
Selection: Select the most suitable candidate solutions from the population by utilizing a selection strategy such as tournament selection, roulette wheel selection, or rank-based selection.
Crossover: Generate new candidate solutions by combining the genes of selected parents. The likelihood of a crossover happening between two parents is determined by the crossover rate.
Mutation: Introduce small random changes to the genes of the offspring population. The likelihood of a gene undergoing mutation is determined by the mutation rate.
Replacement: Replace the old population with the new offspring population.
Termination: Continue the process until a satisfactory solution is obtained or a termination criterion is met. The effectiveness of a genetic algorithm depends on the choice of selection, crossover, and mutation strategies [17].

3.1.4 Artificial Bee Colony

Artificial bee colony (ABC) is a swarm-based optimization algorithm inspired by the behavior of honeybees. ABC employs a population of bees to represent candidate solutions to an optimization problem and communicate with each other to find the best solution. The algorithm involves the following basic steps:

1. Initialization: Create an initial population of candidate solutions randomly.

2. Employed bees phase: Each employed bee explores a neighbor solution and evaluates its fitness based on the problem's objective function.
3. Onlooker bees phase: Select onlooker bees to evaluate the fitness of solutions based on the probabilities calculated from the fitness values. Solutions with higher fitness have a higher probability of being selected.
4. Scout bees phase: If a solution has not improved after a certain number of iterations, it is abandoned, and a new solution is generated randomly.
5. Termination: Repeat the process until a satisfactory solution is found or a termination criterion is met.

The ABC algorithm leverages the information exchange within the hive to improve solution quality, with employed and onlooker bees exploiting this communication and scout bees promoting population diversity. The algorithm's efficacy is affected by the selection of both the search range and the number of scout bees utilized [18–20].

3.1.5 The Grey Wolf Optimizer

The GWO is a metaheuristic optimization algorithm that emulates the social hierarchy and hunting behavior of gray wolves in nature. By utilizing the positions of wolves as candidate solutions to an optimization problem, the algorithm replicates the process of a wolf pack hunting and seeks to identify the optimal solution. The GWO algorithm comprises several fundamental steps, including:

1. Initialization: Generate an initial population of candidate solutions randomly.
2. Evaluation: Calculate the fitness of each candidate solution in the population based on the objective function.
3. The positions of the alpha, beta, and delta wolves are updated based on their fitness values and the positions of other wolves in the pack. The alpha wolf, which represents the current best solution, the beta wolf, which represents the second-best solution, and the delta wolf, which represents the third-best solution, are given priority during the update.
4. The locations of the other wolves within the pack are adjusted by considering the positions of the alpha, beta, and delta wolves.
5. Boundary handling: Check if any wolf has gone beyond the boundary of the search space and move it back to the boundary if required.
6. Termination: Repeat the process until a satisfactory solution is found or a termination criterion is met.

The alpha wolf takes charge in the GWO algorithm, leading the pack and guiding other wolves to explore the search space. The beta and delta wolves follow the alpha's lead and also scour the search space to identify superior solutions. The GWO algorithm's efficacy is contingent on various factors, including the choice of search range, the number of wolves, and striking the right balance between exploration and exploitation.

3.1.6 Firefly Algorithm

The firefly algorithm (FA) is a metaheuristic optimization algorithm that takes inspiration from the flashing behavior of fireflies found in nature. As a population-based algorithm, FA imitates the flashing behavior of fireflies to locate the global optimum of a given problem. The algorithm follows a simple procedure as detailed below:

1. Begin by initializing a population of fireflies randomly in the search space.
2. Evaluate the fitness of each firefly using the objective function.
3. Adjust the brightness of each firefly based on the distance to other fireflies and their attractiveness.
4. Move the fireflies toward the brighter ones.
5. Repeat steps 2–4 until a stopping criterion is satisfied.

To compute the attraction between two fireflies, we can use the equation:

$$\mathrm{A}\ (\mathrm{i}, \mathrm{j}) = \exp\left(-\gamma^{*}\ \mathrm{r}\ (\mathrm{i}, \mathrm{j})\ \hat{}2\right) \tag{9}$$

This equation calculates the attractiveness of firefly i toward firefly j, where r(i,j) is the Euclidean distance between them and γ is a scaling parameter that governs the rate at which the attractiveness decays.

To determine the brightness of a firefly, the algorithm uses the fitness function of the problem being optimized. The movement of a firefly i toward a brighter firefly j is represented using the following equation:

$$\mathrm{r}\ (\mathrm{i}, \mathrm{j}) = \mathrm{r}\ (\mathrm{i}) + \beta^{*}\ (\mathrm{r}\ (\mathrm{j}) - \mathrm{r}\ (\mathrm{i})) + \alpha^{*}\ \varepsilon \tag{10}$$

Here, r(i) denotes the position of firefly i, β is the attractiveness parameter, ε is a random vector with Gaussian distribution, and α regulates the step size of the movement. By continuously adjusting the brightness and position of the fireflies, the firefly algorithm strives to find the global optimum of the objective function [21].

3.1.7 Cuckoo Search Algorithm

The cuckoo search algorithm (CSA) is a metaheuristic optimization algorithm based on the brooding behavior of cuckoo birds in nature, with the aim of finding the global optimum of a given problem. The algorithm involves the following steps:

1. Initialize a population of candidate solutions randomly within the search space.
2. Evaluate the fitness of each candidate solution using the objective function.
3. Select a cuckoo nest randomly and generate a new candidate solution using Lévy flight, a type of random walk that utilizes heavy-tailed distributions.

4. If the fitness of the new candidate solution is superior to that in the nest, replace it.
5. Repeat steps 3–4 until a stopping criterion is reached.

The cuckoo search algorithm (CSA) employs Lévy flight to efficiently explore large search spaces, making it particularly suitable for high-dimensional optimization problems. In CSA, cuckoo nests represent the best solutions discovered so far, and cuckoos lay eggs in these nests to generate new candidate solutions. If a cuckoo finds a better nest than its own, it replaces its egg with the new one. The CSA algorithm employs the "cuckoo fraction" parameter to control the quantity of nests that are substituted with fresh eggs during each iteration, thereby balancing the trade-off between exploration and exploitation. By iteratively generating and replacing candidate solutions, the algorithm aims to locate the global optimum of the objective function [16].

4 Conclusion

Nature-inspired algorithms are a category of optimization algorithms that imitate the behavior of natural systems, such as biology, physics, and chemistry. These algorithms have gained popularity recently because of their ability to solve complex optimization problems efficiently. Several nature-inspired algorithms have been created, each possessing distinct features and benefits. Among the commonly used nature-inspired algorithms are genetic algorithms, particle swarm optimization, ant colony optimization, and artificial bee colony optimization. When comparing these algorithms, it is essential to consider factors such as their convergence speed, robustness, and ability to handle complex optimization problems. In general, genetic algorithms are often better suited for problems with a large search space, while particle swarm optimization is more efficient when the objective function is smooth and continuous. Ant colony optimization is often used for problems involving graph optimization, and artificial bee colony optimization is known for its robustness and ability to handle noisy objective functions. To summarize, every nature-inspired algorithm has its own advantages and disadvantages, and the selection of an algorithm will rely on the particular optimization challenge being addressed. Therefore, it is essential to carefully evaluate the problem requirements and characteristics before selecting an appropriate nature-inspired algorithm.

References

1. Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K. P., & Rastogi, R. (2012, March). Load balancing of nodes in cloud using ant colony optimization. In *2012 UKSim 14th international conference on computer modelling and simulation* (pp. 3–8). IEEE.

2. Dam, S., Mandal, G., Dasgupta, K., & Dutta, P. (2014). An ant colony based load balancing strategy in cloud computing. In *Advanced computing, networking and informatics-Volume 2: Wireless networks and security proceedings of the second international conference on advanced computing, networking and informatics (ICACNI-2014)* (pp. 403–413). Springer.
3. Raghav, Y. Y., Vyas, V., & Rani, H. (2022). Load balancing using dynamic algorithms for cloud environment: A survey. *Materials Today: Proceedings, 69*, 349–353.
4. Raghav, Y. Y., & Vyas, V. (2019, October). A comparative analysis of different load balancing algorithms on different parameters in cloud computing. In *2019 3rd international conference on recent developments in control, automation & power engineering (RDCAPE)* (pp. 628–634). IEEE.
5. Xu, P., He, G., Li, Z., & Zhang, Z. (2018). An efficient load balancing algorithm for virtual machine allocation based on ant colony optimization. *International Journal of Distributed Sensor Networks, 14*(12), 1550147718793799.
6. Pradhan, A., & Bisoy, S. K. (2022). A novel load balancing technique for cloud computing platform based on PSO. *Journal of King Saud University-Computer and Information Sciences, 34*(7), 3988–3995.
7. Visalakshi, P., & Sivanandam, S. N. (2009). Dynamic task scheduling with load balancing using hybrid particle swarm optimization. *International Journal of Open Problems in Computer Science and Mathematics, 2*(3), 475–488.
8. Zhu, Y., Zhao, D., Wang, W., & He, H. (2016). A novel load balancing algorithm based on improved particle swarm optimization in cloud computing environment. In *Human centered computing: Second international conference, HCC 2016, Colombo, Sri Lanka, January 7-9, 2016, revised selected papers 2* (pp. 634–645). Springer.
9. Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., & Dam, S. (2013). A genetic algorithm (ga) based load balancing strategy for cloud computing. *Procedia Technology, 10*, 340–347.
10. Kodli, S., & Terdal, S. (2021). Hybrid max-min genetic algorithm for load balancing and task scheduling in cloud environment. *International Journal of Intelligent Engineering and Systems, 14*(1), 63–71.
11. Kruekaew, B., & Kimpan, W. (2020). Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. *International Journal of Computational Intelligence Systems, 13*(1), 496–510.
12. Sefati, S., Mousavinasab, M., & Zareh Farkhady, R. (2022). Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: Performance evaluation. *The Journal of Supercomputing, 78*(1), 18–42.
13. Susila, N., Chandramathi, S., & Kishore, R. (2014). A fuzzy-based firefly algorithm for dynamic load balancing in cloud computing environment. *Journal of Emerging Technologies in Web Intelligence, 6*(4), 435–440.
14. Kumar, K. P., Ragunathan, T., Vasumathi, D., & Prasad, P. K. (2020). An efficient load balancing technique based on cuckoo search and firefly algorithm in cloud. *Algorithms, 423*, 422–432.
15. LD, D. B., & Krishna, P. V. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft Computing, 13*(5), 2292–2303.
16. Raghav, Y. Y., & Vyas, V. (2023). ACBSO: a hybrid solution for load balancing using ant colony and bird swarm optimization algorithms. *International Journal of Information Technology*, 1–11.
17. Mishra, R., & Jaiswal, A. (2012). Ant colony optimization: A solution of load balancing in cloud. *International Journal of Web & Semantic Technology (IJWesT), 3*(2), 33–50.
18. Mapetu, J. P. B., Chen, Z., & Kong, L. (2019). Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing. *Applied Intelligence*.
19. V. Arulkumar, N. Bhalaji. “Performance analysis of nature inspired load balancing algorithm in cloud environment.”, Journal Ambient Intelligence and Humanized Computing, 2020.

20. Ghumman, N. S., & Kaur, R. (2015). Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system. In *2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*.
21. Surjeet, K., Sabyasachi, P., & Ranjan, A. (2021). Research article a particle swarm and ant colony optimization based load balancing and virtual machine scheduling algorithm for cloud computing environment. *Turkish Journal of Computer and Mathematics Education, 12*(11), 3885–3898.

A Novel Entropy-TOPSIS Approach for Selecting Sustainable Logistics Centre Location Under Pythagorean Fuzzy Environment

Talat Parveen, H. D. Arora, and Pinkey Chauhan

JEL Classification: 94A15, 94A24, 26D15

1 Introduction

Yager [1] proposed the fundamental theory of Pythagorean fuzzy sets (PyFS) to address the shortcoming of Intuitionistic (InFS) fuzzy sets [1, 2], i.e., when the sum of membership (MBV) Ψ value and non-membership (NMBV) π value is more than one, i.e., $\Psi + \pi > 1$ $(0.7 + 0.6 > 1)$. MBV and NMBV in a Pythagorean fuzzy set satisfy the requirement $\Psi^2 + \pi^2 \leq 1$. Since the PyFS membership value domain is larger than the InFS membership value domain, this impression has a wider range than InFS. It is a controlling mechanism to communicate confusing thoughts. Compared to other fuzzy models, it offers more flexibility in dealing with decision-making in the actual world. Yager [1, 2] and Yager and Abbasov [3] investigated the fundamental ideas behind PyFS and clarified the connection between PyFNs and complex numbers. After that, a variety of Pythagorean fuzzy aggregation operators, including PyFWAO, are also suggested. Zhang and Xu [4] then explained the fundamental mathematical procedures for PyFNs. For the selection of the location of distribution centres, a variety of models and approaches, including fuzzy logic, mathematical models, heuristic meta-heuristic methods, and

T. Parveen (✉) · H. D. Arora
Department of Mathematics, Amity Institute of Applied Sciences, Amity University, Noida, Uttar Pradesh, India

P. Chauhan
Department of Mathematics, Jaypee Institute of Information Technology, Noida, Uttar Pradesh, India

V. Garg et al. (eds.), *Women in Soft Computing*, Women in Engineering and Science,
https://doi.org/10.1007/978-3-031-44706-8_4

the multi-criteria decision-making (MCDM) method, have recently been developed [5]. The selection difficulties, however, typically include a wide range of qualitative and quantitative factors. Applying MCDM approaches to create MCDM models is thus a workable solution to such issues. To help decision-makers choose the best distribution centre, a variety of MCDM techniques are available, including AHP, VIKOR, ELECTRE, PROMETHEE, ANP, DEMATEL, WASPAS, and TOPSIS (Technique for Order Preference by Similarity to an Ideal Solution). The current market's fierce competition has forced companies to concentrate on and increase their investments in logistics networks. As a result, researchers and practitioners are increasingly focused on finding ways to reduce logistics expenses and boost the effectiveness of logistics processes [6]. Distribution centres are important nodes in the logistics network that link all logistics-related activities [7]. Therefore, in order to effectively manage the logistics operations and develop the logistics network, it is crucial to handle the logistics distribution centre. The efficiency of a logistics distribution centre is influenced by various factors. One of them, the logistics distribution site, has the biggest impact on the enterprise's overall management costs since it influences the optimization of the logistics distribution system. In order to reduce expenditures and increase profits in a logistics system, it is crucial to select an appropriate distribution centre location [8]. But there isn't enough data to suggest a thorough distribution centre placement and selection model for the logistics sector. This study attempted to construct a hybrid model based on a novel Pythagorean fuzzy entropy measure and the TOPSIS approach to aid in the selection of a distribution centre site. Thus, an integrated entropy-TOPSIS approach is used to compute criterion weights and select the best location for the distribution centre. One of the key objectives of any site selection is to identify the most suitable location that could minimize the total costs. In their research, Agrebi et al. [9] employed the ELECTRE I approach in their study to examine the choice of distribution facilities.

Distribution centre location selection is a significant process in any business and should be done with care. The process of selecting a location for a distribution centre can involve a variety of criteria, such as proximity to customers, access to transportation networks, availability of resources, and cost. To effectively evaluate these factors, the decision-makers must have a comprehensive understanding of the method of decision-making. Accordingly, a literature review of the selection of distribution centre locations can provide useful insight into the various factors that must be considered.

The TOPSIS approach was first introduced by C.L. Hwang and K. Yoon in 1981. This approach successfully addresses the issue of ranking the alternatives. The TOPSIS approach and fuzzy set theory are thus frequently coupled to address problems involving multiple criteria decision-making (MCDM). Han et al. [10] modified the TOPSIS method using entropy and linguistically hesitant Pythagorean fuzzy sets to assess the contribution rates of WSoS- weapon system-of-systems. In order to evaluate the ranks and risk, Akram et al. [11] used the ELECTRE-I technique in a hesitant Pythagorean fuzzy setting. Ulutaş et al. [12] attempted to address the logistic centre location problem for Siva province in Turkey by applying the MCDM model with fuzzy SWARA and the CoCoSo method. Shafiee

et al. [13] employed Pythagorean fuzzy sets with the DEMATEL method to analyse the risk of the consumable product supply chain network during the COVID-19 outbreak. Erdogan and Ayyildiz [14] applied novel AHP-integrated EDAS technology to evaluate pharmaceutical warehouse location selection to address pandemic conditions. Sagnak et al. [15] found transportation cost to be the most important criteria by applying fuzzy best-worst and TOPSIS methods, among several other criteria such as land cost, storage cost, energy cost, investment, etc., in the selection of sustainable collection centres. Yang et al. [16] applied the Pythagorean fuzzy MULTIMOORA method for determining complex criteria weights and selecting the electric vehicle power battery. Yin et al. [17] used IVPyFN to deal with the uncertainty concerning sustainability in rail transit photovoltaic power station site selection, and IVPyFN-TOPSIS was used to aggregate the decision matrix. The Stepwise Weight Assessment Ratio Analysis (SWARA)-TOPSIS in Pythagorean fuzzy environment approach, according to Saeidi et al. [18], is extremely successful and capable of handling the Sustainable Human Resource Management issue in manufacturing organizations. Based on an evaluation model based on Pythagorean fuzzy sets and the TOPSIS method, Li et al. [19] construct an evaluation index system with four aspects using the dispatching results of power systems with a high proportion of renewable energy under four different dispatching modes as the evaluation object. Ayyildiz [20] applied Pythagorean fuzzy SWARA and Pythagorean fuzzy CODAS to estimate the location of the charging station for an electric scooter. Using an integrated methodology based on entropy, Complex Proportional Assessment (COPRAS), and Step-wise Weight Assessment Ratio Analysis (SWARA) methodologies in a Pythagorean fuzzy environment, Alipour et al. [22] addressed supplier selection for fuel cells paired with hydrogen FCH. The supplier selection problem was solved by Yu et al. [23] using the extended fuzzy TOPSIS approach in an interval-valued Pythagorean setting. Parveen et al. [24, 25] applied the entropy-TOPSIS approach to the MCDM problem of selecting an online payment system and ranking the academic institute. Han et al. [26] solved the MCDM problem using TOPSIS based on entropy for Linguistic Pythagorean fuzzy soft sets. The research review indicates that PyFSs are better functionally equipped than InFSs to deal with uncertainty in difficult real-world decision-making situations. As a result, the standard TOPSIS technique that corresponds to the proposed entropy measure in PyFS was expanded in this article. This method is really straightforward and simple to comprehend. The traditional TOPSIS technique has been applied to a wider spectrum of issues. Although several MCDM approaches have been created under the PyFS environment, there have been few studies using PyFS entropy and the TOPSIS approach to evaluate the location of distribution centres. In order to solve this issue, this chapter provides the integrated entropy-TOPSIS technique, which may be used to examine the implicit uncertainty and ambiguity connected to the decision maker's perspective. Here is a quick synopsis of this chapter:

1. A novel PyFS entropy metric is suggested, and it is contrasted with the existing ones.

2. The Integrated Entropy-TOPSIS approach combines Entropy and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), two well-known MCDM techniques. While TOPSIS is used to rank the alternatives according to how well they perform on the criteria, entropy is used to assign weights to the criteria based on their relative importance.
3. Use the new Pythagorean entropy measure to determine the decision experts' weights in PyFS.
4. Calculate the criteria weights using the suggested entropy.
5. A Pythagorean fuzzy-integrated entropy-TOPSIS model has been utilized to solve the multi-criteria decision-making problem of selecting the location of a sustainable distribution centre.
6. A comparison of the suggested method with the existing methodologies is provided to demonstrate the precision and practicality of the integrated PyF entropy-TOPSIS approach.

This chapter is divided into five sections: Section 1 is devoted to the introduction and literature review. Section 2 presents the fundamental principles and preliminary concepts in relation to Pythagorean fuzzy sets. Section 3 proposes the novel entropy measure and proves its validity. Further, Sect. 4 presents the Pythagorean fuzzy MCDM algorithm based on Entropy-TOPSIS and discusses its application to the problem of distribution centre location selection. Section 5 discusses the result and the conclusion.

2 Preliminaries

This section goes through the fundamental definitions of fuzzy sets, intuitionistic fuzzy sets, and Pythagorean fuzzy sets. Some of the PyFS features, operations, and existing entropy measures employed in this chapter are also discussed.

Definition 1 [21] An InFS ξ in C is defined as the following:

$$\xi = \left\{ \langle b \, , \mu_{\mathrm{p}} \, (b), \nu_{\mathrm{p}}(b) \rangle \, | b \in \mathrm{C} \right\} \tag{1}$$

where

$$0 \leq \mu_{\mathrm{p}}(b) + \nu_{\mathrm{p}}(b) \leq 1 \forall b \in \mathrm{C} \tag{2}$$

and the functions $\mu_{\mathrm{p}}(b) : \mathrm{C} \rightarrow [0, 1]$ denote the degree of membership and $\nu_{\mathrm{p}}(b) : \mathrm{C} \rightarrow [0, 1]$ denote the degree of non-membership of b in C. The degree of indeterminacy of b in C indicated by λ and defined as

$$\lambda_{\mathrm{p}}(b) = 1 - \left\{ \mu_{\mathrm{p}}(b) + \nu_{\mathrm{p}}(b) \right\} \tag{3}$$

Definition 2 A PyFS proposed by Yager [1] in finite universe of discourse C is an object having the Form

$$\xi = \left\{ \langle b \, , \mu_{\mathrm{p}} \, (b), \nu_{\mathrm{p}}(b) \rangle | b \in \mathrm{C} \right\}$$

where

$$0 \leq \mu_{\mathrm{p}}^2(b) + \nu_{\mathrm{p}}^2(b) \leq 1 \forall b \in \mathrm{C} \tag{4}$$

and the functions $\mu_{\mathrm{p}}(b) : \mathrm{C} \rightarrow [0, 1]$ denote the degree of membership and $\nu_{\mathrm{p}}(b) : \mathrm{C} \rightarrow [0, 1]$ denote the degree of non-membership of b in C. The degree of indeterminacy of b in C is denoted by λ and defined as

$$\lambda_{\mathrm{p}}(b) = \sqrt{1 - \left\{ \mu_{\mathrm{p}}^2(b) + \nu_{\mathrm{p}}^2(b) \right\}} \tag{5}$$

Definition 3 [1, 4] Let $P = (\mu_{\mathrm{p}}(b), \nu_{\mathrm{p}}(b))$ and $Q = (\mu_{\mathrm{q}}(b), \nu_{\mathrm{q}}(b))$ be two PyFN, then

(a) $P \subset Q$ if $\forall \, b \in \mathrm{C} \; \mu_{\mathrm{p}}(b) \leq \mu_{\mathrm{q}}(b)$ and $\nu_{\mathrm{p}}(b) \geq \nu_{\mathrm{q}}(b)$
(b) $(P)^b = \{\langle b, \nu_{\mathrm{p}}(b), \mu_{\mathrm{p}}(b) \rangle b \in C\}$
(c) $P = Q$ if $P \subset Q$ and $Q \subset P$
(d) $P \bigcup \mathrm{Q} = \left\{ \left\langle \begin{array}{c} b, \max \left[\left(\mu_{\mathrm{p}}(b), \mu_{\mathrm{q}}(b) \right], \\ \min \left[\nu_{\mathrm{p}}(b) \geq \nu_{\mathrm{q}}(b) \right] \; | b \in \mathrm{C} \end{array} \right\rangle \right\}$
(e) $P \cap \mathrm{Q} = \left\{ \left\langle \begin{array}{c} b, \min \left[\left(\mu_{\mathrm{p}}(b), \mu_{\mathrm{q}}(b) \right], \\ \max \left[\nu_{\mathrm{p}}(b) \geq \nu_{\mathrm{q}}(b) \right] \; | b \in \mathrm{C} \end{array} \right\rangle \right\}$
(f) $P \oplus \mathrm{Q} = \left\{ \left\langle b, \sqrt{\mu_{\mathrm{p}}^2(b) + \mu_{\mathrm{q}}^2(b) - \mu_{\mathrm{p}}^2(b) \mu_{\mathrm{q}}^2(b)}, \nu_{\mathrm{p}}(b) \nu_{\mathrm{q}}(b) | b \in \mathrm{C} \right\rangle \right\}$
(g) $P \otimes \mathrm{Q} = \left\{ \left\langle \mathrm{c}, \mu_{\mathrm{p}}(b) \mu_{\mathrm{q}}(b), \sqrt{\nu_{\mathrm{p}}^2(b) + \nu_{\mathrm{q}}^2(b) - \nu_{\mathrm{p}}^2(b) \nu_{\mathrm{q}}^2(b)} | b \in \mathrm{C} \right\rangle \right\}$

Definition 4 Zhang [27] defined the operators based on Yager's [2] PyFWA and PyFWG for PyFNs $P_j = \left(\mu_{\mathrm{p}_j}, \nu_{\mathrm{p}_j} \right)$ for $(j = 1, 2, \ldots . ., n)$ as follows:

PyFWA: Let $P_j = \left(\mu_{\mathrm{p}_j}, \nu_{\mathrm{p}_j} \right)$ for $(j = 1, 2, \ldots . ., n)$ be a collection of PyFNs, then PyFWA operator is defined as:

$$\begin{aligned} \mathrm{PyFWA} \, (P_1, P_1, \ldots .., P_1) &= {}_{j=1}{}^{n} \oplus \left(w_j P_j \right) \\ &= \left(\sqrt{1 - \prod_{j=1}^{n} \left(1 - \mu_{\mathrm{p}_j}^2 \right)^{w_j}}, \prod_{j=1}^{n} \left(\nu_{\mathrm{p}_j} \right)^{w_j} \right) \end{aligned} \tag{6}$$

PyFWG: Let $P_j = \left(\mu_{\mathrm{p}_j}, \nu_{\mathrm{p}_j} \right)$ for $(j = 1, 2, \ldots . ., n)$ be a PyFNs collection, then PyFWG operator is as follows:

$$PyFWG\,(P_1, P_1, \ldots.., P_1) = {}_{j=1}{}^{n} \otimes \left(P_j\right)^{w_j}$$
$$= \left(\prod_{j=1}^{n} \left(\mu_{\mathrm{p}_j}\right)^{w_j}, \sqrt{1 - \prod_{j=1}^{n} \left(1 - \nu_{\mathrm{p}_j}^2\right)^{w_j}} \right) \tag{7}$$

Where w_j indicates the weight of P_j, such as $w_j \geq 0$ for $(j = 1, 2, \ldots. ., n)$ & $\sum_{j=1}^{n} w_j = 1$.

3 Novel Pythagorean Fuzzy Entropy Measure

Entropy is an essential tool for measuring uncertain information. Less entropy means less uncertainty. PyFS is a more powerful structure to represent an information where the existing structure fails. Therefore, establishing the measure of entropy for PyFS is significant in the present situation. The axiomatic formulation of the entropy measure of fuzzy sets was provided by De Luca and Termini [28]. Later, it was expanded to include InFS entropy by Szmidt and Kacprzyk [29]. As an extension of the entropy measure of Pythagorean fuzzy sets [1], we proposed a new concept of the fuzzy entropy measures of PyFS as follows:

Definition 5 For the Pythagorean fuzzy set $\xi = \{\langle b, \mu_{\mathrm{p}}(b), \nu_{\mathrm{p}}(b)\rangle | b \in \mathrm{C}\}$, the Pythagorean fuzzy entropy of ψ is defined as

$$E_{\mathrm{p}}\,(\xi) = \frac{1}{n} \sum_{i=1}^{n} \left[\sqrt{2} \cos \left\{ \frac{1 + \left|\mu_{\mathrm{p}}^2(b) - \nu_{\mathrm{p}}^2(b)\right|}{4} \right\} \pi \right] \tag{8}$$

Axioms for Pythagorean Fuzzy Entropy Measure

The relation $E_{\mathrm{p}} : \mathrm{PFS(C)} \rightarrow [0, 1]$ is stated as Pythagorean fuzzy entropy if it satisfies:

(i) $0 \leq E_{\mathrm{p}}\,(\xi) \leq 1$
(ii) $E_{\mathrm{p}}(\xi) = 0$, if and only if ψ is a crisp set
(iii) $E_{\mathrm{p}}(\xi) = 1$, if and only if $\mu_{\mathrm{p}}(b) = \nu_{\mathrm{p}}(b), \forall\, b \in \mathrm{C}$
(iv) $E_{\mathrm{p}}(\xi^b) = E_{\mathrm{p}}(\xi)$
(v) $E_{\mathrm{p}}(\delta) \leq E_{\mathrm{p}}(\xi)$, if δ is less fuzzy than ψ, i.e., $\mu_{\mathrm{p}}(\delta) \leq \mu_{\mathrm{p}}(\xi)$ and $\nu_{\mathrm{p}}(\delta) \geq \nu_{\mathrm{p}}(\xi)$ for $\mu_{\mathrm{p}}(\xi) \leq \nu_{\mathrm{p}}(\xi)$, or, $\mu_{\delta}(b) \geq \mu_{\xi}(b)$ and $\nu_{\delta}(b) \leq \nu_{\xi}(b)$ for $\mu_{\xi}(b) \geq \nu_{\xi}(b)\ \forall$ $b \in \mathrm{C}$.

Theorem 1 $E_{\mathrm{p}}(\xi)$ is a Pythagorean fuzzy entropy

Proof. $(E1)$ Since $0 \leq \mu_{\xi}(b), \nu_{\xi}(b) \leq 1 \Rightarrow 0 \leq \mu_{\xi}^2(b), \nu_{\xi}^2(b) \leq 1$ so that $0 \leq \left|\mu_{\xi}^2(b) - \nu_{\xi}^2(b)\right| \leq 1$

$$\Rightarrow 1 \leq 1 + \left|\mu_{\xi}^2(b) - \nu_{\xi}^2(b)\right| \leq 2$$

$$\Rightarrow \frac{\pi}{4} \leq \frac{1+\left|\mu_{\xi}^{2}(b)-\nu_{\xi}^{2}(b)\right|}{4} \leq \frac{\pi}{2}$$

$$\Rightarrow \frac{1}{\sqrt{2}} \leq \cos\left\{\frac{1+\left|\mu_{\xi}^{2}(b)-\nu_{\xi}^{2}(b)\right|}{4}\right\}\pi \leq 0$$

$$\Rightarrow 0 \leq \sqrt{2}\cos\left\{\frac{1+\left|\mu_{\xi}^{2}(b)-\nu_{\xi}^{2}(b)\right|}{4}\right\}\pi \leq 1 \quad i \Rightarrow 0 \leq E_{\mathrm{p}}(\xi) \leq 1.$$

(*E*2) If $E_{\mathrm{p}}(\xi)=0 \Rightarrow \cos\left\{\frac{1+\left|\mu_{\xi}^{2}(b)-\nu_{\xi}^{2}(b)\right|}{4}\right\}\pi = 0$

$$\Rightarrow \frac{1+\left|\mu_{\xi}^{2}(b)-\nu_{\xi}^{2}(b)\right|}{4} = \frac{1}{2} \Rightarrow \left|\mu_{\xi}^{2}(b)-\nu_{\xi}^{2}(b)\right| = 1.$$

We have, $\mu_{\xi}(b)=0$, $\nu_{\xi}(b)=1$ or $\mu_{\xi}(b)=1$, $\nu_{\xi}(b)=0$. Hence ξ is a crisp set. If ξ is a crisp set, then $E_{\mathrm{p}}(\xi)=0$.
(*E*3) If $\mu_{\xi}(b)=\nu_{\xi}(b)=\frac{1}{\sqrt{3}}$, then,

$$\Rightarrow \sqrt{2}\cos\left\{\frac{1+\left|\mu_{\xi}^{2}(b)-\nu_{\xi}^{2}(b)\right|}{4}\right\}\pi = 1$$

$$\Rightarrow E_{\mathrm{p}}(\xi) = 1$$

(*E*4) As $E_{\mathrm{p}}(\psi)=\sum_{i=1}^{n}\left[\sqrt{2}\cos\left\{\frac{1+\left|\mu_{\xi}^{2}(b)-\nu_{\xi}^{2}(b)\right|}{4}\right\}\pi\right]$

$$=\sum_{i=1}^{n}\left[\sqrt{2}\cos\left\{\frac{1+\left|\nu_{\xi}^{2}(b)-\mu_{\xi}^{2}(b)\right|}{4}\right\}\pi\right] = E_{\mathrm{p}}\left(\xi^{b}\right)$$

(*E*5) To prove axiom, we construct a function
$f(x, y)=\sqrt{2}\cos\left\{\frac{1+|y^{2}-x^{2}|}{4}\right\}\pi$, where $x, y \in [0,1]$

Case I: When $x \leq y$,then $f_1(x, y) = \sqrt{2}\cos\left\{\frac{1+|y^2-x^2|}{4}\right\}\pi$

Now we need to prove that $f_1(x, y)$ increases with x and decreases with y. Therefore,

$$\frac{\partial f_1}{\partial x} = \frac{2 \times \pi \times x}{4} \sin\left\{\frac{1+|y^2-x^2|}{4}\right\}\pi = \frac{\pi x}{2} \sin\left\{\frac{1+|y^2-x^2|}{4}\right\}\pi;$$

$$\frac{\partial f_1}{\partial y} = -\frac{2 \times \pi \times y}{4} \sin\left\{\frac{1+|y^2-x^2|}{4}\right\}\pi = -\frac{\pi y}{2} \sin\left\{\frac{1+|y^2-x^2|}{4}\right\}\pi$$

Now, when $x \leq y$, $\frac{\partial f_1}{\partial x} > 0$ and $\frac{\partial f_1}{\partial y} < 0$. Therefore, we can say that $f_1(x, y)$ increases with x and decreases with y. In other words, $\mu_\delta(b) \leq \mu_\xi(b)$ and $\nu_\delta(b) \geq \nu_\xi(b)$ for $\mu_\xi(b) \leq \nu_\xi(b) \forall\, b \in \mathrm{C}$, i.e., $E_\mathrm{p}(\delta) \leq E_\mathrm{p}(\xi)$.

Case II: When $x \geq y$,then $f_2(x, y) = \sqrt{2}\cos\left\{\frac{1+|x^2-y^2|}{4}\right\}\pi$

Now we need to prove that $f_2(x, y)$ decreases with x and increases with y. Therefore,

$$\frac{\partial f_2}{\partial x} = -\frac{2 \times \pi \times x}{4} \sin\left\{\frac{1+|x^2-y^2|}{4}\right\}\pi = -\frac{\pi x}{2} \sin\left\{\frac{1+|x^2-y^2|}{4}\right\}\pi;$$

$$\frac{\partial f_2}{\partial y} = \frac{2 \times \pi \times y}{4} \sin\left\{\frac{1+|x^2-y^2|}{4}\right\}\pi = \frac{\pi y}{2} \sin\left\{\frac{1+|x^2-y^2|}{4}\right\}\pi;$$

Now, when $x \geq y$, $\frac{\partial f_1}{\partial x} < 0$ and $\frac{\partial f_1}{\partial y} > 0$. Therefore, we can say that $f_2(x, y)$ decreases with x and increases with y. In other words, $\mu_\delta(b) \geq \mu_\xi(b)$ and $\nu_\delta(b) \leq \nu_\xi(b)$ for $\mu_\xi(b) \geq \nu_\xi(b)\ \forall\, b \in \mathrm{C}$, i.e., $E_\mathrm{p}(\delta) \leq E_\mathrm{p}(\xi)$.

4 Pythagorean Fuzzy MCDM Algorithm Based on Entropy-TOPSIS

To estimate the weight of the criterion, TOPSIS is modified using the Pythagorean entropy metric in the proposed study. Zhang and Xu [6] applied the TOPSIS method to the Pythagorean fuzzy environment to solve the MCDM problem. The chosen alternative must be the furthest away from the NIdS and nearest to the PIdS to establish the TOPSIS method. In this technique, PIdS and NIdS are initially calculated in terms of PyFNs. Following that, the distances of PIdS and NIdS from

each alternative are determined. Using the above distances, the closeness index of each alternative is determined. The rank is then determined using the closeness indices in descending order. The proposed framework uses linguistic variables to evaluate alternatives. To accommodate uncertainty, the PyFN counterparts of those options are then established using the Pythagorean fuzzy weighting scale, as shown in Table 3. Based on the inputs gathered from decision-makers in terms of linguistic factors, the Pythagorean fuzzy decision matrix of criteria is constructed.

Let $P = \{P_1, P_2, \ldots \ldots P_m\}$ indicates the collection of m options from which decision-makers will select one using a set of criteria $Z = \{Z_1, Z_2, \ldots \ldots Z_n\}$ consisting of n criteria. The various steps involved in the hybrid Pythagorean Entropy-TOPSIS method are described as follows.

Decision-Making Algorithm Using PyFS

Step 1: Estimate the Decision-Maker's Weight

Using linguistic terms, Very Very Bad (ls_1), Very Bad (ls_2), Bad (ls_3), Medium Bad (ls_4), Fair (ls_5), Good (ls_6), Very Good (ls_7), Very Very Good (ls_8), Extremely Good (ls_9), Exactly Equal (ls_{10}), determines the significance of the 'r' decision-maker and are stated in Pythagorean fuzzy numbers. Let the PyFN to rate the u^{th} decision-maker be $\theta_{u=}[\mu_u, \nu_u, \pi_u]$. So, the decision-maker's weight is determined as follows:

$$\Delta_k = \frac{\left(\mu_u + \pi_u\left(\frac{\mu_u}{\mu_u+\nu_u}\right)\right)}{\sum_{u=1}^{l}\left(\mu_u + \pi_u\left(\frac{\mu_u}{\mu_u+\nu_u}\right)\right)} \tag{9}$$

and $\sum_{u=1}^{l} \Delta_u = 1$

Step 2: Construct an aggregated PyFDM

Consider $P^{(k)} = \left(p_{ij}^{(k)}\right)_{m\times n}$ to be a fuzzy Pythagorean decision matrix for each decision-maker. Where $\Delta = \{\Delta_1, \Delta_2, \ldots \ldots \Delta_l\}$ represents the decision-maker's weight and $\sum_{k=1}^{l}\Delta_k = 1$, $\Delta_k \in [0, 1]$.

For aggregation, Zhang's [27] PyFWA operator is utilized, $P = (p_{ij})_{m\times n}$, where

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix} \tag{10}$$

where, $p_{ij} = \text{PFWA}_{\Delta}\left(p_{ij}^{(1)}, p_{ij}^{(2)}, \ldots\ldots\ldots, p_{ij}^{(l)}\right)$

$$= \left(\sqrt{1 - \prod_{k=1}^{l}\left(1-\mu_{ij}^2\right)^{\Delta_k}}, \prod_{k=1}^{l}\left(\nu_{ij}\right)^{\Delta_k}\right) \tag{11}$$

Here $p_{ij} = \left(\mu_{I_i}\left(z_j\right) \bullet \nu_{I_i}\left(z_j\right), \pi_{I_i}\left(z_j\right)\right)$
$i = 1, 2, \ldots\ldots, m$, and $j = 1, 2, \ldots\ldots, n$

Step 3: Evaluate Criteria Weight Utilizing Proposed Entropy Measure

Here, the weight vector $W = (\varpi_1^e, \varpi_2^e, \varpi_3^e, \ldots., \varpi_n^e)$ is obtained using the Pythagorean fuzzy entropy measure E_p for criteria C_j $(j = 1, 2, \ldots, n)$ proposed in Eq. (8), where $\varpi_j^e \geq 0$ and $\sum_{j=1}^{n} \varpi_j^e = 1$. The Pythagorean fuzzy entropy measure for each column of P is evaluated using Eqs. (8, 9, and 11), and thus to determine the weights ϖ_j^e of criteria C_j $(j = 1, 2, \ldots, n)$, the following expression is used:

$$\varpi_j^e = \frac{1 - E_p\left(C_j\right)}{n - \sum_{j=1}^{n} E_p\left(C_j\right)} \tag{12}$$

Step 4: Weighted Aggregated PyFDM

As a result, the weight vector W can be calculated and aggregated with the fuzzy Pythagorean decision matrix P [30] to create the weighted PyFDM, H.

$$\bar{H} = W' \otimes \bar{P} = W^T \otimes \left[\bar{z}_{ij}\right]_{m\times n} = \left[\bar{\bar{z}}_{ij}\right] \tag{13}$$

Where $W = \left(\varpi_1^e, \ldots\ldots \varpi_n^e\right)$ and $\bar{\bar{z}}_{ij} = \langle \bar{\bar{\mu}}_{ij}, \bar{\bar{\nu}}_{ij} \rangle = \langle 1-(1-\mu_{ij})^{\varpi_j^e}, \nu_{ij}^{\varpi_j^e} \rangle$, $w_j > 0$

Step 5: Evaluate PIdS and NIdS

Apply TOPSIS to the newly constructed aggregated PyFDM to resolve the MCDM problem. According to Zhang and Xu's [4] definitions, the extrema score values are utilized to determine the Positive Ideal Solution (PIdS) and Negative Ideal Solution (NIdS).

$$\mathrm{PIS}_j = \max_i \left\{s\left(\alpha_{ij}\right)\right\} \tag{14}$$

$$\mathrm{NIS}_j = \min_i \left\{s\left(\alpha_{ij}\right)\right\} \tag{15}$$

Where $s(\alpha_{ij})$ is PyFN's score function

Step 6: Determine the Separation Measures

$$\delta_{PQ^+} = \frac{1}{n}\sum_{i=1}^{n} \max\left\{\left|\mu_P{}^2 - \mu_Q^{+2}\right|, \left|\nu_P{}^2 - \nu_Q^{+2}\right|\right\} \tag{16}$$

$$\delta_{PQ^-} = \frac{1}{n}\sum_{i=1}^{n} \max\left\{\left|\mu_P{}^2 - \mu_Q^{-2}\right|, \left|\nu_P{}^2 - \nu_Q^{-2}\right|\right\} \tag{17}$$

Step 7: Evaluate the Closeness Coefficient

The coefficient of closeness for each alternative is determined w.r.t. the Pythagorean fuzzy ideals using the following expression:

$$C_j = \frac{\delta_{PQ^-}}{\delta_{PQ^+} + \delta_{PQ^-}} \tag{18}$$

Where $0 \leq C_j \leq 1$ and $j = 1, \ldots, m$

Step 8: Ranking of Alternatives

Depending on their C_j value, the alternatives are ordered in descending order.

Application of the Entropy-TOPSIS Technique to Choose the Distribution Centre Location

Ten evaluation criteria and four alternatives were considered for the selection of distribution centre location based on a thorough literature review, which are as follows. Human resources availability (C_{HR}), i.e., the availability of labourers to work at the distribution centre; i.e., the needed space must be suitable for a distribution facility. Ample surrounding space must also be available for future development. Distribution centres must have access to a variety of modes of transportation to enable easy and quick transit because transportation is one of the most crucial factors determining their success. Storage Convenience (C_{STC}): the proximity of the distribution centre to the location where storage services are offered is important. Distance to suppliers (C_{DS}) has to do with suppliers' closeness to the distribution centre. To reduce transit time and ensure supply, the distribution centre should be close to the suppliers. Distance to market (C_{DM}) has to do with how close the marketplaces are to the distribution hub. Due to transportation considerations, distribution centres should be close to markets. Distance to airport (C_{DA}) is the distribution centre's close proximity to airports. Air transportation is appropriate for the delivery of urgent packages and precious cargo. As a result, the distribution centre should be near airports. Distance to other transport facilities (C_{DTF}) is the distribution centre's close access to roads, ports, and railways. Highways make it simple and quick for businesses to convey their goods. The issue of large shipments, particularly economical freight, may be solved by transportation by sea or interior waterways. Railways should be nearby since they are a reliable and affordable form of transportation. Land cost (C_{LC}) refers to the amount of property wherever the distribution centre will be situated. The minimization criterion is the land cost because it influences the overall investment expenses. Logistics cost (C_{LGC}) is the expense incurred to make the items available to the customers. This cost has an impact on the outcomes of corporate performance; hence, minimization is a requirement. Labour cost (C_{LBC}) serves as the remuneration for the labourers who work in the distribution centre. Labour cost is the minimization criterion since it influences company performance outcomes.

Step 1: Estimate the Decision-Maker's Weight

The Pythagorean fuzzy technique integrates the weights of each decision-maker using Eq. (9), and the decision-maker's significance is evaluated. The linguistic

Table 1 Linguistic terms

LTs	*PyFNs*	
Extra significant	0.9	0.2
Significant	0.8	0.35
Average	0.65	0.45
Insignificant	0.35	0.8
More insignificant	0.2	0.9

Table 2 Weight of decision-makers

Decision makers	Linguistics term	Weight (Δ)
I	Very Important	0.4044
II	Medium	0.2583
III	Important	0.3372

Table 3 Linguistic terms

ls_1	0	1
ls_2	0.2	0.9
ls_3	0.35	0.8
ls_4	0.45	0.65
ls_5	0.55	0.55
ls_6	0.65	0.45
ls_7	0.8	0.35
ls_8	0.9	0.2
ls_9	1	0
ls_{10}	0.1965	0.1965

terms used for rating decision makers are displayed in Tables 1 and 2 lists the resulting decision-makers's weights.

Step 2: Construct Aggregated PyFDM

The aggregated Pythagorean fuzzy decision matrix is constructed using the linguistic terms shown in Table 3; the results of the decision-maker's evaluation of each of the four distribution centre locations under consideration are displayed in Table 4. Based on the viewpoint of the typical decision-maker, Eq. (11) was utilized to generate the aggregated PyF decision matrix, which is shown in Table 5.

Step 3: Evaluate Criteria Weight Utilizing Proposed Entropy Measure

In order to calculate the proposed entropy measure for PyFS using Eq. (8) and to estimate the weights for each criteria using Eq. (12), decision-makers calculated 10 independent criteria. The aggregated results of their assessments are shown in Table 6, along with the resulting entropy measure and weights.

Step 4: Weighted Aggregated PyFDM

Using Eq. (13), a weighted aggregate PyFDM is constructed, and thus Table 7 is obtained.

Table 4 Weight of alternative's based on criteria

Criteria	Distribution centre location	Decision-makers		
		I	II	III
C_{HR}	T_1	ls_6	ls_7	ls_6
	T_2	ls_8	ls_7	ls_7
	T_3	ls_6	ls_6	ls_6
	T_4	ls_5	ls_6	ls_4
C_{IR}	T_1	ls_6	ls_6	ls_6
	T_2	ls_7	ls_6	ls_7
	T_3	ls_7	ls_7	ls_6
	T_4	ls_6	ls_6	ls_6
C_{STC}	T_1	ls_7	ls_6	ls_7
	T_2	ls_7	ls_7	ls_6
	T_3	ls_7	ls_7	ls_6
	T_4	ls_6	ls_6	ls_6
C_{DS}	T_1	ls_5	ls_6	ls_5
	T_2	ls_7	ls_7	ls_8
	T_3	ls_6	ls_7	ls_7
	T_4	ls_7	ls_6	ls_6
C_{DM}	T_1	ls_6	ls_7	ls_6
	T_2	ls_8	ls_6	ls_7
	T_3	ls_6	ls_5	ls_5
	T_4	ls_7	ls	ls_6
C_{DA}	T_1	ls_6	ls_7	ls_6
	T_2	ls_7	ls_6	ls_8
	T_3	ls_6	ls_6	ls_6
	T_4	ls_5	ls_4	ls_6
C_{DTF}	T_1	ls_6	ls_7	ls_7
	T_2	ls_8	ls_6	ls_7
	T_3	ls_7	ls_6	ls_6
	T_4	ls_4	ls_4	ls_6
C_{LC}	T_1	ls_6	ls_6	ls_7
	T_2	ls_8	ls_8	ls_7
	T_3	ls_7	ls_6	ls_7
	T_4	ls_6	ls_7	ls_7
C_{LGC}	T_1	ls_7	ls_6	ls_7
	T_2	ls_8	ls_6	ls_8
	T_3	ls_6	ls_6	ls_6
	T_4	ls_5	ls_4	ls_6
C_{LBC}	T_1	ls_6	ls_6	ls_6
	T_2	ls_7	ls_6	ls_7
	T_3	ls_6	ls_6	ls_6
	T_4	ls_4	ls_4	ls_6

Table 5 Decision-maker's evaluation in PyFNs in decision matrix

C/A	T_1	T_2	T_3	T_4
C_{HR}	<0.6942,0.425>	<0.8491,0.28>	<0.6503,0.45>	<0.5458,0.559>
C_{IR}	<0.6503,0.45>	<0.7733,0.371>	<0.7556,0.384>	<0.6503,0.45>
C_{STC}	<0.7733,0.371>	<0.7556,0.384>	<0.7556,0.384>	<0.6503,0.45>
C_{DS}	<0.5761,0.525>	<0.8461,0.284>	<0.7523,0.387>	<0.7218,0.407>
C_{DM}	<0.6942,0.425>	<0.8294,0.297>	<0.5941,0.508>	<0.7218,0.407>
C_{DA}	<0.6942,0.425>	<0.8264,0.301>	<0.6503,0.45>	<0.5735,0.531>
C_{DTF}	<0.7556,0.384>	<0.8294,0.321>	<0.7218,0.412>	<0.5412,0.586>
C_{LC}	<0.7176,0.415>	<0.871,0.259>	<0.772,0.375>	<0.75,0.392>
C_{LGC}	<0.7733,0.371>	<0.8683,0.267>	<0.6503,0.45>	<0.5735,0.543>
C_{LBC}	<0.6503,0.45>	<0.7733,0.371>	<0.6503,0.45>	<0.3646,0.4830>

Table 6 Evaluation of proposed entropy measure and criterion weights

Criteria	C_{HR}	C_{IR}	C_{STC}	C_{DS}	C_{DM}	C_{DA}	C_{DTF}	C_{LC}	C_{LGC}	C_{LBC}
Entropy (E_p)	0.7415	0.8694	0.8098	0.6985	0.7545	0.7686	0.7121	0.8088	0.6991	0.8519
Weights (ϖ_j^e)	0.1131	0.0571	0.0832	0.1319	0.1074	0.1012	0.1259	0.0837	0.1316	0.0648

Table 7 Aggregated weighted PyFDM

C/A	A_1	A_2	A_3	A_4
C_{HR}	0.1254, 0.9077	0.1925, 0.8659	0.112, 0.9136	0.0854, 0.9363
C_{IR}	0.0582, 0.9554	0.0813, 0.9449	0.0773, 0.9467	0.0582, 0.9554
C_{STC}	0.1161, 0.9207	0.1106, 0.9234	0.1106, 0.9234	0.0837, 0.9357
C_{DS}	0.1070, 0.9185	0.2187, 0.847	0.1681, 0.8822	0.1552, 0.8881
C_{DM}	0.1195, 0.9121	0.1730, 0.8777	0.0923, 0.9298	0.1284, 0.9079
C_{DA}	0.1130, 0.917	0.1624, 0.8855	0.1009, 0.9223	0.0826, 0.9379
C_{DTF}	0.1460, 0.8994	0.1813, 0.8793	0.1347, 0.9045	0.0843, 0.9413
C_{LC}	0.1332, 0.9053	0.2067, 0.8583	0.1539, 0.8949	0.1451, 0.8994
C_{LGC}	0.1545, 0.8949	0.2049, 0.8612	0.1120, 0.9136	0.0918, 0.9332
C_{LBC}	0.1120, 0.9136	0.1545, 0.8949	0.1120, 0.9136	0.0500, 0.9209

Step 5: Evaluate PIdS and NIdS

Now, using Eqs. (14 and 15), PIdS and NIdS are estimated in accordance with Zhang and Xu's [4] definitions, and the outcomes are displayed in Table 8.

Step 6, 7, and 8 Calculate the separation measures of each alternative from the PIdS, NIdS, Closeness Coefficient, separation measure, and relative closeness. The coefficients are assessed using Eqs. (16)–(18), and the findings are shown in Table 9. The separation measure is calculated, and alternatives are thus ordered in decreasing order.

Table 8 PIdS and NIdS for Pythagorean fuzzy set

Criteria	A+	A-	Criteria	A+	A-
C_{HR}	0.1292	0.0561	C_{DA}	0.9583	0.9111
	0.9111	0.9583	0.0784	0.1532	
	0.1532	0.0784	0.0755	0.1464	
C_{IR}	0.1275	0.0921	C_{DTF}	0.9292	0.9129
	0.9129	0.9292	0.1281	0.1504	
	0.1504	0.1281	0.1201	0.1439	
C_{STC}	0.116	0.084	C_{LC}	0.921	0.936
	0.921	0.936	0.139	0.117	
	0.139	0.117	0.1324	0.1102	
C_{DS}	0.1174	0.0556	C_{LGC}	0.919	0.958
	0.919	0.958	0.1408	0.0793	
	0.1408	0.0793	0.1356	0.0759	
C_{DM}	0.124	0.065	C_{LBC}	0.913	0.95
	0.913	0.95	0.151	0.0923	
	0.151	0.0923	0.14363	0.0889	

Table 9 Separation measure and ranking of alternative

Alternatives	S+	S−	C_{i+}	Rank
T1	0.32854	0.29507	0.473	3
T2	0.1864	0.43721	0.701	1
T3	0.30691	0.3167	0.508	2
T4	0.38921	0.23441	0.376	4

Table 10 Entropy comparison

Entropy measure	Rank
$E_1(C)$ [31]	T2 > T3 > T1 > T4
$E_2(C)$ [32]	T2 > T3 > T1 > T4
$E_3(C)$ [33]	T2 > T3 > T1 > T4
$E_P(C)$ Proposed	T2 > T3 > T1 > T4

5 Result and Discussion

From the above table, alternative A_2 was chosen as the best option based on the selected criteria, and alternative A_4 is the least preferred distribution centre location as it received the worst rating from the four alternatives taken into consideration for the study. The rank obtained through the application of the proposed Entropy-TOPSIS method is compared with the few Pythagorean fuzzy entropy measures taken from the latest literature. The results thus obtained through the different entropy measures are shown in Table 10.

The following factors make the new PyFS-based entropy measures far more effective:

- The new Pythagorean entropy measure is in perfect agreement with the axiomatic definition of entropy measures.
- TOPSIS is defined using the information for PyFS.

- A hybrid entropy-TOPSIS approach is applied to solve the multi-criteria decision-making problem.
- The use of linguistic variables and PyFSs assures that this approach can better address the uncertainty in the decision-making method.
- The suggested integrated methodology, which uses a newly defined Pythagorean entropy measure, is more appropriate and efficient for ranking alternatives than the current MCDM approaches.

6 Conclusion

Research pertaining to the selection of sustainable logistics centre locations has suggested that the application of the Pythagorean fuzzy entropy measure (PyFEM) can be advantageous. The use of PyFEM yields a set of weights, which are then used to evaluate the multiple criteria associated with the selection of a sustainable logistics centre location. This approach eliminates the need for subjective and time-consuming assessments, which may be prone to bias. Additionally, the application of PyFEM can help identify complex relationships between criteria and provide insights into the best location for a sustainable logistics centre. PyFS provides a tremendous ability to effectively regulate uncertainties and fuzziness. A hybrid entropy-TOPSIS model approach to solving the Pythagorean fuzzy MCDM problem is considered. The selection of a sustainable logistics centre location is used as an example, and Pythagorean fuzzy inputs in the form of linguistic data are used. A comparison of the existing entropy measures with the proposed measure reflects the efficiency of the proposed method. PyFS has the capability to handle the uncertainty in the system. The proposed methodology can be used to choose the best option when there are competing criteria in a variety of real-world decision-making situations in a fuzzy Pythagorean environment. The Pythagorean fuzzy set is a generalization for more effectively incorporating uncertainty in the data, and it is similar to other current fuzzy sets. In addition, a decision-making algorithm is applied to resolve decision-making complications when a group of experts is involved. The results of the case show the potential of the recommended entropy in the field of decision-making with linguistic data. The proposed measurement can be used as a component of new objective methods for setting attribute weights utilizing Pythagorean fuzzy sets in an upcoming analysis on multi-criteria decision-making. Future applications for the proposed Pythagorean fuzzy method are numerous. Additionally, the proposed measurement can be employed in subsequent studies by including various fuzzy systems into decision matrices and employing them to address various MCDM issues with unknown weights.

References

1. Yager, R. R. (2013). Pythagorean fuzzy subsets. In *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, pp. 57–61. https://doi.org/10.1109/ifsa-nafips.2013.6608375
2. Yager, R. R. (2014). Pythagorean membership grades in multicriteria decision making. *IEEE Transactions on Fuzzy Systems, 22*(4), 958–965. https://doi.org/10.1109/tfuzz.2013.2278989
3. Yager, R. R., & Abbasov, A. M. (2013). Pythagorean membership grades, complex numbers, and decision making. *International Journal of Intelligent Systems, 28*(5), 436–452. https://doi.org/10.1002/int.21584
4. Zhang, X., & Xu, Z. (2014). Extension of TOPSIS to multiple criteria decision making with Pythagorean fuzzy sets. *International Journal of Intelligent Systems, 29*(12), 1061–1078. https://doi.org/10.1002/int.21676
5. Okatan, B. S., Peker, I., & Birdogan, B. (2019). An integrated DEMATEL - ANP - VIKOR approach for food distribution center site selection: A case study of Georgia. *Journal of Management Marketing and Logistics, 6*(1), 10–20. https://doi.org/10.17261/Pressacademia.2019.1030
6. Eckhardt, J., & Rantala, J. (2012). The role of intelligent logistics centres in a multimodal and cost-effective transport system. *Procedia Social and Behavioral Sciences, 48*(2), 612–621. https://doi.org/10.1016/j.sbspro.2012.06.1039
7. Chung, S. H., Chan, H. K., & Chan, F. T. S. (2013). A modified genetic algorithm for maximizing handling reliability and recyclability of distribution centers. *Expert Systems with Applications, 40*(18), 7588–7595. https://doi.org/10.1016/j.eswa.2013.07.056
8. Wei, H., Ye, H., Longwei, T., & Yuan, L. (2015). A novel profit-allocation strategy for SDN enterprises. *Enterprise Information Systems, 11*, 4–16. https://doi.org/10.1080/17517575.2015.1053417
9. Agrebi, M., Abed, M., & Omri, M. N. (2017). ELECTRE I based relevance decision-makers feedback to the location selection of distribution centers. *Journal of Advanced Transportation, 2017*, 1–10. https://doi.org/10.1155/2017/7131094
10. Han, Q., Li, W., Xu, Q., Song, Y., Fan, C., & Zhao, M. (2022). Novel measures for linguistic hesitant Pythagorean fuzzy sets and improved TOPSIS method with application to contributions of system-of-systems. *Expert Systems with Applications, 199*, 117088. https://doi.org/10.1016/j.eswa.2022.117088
11. Akram, M., et al. (2022). An integrated ELECTRE-I approach for risk evaluation with hesitant Pythagorean fuzzy information. *Expert Systems with Applications, 200*, 116945–116945. https://doi.org/10.1016/j.eswa.2022.116945
12. Ulutaş, A., Karakuş, C. B., & Topal, A. (2020). Location selection for logistics center with fuzzy SWARA and CoCoSo methods. *Journal of Intelligent & Fuzzy Systems, 38*(4), 4693–4709. https://doi.org/10.3233/jifs-191400
13. Shafiee, M., et al. (2022). A causality analysis of risks to perishable product supply chain networks during the COVID-19 outbreak era: An extended DEMATEL method under Pythagorean fuzzy environment. *Transportation Research Part E: Logistics and Transportation Review, 163*, 102759. https://doi.org/10.1016/j.tre.2022.102759
14. Erdogan, M., & Ayyildiz, E. (2022). Investigation of the pharmaceutical warehouse locations under COVID-19—A case study for Duzce, Turkey. *Engineering Applications of Artificial Intelligence, 116*, 105389. https://doi.org/10.1016/j.engappai.2022.105389
15. Sagnak, M., et al. (2021). Sustainable collection center location selection in emerging economy for electronic waste with fuzzy best-worst and fuzzy TOPSIS. *Waste Management, 127*, 37–47. https://doi.org/10.1016/j.wasman.2021.03.054
16. Yang, C., et al. (2022). A linguistic Pythagorean hesitant fuzzy MULTIMOORA method for third-party reverse logistics provider selection of electric vehicle power battery recycling. *Expert Systems with Applications, 198*, 116808. https://doi.org/10.1016/j.eswa.2022.116808

17. Yin, C., et al. (2022). Site selection framework of rail transit photovoltaic power station under interval-valued Pythagorean fuzzy environment. *Energy Reports, 8*, 3156–3165. https://doi.org/10.1016/j.egyr.2022.02.073
18. Saeidi, P., et al. (2022). Evaluate sustainable human resource management in the manufacturing companies using an extended Pythagorean fuzzy SWARA-TOPSIS method. *Journal of Cleaner Production, 370*, 133380. https://doi.org/10.1016/j.jclepro.2022.133380
19. Li, Y., et al. (2022). Evaluation of dispatching results of power system with high penetration of renewable energy based on Pythagorean fuzzy set and TOPSIS. *Energy Reports, 8*, 524–532. www.sciencedirect.com/science/article/pii/S2352484722015748, https://doi.org/10.1016/j.egyr.2022.08.134
20. Ayyildiz, E. (2023). A novel Pythagorean fuzzy multi-criteria decision-making methodology for E-Scooter charging station location-selection. *Transportation Research Part D: Transport and Environment, 111*, 103459. https://doi.org/10.1016/j.trd.2022.103459
21. Atanassov, K. T. (1986). Intuitionisitic fuzzy sets. *Fuzzy Sets and Systems, 20*(1), 87–96. https://doi.org/10.1016/s0165-0114(86)80034-3
22. Alipour, M., et al. (2021). A new Pythagorean fuzzy-based decision-making method through entropy measure for fuel cell and hydrogen components supplier selection. *Energy, 234*, 121208. https://doi.org/10.1016/j.energy.2021.121208
23. Yu, C., Shao, Y., Wang, K., & Zhang, L. (2019). A group decision making sustainable supplier selection approach using extended TOPSIS under interval-valued Pythagorean fuzzy environment. *Expert Systems with Applications, 121*, 1–17. https://doi.org/10.1016/j.eswa.2018.12.010
24. Parveen, T., Arora, H. D., & Alam, M. (2020a). Intuitionistic fuzzy hybrid multi-criteria decision-making approach with TOPSIS method using entropy measure for weighting criteria. *Strategic System Assurance and Business Analytics*, 351–364. https://doi.org/10.1007/978-981-15-3647-2_26
25. Parveen, T., Arora, H. D., & Alam, M. (2020b). Intuitionistic fuzzy Shannon entropy weight based multi-criteria decision model with TOPSIS to analyze security risks and select online transaction method. *Advances in Computing and Intelligent Systems*, 1–17. https://doi.org/10.1007/978-981-15-0222-4_1
26. Han, Q., Li, W., Lu, Y., Zheng, M., Quan, W., & Song, Y. (2020). TOPSIS method based on novel entropy and distance measure for linguistic Pythagorean fuzzy sets with their application in multiple attribute decision making. *IEEE Access, 8*, 14401–14412. https://doi.org/10.1109/ACCESS.2019.2963261
27. Zhang, X. (2015). A novel approach based on similarity measure for Pythagorean fuzzy multiple criteria group decision making. *International Journal of Intelligent Systems, 31*(6), 593–611. https://doi.org/10.1002/int.21796
28. De Luca, A., & Termini, S. (1972). A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory. *Information and Control, 20*(4), 301–312. https://doi.org/10.1016/s0019-9958(72)90199-4
29. Szmidt, E., & Kacprzyk, J. (2001). Entropy for intuitionistic fuzzy sets. *Fuzzy Sets and Systems, 118*(3), 467–477. https://doi.org/10.1016/s0165-0114(98)00402-3
30. Atanassov, K. T. (1999). *Open problems in intuitionistic fuzzy sets theory*. Intuitionistic Fuzzy Sets, pp. 289–291. https://doi.org/10.1007/978-3-7908-1870-3_6
31. Thakur, P., Kaczyńska, A., Gandotra, N., Saini, N., & Sałabun, W. (2022). The application of the new Pythagorean fuzzy entropy to decision-making using linguistic terms. *Procedia Computer Science, 207*, 4525–4534. https://doi.org/10.1016/j.procs.2022.09.516
32. Gandotra, N., Kizielewicz, B., Anand, A., Bączkiewicz, A., Shekhovtsov, A., Wątróbski, J., Rezaei, A., & Sałabun, W. (2021). New Pythagorean entropy measure with application in multi-criteria decision analysis. *Entropy, 23*(12), 1600. https://doi.org/10.3390/e23121600
33. Yang, M.-S., & Hussain, Z. (2018). Fuzzy entropy for Pythagorean fuzzy sets with application to multicriterion decision making. *Complexity, 2018*, 1–14. https://doi.org/10.1155/2018/2832839

Improved Jaya Algorithm with Inertia Weight Factor

Sonal Deshwal, Pravesh Kumar, and Sandeep Kumar Mogha

1 Introduction

The constrained optimization problem in mathematical form can be defined as:

$$\begin{aligned} &\text{Min } f(X) : X = [x_1, x_2, \ldots, x_N] \in R^N \\ &\text{s.t.} : g_j(X) \leq 0,\ j = 1, 2, \ldots, m \\ &\quad h_k(X) = 0, k = 1, 2, \ldots, n \\ &\quad \text{low}_i \leq x_i \leq \text{upp}_i, i = 1, 2, \ldots, N \end{aligned}$$

of efficiency, that is, the objective is to find the candidate solution $X^* \in R^N$ to minimize the function while satisfying all constraints. low_i represents the lower bound and upp_i represents the upper bound for x_i.

The optimization strategy is crucial in determining how to maximize or minimize a function in terms of the decision variables. Numerous nonlinear constraints provide a huge number of solution spaces for many real-world issues. These issues are nonconvex, complex, and come with a significant computing cost. Therefore, it is exceedingly difficult to solve problems with such a vast number of variables and constraints. Additionally, local optimal solutions produced from several conventional approaches might not always suggest the best choice. To solve these issues, the researchers propose a number of metaheuristic optimization strategies [1] which are extremely effective in solving extremely complex situations [2]. The invention of simple, flexible, and inexpensive metaheuristic algorithms has received more attention from researchers.

S. Deshwal · S. K. Mogha
Department of Mathematics, Chandigarh University, Ludhiana, Punjab, India

P. Kumar (✉)
Department of Mathematics, REC Bijnor (AKTU Lucknow), Bijnor, Uttar Pradesh, India

V. Garg et al. (eds.), *Women in Soft Computing*, Women in Engineering and Science,
https://doi.org/10.1007/978-3-031-44706-8_5

The two main categories of population-based methods are algorithms based on swarm intelligence (SI) and evolutionary algorithms (EA). Different EA take ideas from natural processes, including reproduction, mutation, recombination, and selection. The basis for each of these EA is the candidate's population-level survival fitness. Several well-known evolutionary algorithms include DE (Differential Evolution) [3], Bacterial Foraging Optimization (BFO) [4], and GA (Genetic Algorithm) [5]. There are numerous algorithms based on swarm intelligence that are well known. Grey Wolf Optimizer (GWO) [6], PSO (Particle Swarm Optimization) [7], ACO (Ant Colony Optimization) [8], Fire Fly (FF) technique [9], SFL (Shuffled Frog Leaping) [10], Artificial Bee Colony [11], and others are a few of them. In addition to algorithms based on swarm intelligence and evolution, certain other algorithms are built on the fundamentals of many different natural phenomena. Biogeography-Based Optimization (BBO) [12], APA [13], IRA [14], GFA [15], Harmony Search (HS) method [16], Charged System Search Algorithm (CSSA) [17] and others are among them.

It is also essential for one to understand that many of these metaheuristic techniques rely on a small number of (often fine-tuned) parameters, the impacts of which might drastically alter the algorithmic functioning hinge on the individual optimization problem. As opposed to that, from the viewpoint of a user, it would be desirable to have a method that requires the fewest parameters to generalize its usage without necessitating extra labor to adjust the parameter value for each particular circumstance. As a result, one current path in metaheuristic algorithms is to design self-adaptive techniques that can be utilized to instantly resolve any optimization problem. One illustration of this pattern is the Teaching Learning Based Optimization (TLBO) [18] approach. Only two parameters are needed: the population size and the maximum number of generations, which appear to be prerequisites for all swarm intelligence algorithms. TLBO has recently served as an inspiration for several algorithms, including the Jaya algorithm, which was constructed on a similar idea.

The Jaya algorithm is a recently established method proposed by Rao [19]. It is a population-based technique developed to address both constrained and unconstrained issues. The Jaya algorithm version is used to address both single and multi-objective real-world optimization issues. A few newly developed Jaya algorithm forms and their applications are presented in [20–28]. The most notable feature of Jaya is that when an individual's position in the population is updated, both the one with the global best function value and the one with the global worst function value are taken into consideration. Jaya provides a broad search area in the search space by using both the global best and global worst positions. Using Jaya's organizational structure, local minimums may be avoided. However, there is space for improvement in terms of exploration, accuracy, and speed to a better answer. In order to increase its exploration, exploitation capacities, and convergence

speed, the inertia weight factor is presented in this study. The effectiveness of the upgraded Jaya method is examined for parameter optimization of literature-available benchmark functions. According to the experimental findings, the Jaya with inertia weight performs better than the Jaya and three other well-known algorithms named GA, PSO, and DE.

Following is the organization of the remaining sections: A brief description of Jaya is provided in Sect. 2. The suggested W-Jaya algorithm is briefly described in Sect. 3. Results and comparisons are presented in Sect. 4, and Sect. 5 concludes with inferences made from the current study.

2 Jaya Algorithm

Jaya algorithm is a stochastic search technique based on a population that addresses constrained as well as unconstrained optimization problems. Every solution obtained by Jaya is termed as particle. Each particle within the searching zone chooses the best objective solution and avoids the worst.

The name Jaya means "victory" in Sanskrit. Its cornerstone is the "survival of the fittest" theory of natural selection. This demonstrates how the best worldwide outcomes are pursued while the worst ones are disregarded within the Jaya population. As a result, better solutions are placed over the worst solutions in each iteration.

Let $f(x)$ be the objective function that has been minimized (or maximized). Consider a population with a size of N (i.e., $i = 1, 2, 3, \ldots, N$) and design variables D (i.e., $j = 1,2,3, \ldots, D$). Let $X_{\text{worst},\,j}(g)$ and $X_{\text{best},\,j}(g)$ be the current global worst and optimum vectors in the search space, then Eq. 1 creates the new position for any vector $X_{i,\,j}(g)$.

$$\begin{aligned} X_{i,\,j}\,(g+1) = X_{i,\,j}(g) &+ \text{rand}_1{}^{*}\left(X_{\text{best},\,j}(g) - |X_{i,\,j}(g)|\right) \\ &- \text{rand}_2{}^{*}\left(X_{\text{worst},\,j}(g) - |X_{i,\,j}(g)|\right) \end{aligned} \tag{1}$$

where rand_1 and rand_2 are any uniform random numbers with values between [0,1]. g means the current generation number. $X_{i,\,j}(g+1)$ represents $X_{i,\,j}(g)$'s most recent value. $X_{i,\,j}(g+1)$ is retained if it has better fitness value than $X_{i,\,j}(g)$. If not, $X_{i,\,j}(g)$ will be kept for the upcoming iteration.

Working of Jaya Algorithm

- Start by setting up the parameters for the algorithm and the optimization problem.
- Generate the initial population and then compute the fitness function.
- Set g = 1 (first generation)
- **While** g < MaxGen (Maximum generation)
- Identify the best and worst individuals.
- **For** i= 1:PS (Population size)
- **For** j=1:D (dimension)
- Update the initial population with the help of equation 1.
- **End For**
- **If** solution $X_{i,j}(g+1)$ is better than $X_{i,j}(g)$ then (According to the fitness function's solution)
- Set $X_{i,j}(g) = X_{i,j}(g+1)$
- **Else**
- Set $X_{i,j}(g) = X_{i,j}(g)$
- **End If**
- **End For**
- Set g = g+1
- **End while**

3 Jaya Algorithm with Inertia Weight Factor

The challenging aspect of optimization algorithm is striking a balance between diversification and intensification. In order to develop a balance between exploration and exploitation more effectively and to obtain a fast convergence speed, an inertia weight factor is combined with Jaya and its name is suggested as "W-Jaya Algorithm." Next, inertia weight factor and W-Jaya are defined in detail as follows:

3.1 Inertia Weight Factor

The idea of an inertia weight factor is generally used in the PSO algorithm to balance diversification and intensification. The optimization technique may balance exploitation and exploration using time-varying inertia weight (TVIW). The idea of TVIW has been successfully implemented in PSO [29–32], where it has been proven that a high TVIW makes exploration easier, while a relatively small TVIW makes exploitation easier. The inertia weight may be changed to dynamically alter the search capacity. Therefore, it would be important to test this modification on a variety of problems and compare its performance to the original Jaya algorithm and other optimization algorithms.

The following is the mathematical equation for the time-varying inertia weight:

$$W(g) = W_{\max} - \frac{(W_{\max} - W_{\min}) \times g}{g_{\max}} \tag{2}$$

where $W(g)$ is the weight of inertia at g iteration, $W_{\max}$ is the inertia weight's maximum value, $W_{\min}$ is the inertia weight's minimum value, and $g_{\max}$ is the maximum number of iterations.

3.2 Planned Jaya with Inertia Weight Factor

Step 1: The W-Jaya algorithm's settings are established in the first phase of the run. The algorithmic parameters are the population size N, $W_{\max}$,$W_{\min}$ and $g_{\max}$.

Step 2: The initial population of the W-Jaya method is constructed using Eq. 3 and saved in an augmented matrix of size $N \times D$, where N is the population size & D denotes the number of design variables, as shown in Eq. 4.

$$X_{ij} = \text{low}_j + \left(\text{upp}_j - \text{low}_j\right) \times \text{rand}, \text{ where } i = 1, 2, \ldots, N \&\ \ j = 1, 2, \ldots, D \tag{3}$$

The uniform function rand produces random values between 0 and 1, with low_j and upp_j denoting the lower and upper boundaries of the search region, respectively.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \cdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix} \tag{4}$$

The objective function f(x) for each solution is evaluated. Thereafter, $X^g_{\text{best},j}$and $X^g_{\text{worst},j}$are determined, whereas $X^g_{\text{best},j}$is the global best vector in the search space for generation g and $X^g_{\text{worst},j}$ is the global worst vector.

Step 3: Evolution of W-Jaya in process. All choice variables for all solutions in the X are modified iteratively using Eq. 5.

$$\begin{aligned} X^{g+1}_{i,j} = {} & W(g) \times \text{rand}_1 \times X^g_{i,j} + \text{rand}_2 \times \left(X^g_{\text{best},j} - |X^g_{i,j}|\right) - \text{rand}_3 \\ & \times \left(X^g_{\text{worst},j} - |X^g_{i,j}|\right) \end{aligned} \tag{5}$$

where rand_1, rand_2 and rand_3 are any uniform random numbers with values between [0,1]. The updated value for $X^g_{i,j}$ is $X^{g+1}_{i,j}$.

Step 4: The X solutions given in (4) will be revised after each iteration g. As indicated in Eq. 6, $X_{i,j}^{g}$ will be replaced by the newly produced vector $X_{i,j}^{g+1}$ if it has the best optimized value, else previous value $X_{i,j}^{g}$ remains the same for the upcoming generation. Then the updated values of X are given by:

$$X_{i,j}^{g+1} = \begin{cases} X_{i,j}^{g+1}, if\ f\left(X_{i,j}^{g+1}\right) < f\left(X_{i,j}^{g}\right) \\ X_{i,j}^{g} \quad \text{else} \end{cases} \tag{6}$$

Step 5: Steps 3 and 4 are continued until the W-Jaya algorithm reaches the termination criteria, which is often the maximum number of generation $g_{\max}$.

3.3 Flow Chart of W-Jaya

A flowchart is a graphical representation of a process, system, or algorithm using symbols, shapes, and connecting lines to illustrate the sequence of steps, decision points, and interactions involved. W-Jaya's flowchart is shown in Fig. 1.

Explanation of Steps:

1. *Start*: This is the initial point of the process. The flowchart begins here.
2. *Initialize W-Jaya algorithm's parameters*: This step involves setting up the parameters required for the W-Jaya algorithm.
3. *Create the first population*: Generate the initial population for the optimization problem using Eq. 3.
4. *Locate the population's best and worst solutions*: Evaluate the fitness of each solution to determine the best and worst solutions based on the problem's objective function. The best solution is the one with the highest fitness, while the worst solution has the lowest fitness.
5. *Using Eq. 5, update the existing solution*: Apply Eq. 5 to update the existing solution.
6. *Is new solution better than old solution*: It's checked whether the newly updated solution is better than the old solution by comparing the fitness of the new solution with that of the old solution.
   I. *Accept the new solution and replace it*: If the new solution is better (higher fitness) than the old solution, accept the new solution and replace the old one with it.
   II. *Keep the old solution*: If the new solution is not better than the old solution, retain the old solution.
7. *Is there fulfillment of the termination requirement*: Check whether the termination criteria for the algorithm have been met. These criteria determine when the algorithm should stop running.

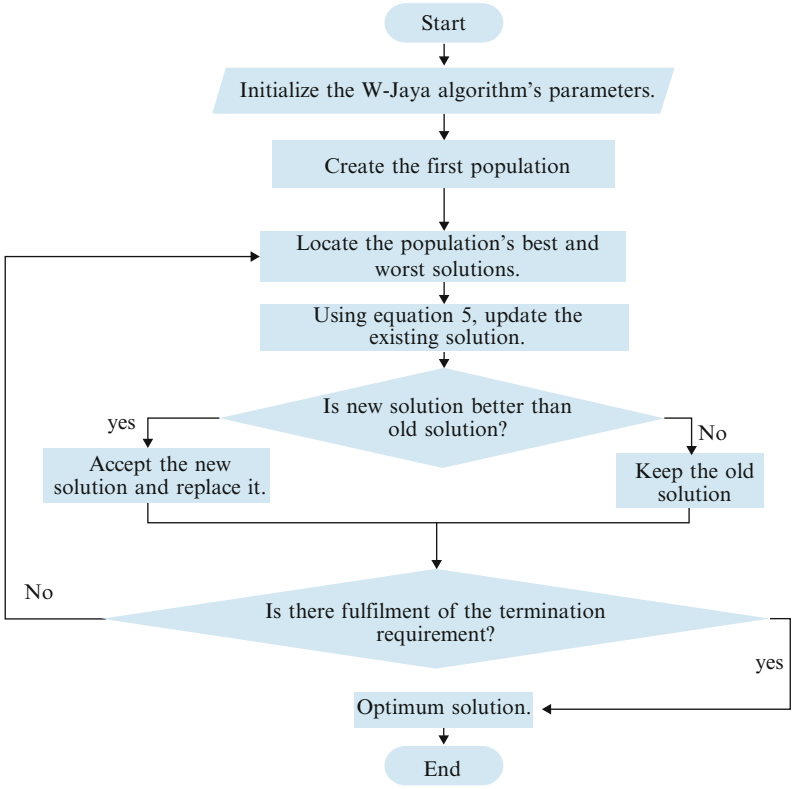


Fig. 1 Flowchart of W-Jaya

I. *Report the optimum solution*: If the termination criteria are met, report the best solution found by the algorithm as the optimum solution for the given problem.
II. *If not, repeat steps from locating the population's best and worst solutions*: If the termination criteria are not met, the process goes back to Step 4 to locate the best and worst solutions in the updated population.

8. Steps 4 through 7 are repeated until the termination criteria are fulfilled.
9. *End*: The process ends.

4 Experimental Results and Discussion

A total of 30 standard functions are being used to test the effectiveness of the W-Jaya algorithm. Out of which 20 are unconstrained and 10 are constrained. These functions are the traditional ones that are frequently used nowadays. We've selected

these test functions so that we can compare our results to those of the standard JAYA approach and three meta-heuristics methods (GA, PSO, DE). This section covers the interpretation of experimental data and a thorough explanation of functions, parameter settings, and statistical analysis of results.

4.1 Unconstrained and Constrained Benchmark Problems

There are two main categories of optimization problems: constrained and unconstrained problems. The optimization algorithm of the unconstrained type searches for an objective function's minimum or maximum value. As shown in Appendix 1, the three different unconstrained benchmark function types—unimodal, multimodal, and fixed-dimension multimodal—are presented in Tables 5, 6, and 7, respectively, where Dim denotes the function's dimension, range is the variation in functional value, and $f_{\min}$ denotes the optimal one.

A constrained type of problem has a more complicated approach. The optimum solution for this type of problem must meet the conditions (constraints) given for the problem. In this study, the W-Jaya method and four additional algorithms were used to optimize 10 constrained benchmark problems. The article's Appendix 2 has more details on the functions. Table 3 shows the optimal average values. It is clear that the algorithm has achieved ideal or very close to optimal results, and its performance has been acceptable.

Constrained Handling Technique

A way of handling methodology is required to apply constraint conditions in the optimization process while optimizing constrained problems. The literature offers a variety of techniques for dealing with constraints [33–35]. This chapter uses the method of applying a penalty to the value of the objective function. The mathematical form of penalty is shown in Eq. 7 to Eq. 10. The fitness function $Q(x)$ is described as the sum of a penalty term and the objective function $f(x)$. $P(x)$ is defined as the constraint values given in Eq. 8. The total number of constraints is t, the number of inequality constraints is m, and the number of equality constraints is n. To manage equality constraints, they are often converted to inequality constraints, as shown in Eq. 8. The parameter δ represents a small positive number (10^{-4} in this chapter).

$$Q(x) = f(x) + 10^{20} \sum_{1}^{t} (P_t(x))^2 \tag{7}$$

where

$$P_t(x) = \left\{ \begin{array}{ll} \max\left(0, g_t(x)\right), & if \;\; 1 \le t \le m \\ \max\left(0, \left(|h_t(x)| - \delta\right)\right), & if \;\; 1 \le t \le n \end{array} \right\} \tag{8}$$

Table 1 Setup of parameters

S. no	Name	Parameter setting
1	Maximum Generation	500
2	Population Size (N)	30
3	Total Run	20
4	$rand_1$, $rand_2$ & $rand_3$	[0,1]
5	W_{max}	0.9
6	W_{min}	0.4

$$g_t(x) \leq 0, t = 1, 2, 3, 4, \ldots, m \tag{9}$$

$$h_t(x) = 0, t = 1, 2, 3, 4, \ldots, n \tag{10}$$

4.2 Parameter Setting

The test's parameter settings are shown in Table 1. The maximum number of generations and the population size were chosen to be 500 and 30, respectively, to ensure that all functions in optimization problems could be adequately compared. The suggested method was implemented in MATLAB version R2016a on a laptop with the following specifications: Windows 11, 12th Gen Intel(R) Core (TM) i7-1255U @ 1.70 GHz, 16GB RAM, 512GB SSD.

4.3 Results and Comparison

In this part, the effectiveness of the suggested W-Jaya is assessed through comparison to other optimization techniques, such as Jaya, PSO, GWO, and DE (Table 2). We have executed each technique on each function 20 times in order to minimize the effects of randomness. After the same number of runs, the average results are compared to those of other methods. Here, it is evident that when compared to alternatives, the recommended W-Jaya obtained the desired results for all functions. In addition, Table 4 displays the results of a nonparametric Wilcoxon statistical test.

The results of Table 3 show that W-Jaya outperforms all other methods for functions G01, G02, G03, G05, G06, and G10. Its performance in the G04, G07, and G09 functions is on par with DE and better than Jaya, PSO, and GWO. W-Jaya performs better than Jaya but is equivalent to PSO, GWO, and DE for function G08. It should be noted that boldface is used to highlight the best outcomes.

Table 2 Performance comparisons between unconstrained test problems

F	W-Jaya			JAYA			PSO			GWO			DE		
	Best	Avg.	SD	Best	Avg.	SD	Best	Avg.	SD	Best	Avg.	SD	Best	Avg.	SD
F_1	**4.28E-59**	1.33E-55	3.33E-55	1.554	2.675	0.701	3.59E-05	0.0028	0.0039	4.28E-28	6.59E-28	6.34E-05	7.5E-16	8.2E-14	5.9E-14
F_2	**6.78E-33**	6.13E-32	7.74E-32	0.95632	1.45564	0.741	0.00072	1.0467	2.5858	5.42E-17	7.18E-17	0.02901	9.6E-09	1.5E-09	9.9E-10
F_3	**8.73E-19**	6.81E-19	7.85E-19	24010.14	31159.67	4490.01034	6177.693	9187.638	1938.147	9.23E-07	3.29E-06	79.14958	9.8E-11	6.8E-11	7.4E-11
F_4	8.96E-20	2.55E-17	4.69E-17	10.0308	16.79550	2.735819	9.574	15.352	3.7463	2.25E-07	5.61E-07	1.315088	**0**	0	0
F_5	23.55747	23.70867	0.127588	185046.1	600421.9	302341.4	94.657	96.718	60.11559	26.3454	26.81258	69.90499	**0**	0	0
F_6	**0**	0	0	1933.624	4024.116	1129.137	0.0001	0.000102	8.28E-05	0.66682	0.816579	0.000126	**0**	0	0
F_7	**9.20E-05**	0.000401	0.000241	0.09652	0.16243	0.074184	0.05896	0.13658	0.054003	0.000577	0.002213	0.100286	0.00765	0.00463	0.0012
F_8	−4501.88	−3181.37	463.9153	−8254.56	−5969.11	1350.122	−6549.4	−6123.1	−4087.44	−4233.627	−4841.29	1152.814	**−11230.2**	−11080.1	574.7
F_9	**0**	0	0	207.7219	236.1326	22.25992	40.1552	65.4532	20.861	1.14E-13	0.310521	47.35612	62.7	69.2	38.8
F_{10}	**8.98E-16**	8.88E-16	0	1.989334	2.272152	0.614193	1.4661	2.0847	0.46418	1.04E-13	1.06E-13	0.077835	8.9E-09	9.7E-08	4.2E-08
F_{11}	**0**	0	0	0.992235	1.023728	0.015853	0.00018	0.018618	0.016824	8.22E-09	0.004485	0.006659	**0**	0	0
F_{12}	**1.57E-25**	1.3E-25	5.2E-25	9.5791	17.86898	7.033638	0.00254	0.7806	0.84438	0.054699	0.053438	0.020734	8.9E-15	7.9E-15	8E-15
F_{13}	**2.78E-28**	2.35E-28	1.67E-28	73110.69	569222.7	503253.8	0.000967	0.006675	0.008907	0.8134	0.654464	0.004474	6.78E-15	5.1E-14	4.8E-14
F_{14}	**0.99807**	0.998002	4.8E-16	**0.99807**	0.998158	0.000298	2.876	3.627168	2.560828	2.9821	4.042493	4.252799	**0.99807**	0.998004	3.3E-16
F_{15}	**0.00030**	0.000389	9.35E-05	0.00034	0.000458	4.909E-05	0.00043	0.000577	0.000222	0.00031	0.000337	0.000625	**0.00030**	0.00033	4.5E-2
F_{16}	**−1.0316**	−1.03172	4.57E-7	**−1.0316**	−1.03162	1.223E-05	**−1.0316**	−1.03163	6.25E-16	**−1.0316**	−1.03163	−1.03163	**−1.0316**	−1.03163	3.1E-13
F_{17}	**0.398**	0.398742	0.000582	**0.398**	0.397887	5.6882E-17	**0.398**	0.397887	0	**0.398**	0.397889	0.397887	**0.398**	0.397887	9.9E-09
F_{18}	**3**	3.000743	0.001344	**3**	3.001354	0.001694	**3**	3	1.34E-15	**3**	3.000028	3	**3**	3	2E-15
F_{19}	**−3.86**	−3.86084	9.87E-18	**−3.86**	−3.86278	9.10E-16	**−3.86**	−3.86278	2.58E-15	**−3.86**	−3.86263	−3.86278	N/A	N/A	N/A
F_{20}	**−3.32**	−3.32	0.02309	−3.24	−3.16	0.06347	−3.57	−3.23	0.06052	−3.19	−3.28	−3.26	N/A	N/A	N/A

Table 3 Performance comparisons between constrained test problems

F	W-Jaya			Jaya			PSO			GWO			DE		
	Best	Avg.	SD	Best	Avg.	SD	Best	Avg.	SD	Best	Avg.	SD	Best	Avg.	SD
G01	**−15**	−15	0	−11.99	−11.672	−0.548	−12.57	−12.33	1.86E-15	−11.85	−11.738	0.116992	−14.99	−14.98	2.2E-04
G02	**−0.803619**	−0.80358	5.97E-05	−0.3293	−0.3293	3.02E-05	−0.329	−0.329	4.05E-05	−0.45415	−0.4542	3.77964E-06	−0.803431	−0.8039	3.3E-05
G03	**−1.0002**	−1.00011	1.6E-04	0	0	0	−0.597	−0.503	0.136983						
	0	0	0	0	0	0									
G04	**−30665.539**	−30665.54	0	−30665.5	−30665.53	1.51E-05	−30665.5	−30665.536	1.4E-06	−30615.2	−30585.4	29.71166	**−30665.539**	−30665.538	5.4E-06
G05	**5126.4972**	5126.4972	9.58692E-13	1990.261237	1990.261207	1.5E-05	4.37E+15	4.37E+15	0	1990.261237	1990.261207	1.49672E-05	5126.497	5126.497	7.2E-05
G06	**−6961.81388**	−6961.81388	9.43827E-13	−6961.81	−6961.81	3.03E-06	−696.82	−696.82	2.38472E-13	−6871.22	−6784.68	58.88986	−6961.813	−6847.26	140.1
G07	**24.30620**	24.30620	3.2E-05	40.8407	53.05258	19.42991	25.07	25.07	3.69778E-15	53.1074	62.03288	5.816949	**24.30620**	24.30620	1.2E-05
G08	**−0.095825**	−0.095825	0	−2.614	−2.6149	4.65765E-16	**−0.095825**	−0.095825	0	−2.6109	−2.61107	0.002529	**−0.095825**	−0.095825	0
G09	**680.6300**	680.6300	2.3E-04	684.1901	685.2175	37.25513	681.87	681.87	1.17677E-13	638.5639	638.4587	1.411203	**680.6300**	680.6300	1.3E-04
G10	**7049.248**	7049.248	1.883E-12	7059.748	7059.748	1.883E-12	7056.56	7056.56	1.883E-12	7062.56	7062.56	1.888E-12	7050.36	7051.80	7.9E-01

Table 4 Wilcoxon test results for unconstrained optimization problem

Functions	Wilcoxon test			
	1/2	1/3	1/4	1/5
F_1	+	+	+	+
F_2	+	+	+	+
F_3	+	+	+	+
F_4	+	+	+	−
F_5	+	+	+	−
F_6	+	+	+	=
F_7	+	+	+	+
F_8	+	+	+	−
F_9	+	+	+	+
F_{10}	+	+	+	+
F_{11}	+	+	+	=
F_{12}	+	+	+	+
F_{13}	+	+	+	+
F_{14}	=	+	+	=
F_{15}	=	=	=	=
F_{16}	=	=	=	=
F_{17}	=	=	=	=
F_{18}	=	=	=	=
F_{19}	=	=	=	+
F_{20}	+	+	+	+

Comparing the performance of different algorithms on a set of benchmark instances using the Wilcoxon test is a common statistical method in the field of optimization. The Wilcoxon test is a nonparametric test used to determine if there are significant differences between the performances of two or more algorithms. Here's a step-by-step guide on how to perform this comparison:

1. Collect the results of the different algorithms on the set of benchmark instances. For each algorithm, you should have a set of performance measurements (e.g., optimum value) for each benchmark instances.
2. Select the algorithm that will be compared with others. For example, W-Jaya has been compared with Jaya, PSO, GWO, and DE in Table 4. We put W-Jaya as number one and Jaya, PSO, GWO, and DE as 2, 3, and 4, accordingly.
3. Now reintroduce the signs. If for a function algorithm 1 performed better than algorithm 2, assign a positive (+) sign to the corresponding function. If algorithm 1 performed worse than algorithm 2, assign a negative (−) sign. If the performance is identical for the algorithms 1 and 2, assign a equal to (=) sign.

The Wilcoxon test indicates that for functions F_1, F_2, F_3, F_7, F_9, F_{10}, F_{12}, F_{13}, and F_{20}, W-Jaya surpasses all other algorithms. Its performance for functions F_6 and F_{11} is comparable to DE but superior to Jaya, PSO, and GWO. W-Jaya performs worse than DE for functions F_4, F_5, and F_8 but better than Jaya, PSO, and GWO. W-Jaya performs better than Jaya but less than PSO, GWO, and DE for function F_9.

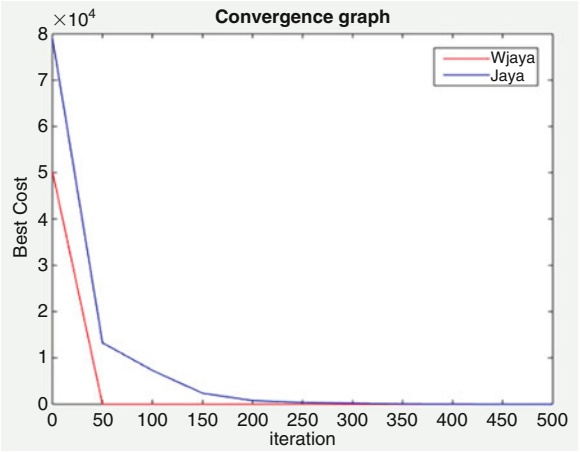


Fig. 2 Convergence graph of F_1

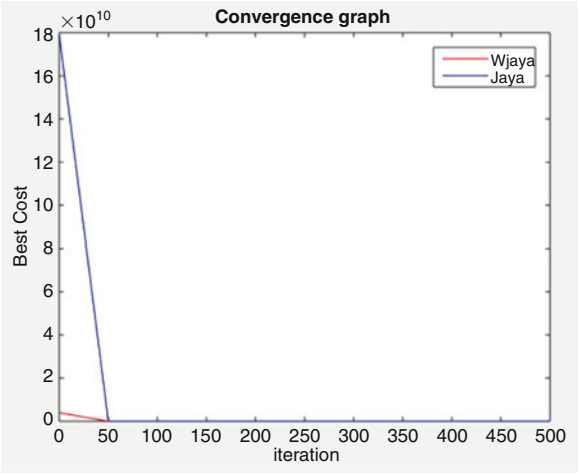


Fig. 3 Convergence graph of F_2

For functions F_{15}, F_{16}, F_{17}, and F_{18}, comparable to Jaya, PSO, GWO, and DE. For F_{19}, Jaya, PSO, and GWO are equivalent. For function F_{14}, W-Jaya is on par with Jaya but better than DE, PSO, and GWO.

4.4 Convergence Graph

This section presents the Jaya and suggested W-Jaya convergence graphs for the functions F_1, F_2, F_9, and F_{11}. Fig. 2, 3, 4, and 5 show the convergence graphs for the functions F_1, F_2, F_9, and F_{11}, respectively. These graphs over Jaya make it simple to examine the rate of W-Jaya's convergence.

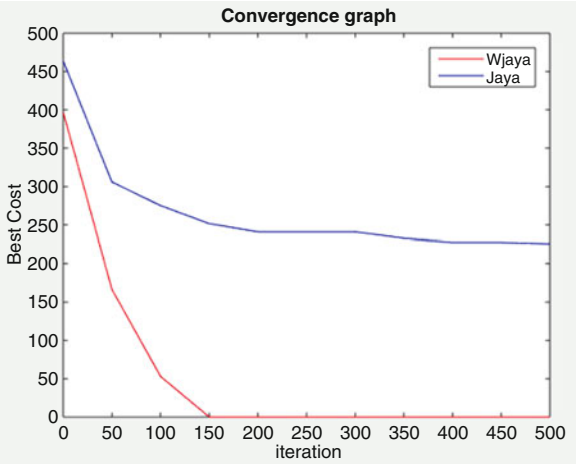


Fig. 4 Convergence graph of F_9

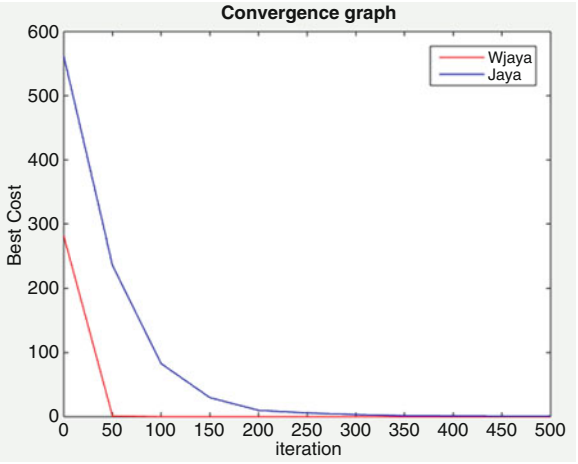


Fig. 5 Convergence graph of F_{11}

By the graphs it can be easily seen that W-Jaya's rate of convergence on the benchmark functions is unquestionably faster than that of traditional Jaya.

5 Conclusion

In this chapter, an Improved Jaya (W-Jaya) algorithm with time-varying inertia weight factor is introduced to determine unconstrained and constrained optimization problems. The proposed idea of time-varying inertia weight (TVIW) with Jaya helps to improve the exploitation and exploration capacity of the algorithm. The performance of W-Jaya algorithm has been tested on 20 unconstrained and 10 constrained test functions and compared with well-known optimization techniques,

such as conventional Jaya, PSO, GWO and DE. Finally, the results reveal that the W-Jaya offers high-quality solutions in a reasonable timeframe and enhances the exploitation-exploration capabilities of Jaya.

A.1 Appendices

A.1.1 Appendix 1: List of Unconstrained Test Problems (Tables 5, 6, and 7)

A.1.2 Appendix 2: List of Constrained Test Problems (Table 8)

Problem-1: G01

$$\text{Min } f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

Table 5 Unimodal benchmark problems

Function	Dim	Range	$f_{\min}$
$F_1 = \sum_{i=1}^{n} x_i^2$	30	[−100,100]	0
$F_2 = \sum_{i=1}^{n} \mid x_i \mid + \prod_{i=1}^{n} x_i$	30	[−10,10]	0
$F_3 = \sum_{i=1}^{n} \left(\sum_{i=1}^{n} x_i \right)^2$	30	[−100,100]	0
$F_4 = \max \{ \mid x_i \mid , 1 \le i \le n$	30	[−100,100]	0
$F_5 = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right]$	30	[−30,30]	0
$F_6 = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[−100,100]	0
$F_7 = \sum_{i=1}^{n} i x_i^4 + \text{random}\,[0, 1)$	30	[−1.28,1.28]	0

Table 6 Multimodal benchmark problems

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

Subject to:

$$
\begin{aligned}
g_1(x) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0 \\
g_2(x) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0 \\
g_3(x) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0 \\
g_4(x) &= -8x_1 + x_{10} \le 0 \\
g_5(x) &= -8x_2 + x_{11} \le 0 \\
g_6(x) &= -8x_3 + x_{12} \le 0 \\
g_7(x) &= -2x_4 - x_5 + x_{10} \le 0 \\
g_8(x) &= -2x_6 - x_7 + x_{11} \le 0 \\
g_9(x) &= -2x_8 - x_9 + x_{12} \le 0
\end{aligned}
$$

Table 7 Fixed multidimensional benchmark problems

Function	Dim	Range	$f_{\min}$
$F_{14} = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right)^{-1}$	2	[−65,65]	1
$F_{15} = \sum_{i=1}^{11} \left[a_i - \frac{x_1 (b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$	4	[−5,5]	0.00030
$F_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	2	[−5,5]	−1.0316
$F_{17} = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	2	[−5,5]	0.398
$F_{18} = \left\{ \begin{array}{l} \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2\right) \right] \times \\ \left[30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2\right) \right] \end{array} \right\}$	2	[−2,2]	3
$F_{19} = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij} \left(x_j - p_{ij} \right)^2 \right)$	3	[1,3]	−3.86
$F_{20} = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij} \left(x_j - p_{ij} \right)^2 \right)$	6	[0,1]	−3.32

Table 8 Constrained benchmark problems

Function name	Number of constraints	Dimension	Global best
G01	9	13	−15
G02	2	20	0.803619
G03	1	3	−1
G04	6	5	−30 665.539
G05	5	4	5126.4981
G06	2	2	−6961.81388
G07	8	10	24.3062091
G08	2	2	0.095825
G09	4	7	680.6300573
G10	6	8	7049.3307

Properties:

$0 \le x_i \le 1 \ (i = 1, 2, \ldots\ldots, 9), 0 \le x_i \le 100 \ (i = 10, 11, 12), \ 0 \le x_{13} \le 1$
$x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$
Constraints $g_1, g_2, g_3, g_7, g_8, g_9$ are active.

Problem-2: G02

$$\text{Max } f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sum_{i=1}^{n} i x_i^2} \right|$$

Subject to:

$$g_1(x) = 0.75 - \prod_{i=1}^{D} x_i \leq 0$$
$$g_2(x) = \sum_{i=1}^{D} x_i - 7.5D \leq 0$$

Properties:

$$0 \leq x_i \leq 10\,(i = 1, 2, \ldots.., D)\,,\ D = 20$$

The known best value is $f(x^*) = 0.803619$
Constraint g_1 is an active constraint.

Problem-3: G03

$$\text{Min } f(x) = -\left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i$$

Subject to:

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

Properties:

$$0 \leq x_i \leq 10\,(i = 1, 2, \ldots.., n)$$
$$x^* = \left(\frac{1}{\sqrt{n}}\right), n = 10$$

Problem-4: G04

$$\text{Min } f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

Subject to:

$$g_1(\mathrm{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5$$
$$g_2(\mathrm{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 - 0.0021813x_3^2$$
$$g_3(\mathrm{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4$$
$$0 \le g_1(x) \le 92$$
$$90 \le g_2(x) \le 110$$
$$20 \le g_3(x) \le 25$$

Properties:

$$78 \le x_1 \le 102, 33 \le x_2 \le 45, 27 \le x_i \le 45\,(i = 3, 4, 5)$$
$$x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$$

Problem-5: G05

$$\text{Min } f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + \left(\frac{0.000002}{3}\right)x_2^3$$

Subject to:

$$g_1(x) = -x_4 + x_3 - 0.55 \le 0$$
$$g_2(x) = -x_3 + x_4 - 0.55 \le 0$$
$$h_3(x) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$
$$h_4(x) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$
$$h_5(x) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

Properties:

$$0 \le x_i \le 1200\,(i = 1, 2)\,, -0.55 \le x_2 \le 0.55\,(i = 3, 4)\,.$$
$$x^* = (679.9463, 1026.067, 0.1188764, -0.3962336)\,.$$

Problem-6: G06

$$\text{Min } f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

Subject to:

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$$
$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$$

Properties:

$$13 \leq x_i \leq 100, 0 \leq x_2 \leq 100$$
$$x^* = (14.095, 0.84296)\,.$$

Problem-7: G07

$$\text{Min } f(x) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 5)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

Subject to:

$$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$
$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$
$$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$
$$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$
$$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$
$$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1 x_2 + 14x_5 - 6x_6 \leq 0$$
$$g_7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$
$$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

Properties:

$-10 \leq x_i \leq 10 \ \ (i = 1, 2, \ldots\ldots, 10)$
$x^* = \Big(2.171996, 2.363683, 8.773926, 5.095984, 0.99065481, 1.430574,$
$1.321644, 9.828726, 8.280092, 8.375927\Big)$
Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

Problem-8: G08

$$\text{Max } f(x) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

Subject to:

$$g_1(x) = x_1^2 - x_2 + 1 \leq 0$$
$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

Properties:

$$0 \leq x_i \leq 10\,(i = 1, 2)\,.$$
$$x^* = (1.2279713, 4.2453733)\,.$$

Problem-9: G09

$$\text{Min } f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 0x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

Subject to:

$$g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$
$$g_2(x) = -282 + 7x_1 + +3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$
$$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$$
$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

Properties:

$$-10 \leq x_i \leq 10\,(i = 1, 2, \ldots\ldots, 7)$$
$$x^* = \begin{pmatrix} 2.330499, 1.951372, -0.4775414, 4.365726, \\ -0.624487, 1.038131, 1.5942270 \end{pmatrix}.$$

Problem-10: G10

$$\text{Min } f(x) = x_1 + x_2 + x_3$$

Subject to:

$$g_1(x) = -1 + 0.0025\,(x_4 + x_6) \leq 0$$
$$g_2(x) = -1 = 0.0025\,(x_5 + x_7 - x_4) \leq 0$$
$$g_3(x) = -1 + 0.01\,(x_8 - x_5) \leq 0$$
$$g_4(x) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$$
$$g_5(x) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$
$$g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$

Properties:

$$-100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000\,(i = 2, 3)\,,$$
$$10 \leq x_i \leq 1000\,(i = 4, \ldots, 8)\,.$$
$$x^* = \left(\begin{pmatrix} 579.19, 1360.13, 5109.5979, 182.0174, 295.5985, \\ 217.9799, 286.40, 395.5979 \end{pmatrix}\right).$$

References

1. Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. Wiley.
2. Koutitas, G., & Kaveh, A. (2016). *Advances in metaheuristic algorithms for optimal design of structures* (2nd ed.). Springer Nature.
3. Storn, R., & Price, K. (1997). Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.
4. Passino, K. M. (2012). Bacterial foraging optimization. *International Journal of Swarm Intelligence Research, 1*(1), 1–16.
5. Wright, A. H. (1991). Genetic algorithms for real parameter optimization. In *Foundations of genetic algorithms* (Vol. 1, pp. 205–218). Elsevier.
6. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.
7. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings, IEEE international conference on neural networks, 1995* (pp. 1942–1948).
8. Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *Computational Intelligence Magazine, IEEE, 1*, 28–39.
9. Yang, X. S., & He, X. (2013). Firefly algorithm: recent advances and applications. *International Journal of Swarm Intelligence, 1*(1), 36.
10. Eusuff, M., Lansey, K., & Pasha, F. (2006). Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Engineering Optimization, 38*(2), 129–154.
11. Karaboga, D., & Basturk, B. (2007). *Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems* (LNAI 4529) (pp. 789–798). Springer.
12. Simon, D., & Member, S. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6), 702–713.
13. Xie, L., Zeng, J., & Cui, Z. (2009). General framework of artificial physics optimization algorithm. In *2009 world congress on nature & biologically inspired computing (NaBIC)* (pp. 1321–1326).
14. Chuang, C. L., & Jiang, J. A. (2007). Integrated radiation optimization: Inspired by the gravitational radiation in the curvature of space-time. In *2007 IEEE Congress on Evolutionary Computaton CEC 2007, no. 3157* (pp. 3157–3164).

15. Zheng, M., Liu, G. X., Zhou, C. G., Liang, Y. C., & Wang, Y. (2010). Gravitation field algorithm and its application in gene cluster. *Algorithms for Molecular Biology, 5*(1), 1–11.
16. Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation, 76*(2), 60–68.
17. Kaveh, A., & Talatahari, S. (2010). A novel heuristic optimization method: Charged system search. *Acta Mechanica, 213*(3–4), 267–289.
18. Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *CAD Computer Aided Design, 43*(3), 303–315.
19. Rao, R. V. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations, 7*(1), 19–34.
20. Mishra, S., & Ray, P. K. (2016). Power quality improvement using photovoltaic fed DSTATCOM based on Jaya optimization. *IEEE Transactions on Sustainable Energy, 99*, 1–9.
21. Gong, C. (2017). An enhanced Jaya algorithm with a two group adaption. *International Journal of Computational Intelligence Systems, 10*, 1102–1115.
22. Yu, K., Liang, J., Qu, B., Chen, X., & Wang, H. (2017). Parameters identification of photovoltaic models using an improved JAYA optimization algorithm. *Energy Conversion and Management, 150*, 742–753.
23. Gao, K., Zhang, Y., Sadollah, A., Lentzakis, A., & Su, R. (2017). Jaya harmony search and water cycle algorithms for solving large-scale real-life urban traffic light scheduling problem. *Swarm and Evolutionary Computation, 37*, 58–72.
24. Rao, R. V., & More, K. C. (2017). Design optimization and analysis of selected thermal devices using self-adaptive Jaya algorithm. *Energy Conversion and Management, 140*, 24–35.
25. Singh, S. P., Prakash, T., Singh, V., & Babu, M. G. (2017). Analytic hierarchy process based automatic generation control of multi-area interconnected power system using jaya algorithm. *Engineering Applications of Artificial Intelligence, 60*, 35–44.
26. Rao, R. V., & Saroj, A. (2017). Economic optimization of shell-and-tube heat exchanger using jaya algorithm with maintenance consideration. *Swarm and Evolutionary Computation, 116*, 473–487.
27. Rao, R. V., & Saroj, A. (2017). A self-adaptive multi-population based jaya algorithm for engineering optimization. *Swarm and Evolutionary Computation, 37*, 1–37.
28. Rao, R. V., & Saroj, A. (2018). Multi-objective design optimization of heat exchangers using elitist-jaya algorithm. *Energy Systems, 9*, 305–341.
29. Xin, J., Chen, G., & Hai, Y. (2009). A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. In *International joint conference on computational sciences and optimization 1* (pp. 505–508).
30. Umapathy, P., Venkataseshaiah, C., & Arumugam, M. S. (2010). Particle swarm optimization with various inertia weight variants for optimal power flow solution. *Discrete Dynamics in Nature and Society*.
31. Arumugam, M. S., & Rao, M. V. C. (2006). On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems. *Discrete Dynamics in Nature and Society*.
32. Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., & Abraham, A. (2011). Inertia weight strategies in particle swarm optimization. In *Third world congress on nature and biologically inspired computing* (pp. 633–640).
33. Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*(2), 113–127.
34. Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 186*(2-4), 311–338.
35. Mezura-Montes, E., & Coello, C. A. C. (2011). Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation, 1*(4), 173–194.

A State-of-the-Art Literature Review on Drone Optimization

Vanita Garg and Dimple Kumari

1 Introduction

Drones, also known as unmanned aerial vehicles (UAVs), are flying devices that can be controlled remotely or programmed to fly autonomously. They come in various sizes and shapes, from small handheld models to larger ones that resemble airplanes or helicopters.

Drones have become increasingly popular in recent years due to their versatility and usefulness in various industries such as photography, videography, agriculture, search and rescue, and surveillance. They are also commonly used for recreational purposes such as racing and aerial photography.

One of the advantages of using drones is their ability to access and navigate areas that are difficult or dangerous for humans. For example, drones can be used to inspect infrastructure such as bridges, power lines, and wind turbines without risking the lives of workers. They can also provide valuable data in emergency situations such as natural disasters or accidents.

Overall, drones have the potential to be a valuable tool in various industries, but their use should be regulated to ensure they are used safely and responsibly.

Abraham Kareem is the inventor of drone. He is regarded as " the father of UAV technology."

V. Garg (✉) · D. Kumari
Galgotias University, Greater Noida, India

V. Garg et al. (eds.), *Women in Soft Computing*, Women in Engineering and Science,
https://doi.org/10.1007/978-3-031-44706-8_6

2 Drone Anatomy

Frame
The frame is the body of the drone, which holds all the other components together. It is usually made of lightweight materials like carbon fiber or plastic.

Motors
The motors are the powerhouses of the drone. They spin the propellers to generate lift and propel the drone in the air. There are usually four motors on a drone, one for each propeller.

Propellers
The propellers are attached to the motors and spin rapidly to create lift and thrust. They come in different shapes and sizes, depending on the drone's intended use.

Battery
The battery provides power to the drone's electronics, motors, and propellers. It is usually a rechargeable lithium-ion battery.

Flight Controller
The flight controller is the brain of the drone. It receives input from the pilot's controller and uses sensors like accelerometers, gyroscopes, and GPS to maintain stability and control the drone's movements.

Camera
Many drones come with a built-in camera, which can be used for aerial photography or videography. The camera is usually mounted on a gimbal to keep it steady during flight.

Transmitter
The transmitter is the handheld controller that the pilot uses to fly the drone. It sends signals to the flight controller, which translates them into the drone's movements.

3 Four Main Types of Drones

3.1 Multi-Rotor Drones

Multi-rotor drones offer an economical and straightforward way to get a bird's eye view. They have limited capacity to adjust location and design, making them ideal for aerial photography and monitoring. These are referred to as multi-rotor because they possess more than one motor, with quadcopters being the most common type of multi-rotor drone.

3.2 Fixed-Wing Drones

A fixed wing drone has a single, solid wing that is designed to resemble an airplane, providing lift through aerodynamic forces rather than rotors pointing upward. Consequently, this type of drone only requires energy to propel it forward and not to remain aloft. This renders them efficient in terms of energy usage.

3.3 Single-Rotor Drones

Single-rotor drones are known for their robustness and longevity. They resemble real helicopters in both form and function. A single-rotor aircraft has one large rotating section and a tail rotor to maintain directional control and steadiness.

3.4 Fixed-Wing Hybrid VTOL

Combining the advantages of fixed-wing and rotor-based structures, the Wing Hybrid VTOL drone class offers a powerful combination. This particular drone has rotors affixed to the main frame which enable it to hover and ascend/descend in a vertical direction. This new grouping of mixed breeds is just a demand right now, but as technology progresses, this option can become far more widespread in the near future.

4 Optimization

Optimization means finding the best possible solution to a problem or situation using the available resources. It involves maximizing or improving certain criteria or objectives, such as efficiency, cost, quality, or speed.

Optimization is used in many areas, such as business, engineering, mathematics, and computer science, to achieve the best possible results with the resources at hand.

4.1 Optimization in Drones

Drone optimization means making drones work better by improving their design, components, sensors, software, and control systems. This can involve making drones faster, more efficient, more stable, and more effective at their intended tasks. Some common areas where optimization is applied in drones include the following.

Flight Path Planning
Optimization techniques can be used to plan the most efficient flight path for a drone to complete a task, such as surveying a large area or delivering a package. This involves considering factors such as wind, weather, altitude, and battery life to minimize the time and resources required to complete the task.

Battery Life
Optimization techniques can be used to extend the battery life of a drone, which is a crucial factor in determining its flight time and range. This can involve minimizing the weight of the drone, using more efficient motors, or adjusting the flight path to reduce energy consumption.

Payload Capacity
Optimization techniques can be used to maximize the payload capacity of a drone, which is the amount of weight it can carry while still maintaining stable flight. This can involve adjusting the drone's design, such as using stronger materials or more powerful motors or optimizing the flight path to minimize the energy required to carry the payload.

Data Processing
Optimization techniques can be used to process and analyze data collected by drones, such as aerial images or sensor readings. This involves using algorithms to identify patterns and insights from the data, which can help to improve decision-making and optimize drone operations.

Overall, optimization plays a crucial role in the efficient and effective use of drones, allowing them to perform tasks more quickly, accurately, and cost-effectively.

4.2 Drone Flocking Optimization

Drone flocking optimization refers to the process of making a group of drones work together in a coordinated manner, like a flock of birds. This involves optimizing the algorithms and software that control the drones' flight paths, so that they can fly in formation and avoid collisions with each other. By optimizing drone flocking, we can make the group of drones work more efficiently, accomplish tasks more quickly, and achieve better results. This technology has various applications, such as search and rescue missions, precision agriculture, and aerial surveillance.

5 Commercial Uses of Drones

Almost every industry now uses drones, extending their reach and expanding their areas of expertise. Here is a brief overview of the application of drones.

Aerial Photography and Videography
Drones equipped with high-resolution cameras can capture stunning aerial footage for a wide range of applications, including films and real estate listings. They can also provide real-time monitoring for live events and sports.

Agriculture
Drones can be equipped with sensors and cameras to monitor crop health and identify pests and diseases. This can help farmers make data-driven decisions to improve crop yields and reduce costs.

Construction
Drones can provide real-time aerial surveys and inspections, track progress, and create 3D models of construction sites.

Infrastructure Inspection
Drones can inspect bridges, power lines, pipelines, and other infrastructure for maintenance and safety purposes. This can help reduce the need for human inspection and improve safety by identifying potential issues before they become problems.

Delivery
Drones can deliver packages, food, and medical supplies to remote or hard-to-reach areas, reducing delivery times and improving accessibility.

Search and Rescue
Drones equipped with thermal imaging cameras can locate missing persons or survivors in search and rescue operations, even in low light or difficult terrain.

Environmental Monitoring
Drones can monitor wildlife, track changes in land use, and measure pollution levels in the environment.

Surveying and Mapping
Drones can provide detailed topographic and geographic data for surveying and mapping purposes, allowing for more accurate and efficient land management and development.

Overall, drones have many potential commercial uses that can help improve efficiency, reduce costs, and provide valuable insights and data.

6 Flocking Rules of Drones

The flocking rules of drones are a set of algorithms and rules that are used to coordinate the flight of a group of drones in a flock-like formation. These rules are typically based on three key behaviors:

Separation Each drone maintains a safe distance from its neighbors to avoid collisions.

Alignment Each drone adjusts its velocity and heading to match its neighbors, so they move in the same direction.

Cohesion Each drone moves toward the center of the flock, so they stay together.

The flocking rules are implemented using sensors, such as GPS and cameras, to enable each drone to track its position and the positions of its neighbors. These rules allow the drones to fly in a coordinated manner and accomplish tasks more efficiently, such as surveying an area or inspecting a structure.

7 Literature Review

Paper 1: Drone Flocking Optimization Using NSGA-II and PCA
Bansal et al. [1] proposed an efficient drone flocking in a constrained area with several conflicting objectives. Avoiding collisions, moving quickly, correlating data, and communicating are the goals. The paper proposes a method for optimizing drone flocking behavior using a combination of two techniques: NSGA-II and principal component analysis (PCA).

NSGA-II is a type of evolutionary algorithm that can find optimal solutions to multi-objective problems by generating and evolving a set of candidate solutions. The authors use NSGA-II to optimize two objectives: minimizing the distance between drones and maximizing the angle between their velocities. This helps ensure that the drones stay close together while avoiding collisions.

PCA is a statistical technique that can reduce the complexity of high-dimensional data by identifying the most important variables that contribute to the data's variability. The authors use PCA to extract the most significant features from the drones' sensor data, such as position, velocity, and acceleration. This helps simplify the input data for NSGA-II and improve the optimization results.

The proposed method is tested on a simulated drone flocking scenario, where multiple drones need to move together while avoiding obstacles and maintaining formation. The results show that the NSGA-II and PCA approach outperforms other optimization methods in terms of convergence and diversity of solutions. The authors also demonstrate how the optimized flocking behavior can be applied to real-world applications, such as surveillance and monitoring.

The proposed method has potential applications in various domains that require coordinated and efficient drone operations.

Paper 2: Optimized Flocking of Autonomous Drones in Confined Environment
Vásárhelyi et al. [2] proposed a flocking model for actual drones that uses an evolutionary framework with carefully selected order parameters and fitness function. They numerically showed that the induced swarm behavior remained stable under realistic conditions for high flock numbers and particularly for big velocities.

Early microscopic agent-based models claim that initiating and maintaining collision-free cohesive flocking only need three straightforward interactions

between idealistic agents: repulsion in the short range, velocity alignment in the medium range, and attraction in the long range. Therefore, we used this algorithm to find good settings for our model. The populations in our evolutionary algorithm consisted of parameter vectors whose fitness was determined by a 10-min-long realistic simulation of the system. Each partial fitness, as well as the final fitness value, takes values between 0 (worst case) and 1 (ideal case).

Evolutionary optimization produced a huge number of stochastic fitness evaluations, which also contain precious information about the reasonable parameter ranges where fitness is expected to be high.

Overall, we achieved our first goal: The overall model could be successfully recreated in simulation with the right parameter values; the optimal setup showed a strong and efficient flock in the studied velocity range, which can act as the foundation for actual field investigations.

Paper 3: Vision-Based Drone Flocking in Outdoor Environment
Schilling et al. [5] present a study on how drones can be programmed to fly together in a coordinated manner using visual information.

The researchers focused on outdoor environments and developed a system that allows drones to flock, imitating the behavior of birds flying together. They utilized vision-based techniques, meaning the drones used their onboard cameras to gather information about their surroundings.

To achieve flocking behavior, the drones employed algorithms that allowed them to estimate their positions relative to each other and maintain a desired distance and formation. The drones communicated with each other through wireless connections, exchanging information to make collective decisions.

The experiments conducted by the researchers demonstrated successful drone flocking, showing that the drones could fly in formation while adapting to changes in the environment. This technology has potential applications in various areas such as surveillance, search and rescue missions, and even package delivery.

Paper 4: Optimization for Drone and Drone-Truck Combined Operations: A Review of the State-of-the Art and Future Directions
Chung et al. [3] provide a survey on the optimization method for drone operations (DO) and drone truck combined operations (DTCO) research that focus more on DTCO. As discuss earlier, drone can play an important role in a variety of application areas such as construction, agriculture, logistics, security management, and entertainment. A well-planned drone operation with the aid of state-of-art optimization techniques solves the above problems and maximizes such potential benefits. This paper also discusses the brief review of DO and DTCO research and associates such cases to the variant of traveling salesman problem (TSP) and vehicle routing problem (VRP) using their taxonomy.

We observe that combining drones with other vehicles, such as trucks, can considerably increase the effectiveness and efficiency of DO. The development of drone research, however, is still in its early stages and will take some time.

Paper 5: Optimization Approaches for Civil Applications of Unmanned Aerial Vehicles (UAVs) or Aerial Drones: A survey

Otto et al. [6] provide a literature survey on optimization approaches to civil applications of drones. This article is the first to provide a comprehensive general overview of optimization problems arising in operations planning of civil drone applications.

They discussed that drones are an emerging technology, and new concepts and inventions are constantly appearing, such as floating (flying) warehouses, automated battery switching, and in-flight payload switching between drones. They also give a short summary of the technological characteristics of drones. Understanding the economic and social benefits of drones in different applications and developing thorough business cases are crucial. More broadly, it's important to develop technology and regulations that will address public concerns about privacy and safety without impeding the use of drones to create value. According to several experts, the largest market potential may be related to drone applications in monitoring of infrastructure and construction sites, agriculture, and delivery applications.

In some civil applications, like disaster management, transport of medical supplies or environmental monitoring, drones may even help reduce human injuries and save lives.

Paper 6: Prospect and Recent Research & Development for Civil Use Autonomous Unmanned Aircraft as UAV and MAV

Nonami [7] explore the potential and advancements in the field of autonomous unmanned aircraft for civil applications, specifically focusing on unmanned aerial vehicles (UAVs) and micro air vehicles (MAVs).

This paper highlights the increasing importance of UAVs and MAVs in various civil domains such as surveillance, aerial photography, environmental monitoring, and disaster management. It emphasizes the advantages offered by these autonomous systems, including cost-effectiveness, increased safety, and the ability to access hard-to-reach areas.

It discusses the challenges and technical requirements associated with the development of autonomous UAVs and MAVs. These include navigation and control algorithms, sensing and perception technologies, communication systems, and power supply considerations

This paper provides an overview of recent research and development efforts in the field. It discusses advancements in autonomy, including sensor fusion techniques, obstacle avoidance algorithms, and path planning strategies. Furthermore, the integration of UAVs and MAVs with other systems, such as ground control stations and satellite networks, enhanced their capabilities and data transmission. The authors emphasize the potential of swarm intelligence, where multiple autonomous vehicles work collaboratively to achieve complex tasks.

Paper 7: The Flying Sidekick Traveling Salesman Problem – Optimization of Drone-Assisted Parcel Delivery

Murray and Chu [8] discuss the potential benefits of using drones for parcel delivery in combination with traditional delivery methods. The authors propose a new optimization model for the "flying sidekick traveling salesman problem," which involves determining the most efficient route for a delivery person to visit multiple locations while being assisted by a drone.

The authors argue that using drones for delivery can significantly reduce the time and cost associated with traditional methods, particularly in remote or hard-to-reach areas. However, they also note that the integration of drones into existing delivery systems poses several challenges, such as regulatory and technical issues, which need to be addressed for the implementation of such systems.

To address these challenges, they propose a new optimization model that takes into account the location of the delivery person, the weight and size of the packages, and the range and capacity of the drone. The model aims to minimize the delivery time and cost while also ensuring that the drone operates within its range and capacity limits.

Overall, the paper provides insights into the potential benefits and challenges of using drones for delivery and proposes a new optimization model for solving the "flying sidekick traveling salesman problem."

Paper 8: Impact of Drone Delivery on Sustainability and Cost – Realizing the UAV Potential Through Vehicle Routing Optimization

Chiang et al. [9] propose a mixed-integer multi-vehicle green routing model for UAVs to incorporate the sustainability aspects of the use of UAVs for last-mile parcel deliveries as well as cost savings. The paper explores the potential impact of using drones for delivery services on both sustainability and cost, with a focus on vehicle routing optimization. The authors argue that drone delivery can significantly reduce carbon emissions and costs, but its potential can only be fully realized through effective optimization of routing strategies.

To evaluate the impact of drone delivery, the authors conducted a case study on a hypothetical delivery network in Taiwan, comparing the use of traditional delivery vehicles with drone delivery. They analyzed various factors such as delivery distance, load capacity, and battery life to determine the optimal drone route for each delivery location.

The results of the study showed that drone delivery can significantly reduce carbon emissions and costs compared to traditional delivery vehicles. However, to achieve maximum efficiency, it is essential to optimize routing strategies, considering factors such as battery life, delivery distance, and load capacity.

Paper 9: Development of a Fuel Consumption Optimization Model for the Capacitated Vehicle Routing Problem

Xiao et al. [10] focus on developing a fuel consumption optimization model for the capacitated vehicle routing problem. The authors address the need for optimizing fuel consumption as it is a crucial factor in the transportation industry, not only for environmental reasons but also for economic purposes. The proposed model uses a

combination of genetic algorithms and a mathematical programming technique to develop a routing solution that minimizes fuel consumption while considering the vehicle capacity constraints. The authors provide a detailed description of the model, including the problem formulation and the optimization process. The results of the experiments conducted to validate the proposed model demonstrate its effectiveness in achieving significant fuel savings while maintaining feasible routes that meet the vehicle capacity constraints.

The paper highlights the importance of considering fuel consumption in the vehicle routing problem and provides a useful framework for optimizing fuel consumption in transportation planning.

Paper 10: Vehicle Routing Problem with Fuel Consumption and Carbon Emission

Zhang et al. [11] addresses the vehicle routing problem (VRP) with the consideration of fuel consumption and carbon emissions. The authors propose a mathematical model for the problem, which aims to minimize the total cost of the VRP while satisfying capacity and time window constraints. The model considers the fuel consumption and carbon emissions of the vehicles, which depend on various factors such as the load, speed, and terrain of the routes. They propose a solution method based on a two-stage approach, where the first stage determines the optimal routes and the second stage optimizes the vehicle assignments to the routes.

The proposed method is tested on benchmark instances from the literature and compared to other VRP models that do not consider fuel consumption and carbon emissions. The results show that the proposed model can significantly reduce the fuel consumption and carbon emissions compared to the other models while maintaining a similar level of cost and service quality.

The paper concludes that incorporating fuel consumption and carbon emissions into VRP models can lead to more sustainable and efficient transportation operations.

Paper 11: Strategic Design for Delivery with Drones and Trucks

Campbell et al. [12] discuss the potential benefits of using a combination of drones and trucks for delivery services. The paper focuses on the strategic design of the delivery network, taking into consideration factors such as customer demand, delivery locations, and delivery times.

The authors argue that the use of drones for last-mile delivery can significantly reduce delivery times and costs. Drones can be used to deliver packages to customers in hard-to-reach areas, such as rural regions or congested urban areas. However, drones have limited payload capacities, and their range is restricted by battery life. Therefore, the authors propose the use of trucks as a base for the drones, where the trucks can be used to transport the drones and recharge their batteries.

The paper also discusses the potential challenges of integrating drones into the delivery network, such as regulatory issues, safety concerns, and operational complexities. The authors suggest that a hybrid delivery system, consisting of both drones and trucks, can address these challenges and provide a more efficient and effective delivery service.

Paper 12: A Multi-Objective Approach for Unmanned Aerial Vehicle Routing Problem with Soft Time Windows Constraints

Guerriero et al. [13] proposed a new approach for solving the unmanned aerial vehicle (UAV) routing problem with soft time window constraints.

The objective of this problem is to find optimal routes for multiple UAVs to deliver packages to different locations within specific time windows while minimizing the total distance traveled and missed deliveries.

Their approach was multi-objective, considering several objectives at once, including the two mentioned above and the flexibility in the delivery times to account for uncertainty in the time windows. The proposed method was tested on various instances of the problem, and it was found to outperform existing approaches in terms of solution quality and time required to find them.

The authors tested their approach on a set of randomly generated instances of the problem, and the results showed that their approach outperformed existing approaches in terms of both the quality of the solutions and the time required to find them. This paper presents an innovative approach to the UAV routing problem with soft time window constraints, which could have practical applications in logistics and transportation industries.

Paper 13: Multi-Objective Optimization Model for a Green Vehicle Routing Problem

Jabir et al. [14] present a model for optimizing the routing of green vehicles. The authors aim to minimize the total travel distance and fuel consumption of the vehicles while also minimizing their carbon footprint. The proposed model is a multi-objective optimization model that considers the distance, fuel consumption, and carbon emissions of the vehicles.

The authors used a genetic algorithm to solve the optimization problem and compared the results with those obtained from a traditional single-objective optimization approach. The results showed that the multi-objective optimization model was able to generate better solutions than the single-objective optimization model in terms of both distance and emissions.

This paper provides insights into how green vehicle routing problems can be optimized using multi-objective optimization models. It highlights the importance of considering multiple objectives while optimizing the routing of green vehicles and shows that a multi-objective approach can lead to better outcomes compared to a single-objective approach.

Paper 14: The Design Challenges of Drone Swarm Control

Saffre et al. [15] discuss the challenges of controlling drone swarms. The authors argue that as drone technology advances and their usage become more widespread, controlling a large number of drones will be critical to ensure their safe and efficient operation.

The paper identifies several crucial design challenges that must be addressed to control drone swarms effectively. These include developing communication, coordination, and control systems, addressing issues related to power and propulsion, navigation, and sensing.

The authors also emphasize the importance of considering ethical and legal issues when designing drone swarm control systems. They argue that these systems must be designed to protect privacy and security, prevent harm to people or property, and adhere to relevant laws and regulations.

Overall, the paper sheds light on the complex design challenges associated with controlling drone swarms, highlighting the need for interrelated coordination to develop effective solutions.

Paper 15: Flocking Algorithm for Autonomous Flying Robots

Virágh et al. [16] present an algorithm for controlling the motion of autonomous flying robots in a flocking behavior inspired by the behavior of birds.

The algorithm proposed in this paper is based on three main rules: separation, alignment, and cohesion. The separation rule ensures that the robots maintain a minimum distance from each other to avoid collisions. The alignment rule ensures that the robots move in the same direction as their neighbors. The cohesion rule ensures that the robots move toward the center of the flock.

To implement these rules, the authors propose a decentralized control architecture where each robot communicates with its nearest neighbors to exchange information about their positions and velocities. Based on this information, each robot calculates its own acceleration using the three rules mentioned above.

The authors demonstrate the effectiveness of their algorithm through simulation experiments where a group of autonomous flying robots successfully perform flocking behavior. They also show that their algorithm is robust to changes in the number of robots, initial positions, and velocities. The paper presents a well-designed algorithm that can be used to control the motion of autonomous flying robots in a flocking behavior, which could be useful in various applications such as search and rescue missions and environmental monitoring.

Paper 16: Optimal Path Planning for Drones Based on Swarm Intelligence Algorithm

Saeed et al. [17] present a new approach for drone path planning using a swarm intelligence algorithm. The authors highlight the importance of finding efficient and optimal paths for drones in various applications.

The authors begin by highlighting the significance of path planning in drone applications, emphasizing the need for efficient and optimal routes to minimize energy consumption and ensure timely completion of tasks.

The proposed algorithm, based on swarm intelligence, consists of three main phases: initialization, exploration, and exploitation. In the initialization phase, the drones are deployed in the search space, and the algorithm initializes the parameters and variables required for path planning. The exploration phase involves the drones searching the environment and sharing information about their positions and the quality of their paths. This information exchange enables the drones to collectively explore the search space and avoid redundant search efforts. The exploitation phase focuses on exploiting the gathered information to converge toward the optimal path.

To evaluate the performance of the proposed algorithm, the authors conduct several experiments and compare the results with other state-of-the-art algorithms

used for drone path planning. The experiments involve scenarios with different complexities and obstacles, and performance metrics such as path length, energy consumption, and convergence speed are considered. It leads to shorter paths, reduced energy consumption, and faster convergence.

The algorithm demonstrates superior performance, contributing to improved efficiency and effectiveness in drone applications.

Paper 17: Binary Drone Squadron Optimization Approaches for Feature Selection

Singh et al. (2022) propose an efficient approach to select the most relevant features from a given dataset, with the goal of optimizing the performance and capabilities of a drone squadron.

The authors propose three optimization algorithms: Binary Bat Algorithm (BBA), Binary Whale Optimization Algorithm (BWOA), and Binary Particle Swarm Optimization (BPSO). These algorithms help choose the most relevant features from a given dataset, improving the overall performance of the drone squadron.

The BBA algorithm mimics the behavior of bats to explore the solution space. The BWOA algorithm draws inspiration from whale social behavior to identify relevant features. The BPSO algorithm simulates the movement of particles in a swarm to search for the best feature combination.

The experiments show that the proposed algorithms effectively reduce the dimensionality of the feature space while maintaining or even improving the classification accuracy. The authors also analyze the convergence speed and stability of their algorithms, highlighting their effectiveness in finding near-optimal feature subsets.

Paper 18: The Fast Flight Trajectory Verification Algorithm for Drone Dance System

Kung et al. [18] introduce a novel algorithm specifically designed to verify the flight trajectories of drones in a Drone Dance System. Drone Dance Systems involve the coordinated and synchronized flight of multiple drones to create visually captivating performances. In such systems, it is crucial to ensure that the drones follow their planned flight paths accurately, maintaining synchronization and achieving the desired visual effects.

To address this requirement, the authors propose a fast flight trajectory verification algorithm that efficiently compares the planned flight trajectories of the drones with the actual flight data. The algorithm aims to quickly detect any discrepancies or deviations between the planned and actual trajectories, enabling real-time adjustments to be made during the performance.

The algorithm's primary focus is on speed and efficiency, as it needs to operate in real time to verify the flight trajectories effectively.

The authors conducted experiments to evaluate the performance of their proposed algorithm. The results demonstrated the algorithm's effectiveness in accurately verifying the flight trajectories of the drones. It successfully detected deviations and inconsistencies, enabling timely adjustments to be made during the Drone Dance

System performances. The algorithm's speed and efficiency make it a valuable tool for ensuring the synchronization and precision of drone flights in various performance settings.

Overall, the paper presents a fast flight trajectory verification algorithm that plays a crucial role in ensuring the accuracy and synchronization of drone flight paths in a Drone Dance System.

Paper 19: Optimal Route Planning for Truck–Drone Delivery Using Variable Neighborhood Tabu Search Algorithm

Tong et al. [19] introduce a Variable Neighborhood Tabu Search (VNTS) algorithm to optimize the route planning process.

The aim of the study is to address the challenges of last-mile delivery, where the integration of drones with traditional delivery vehicles (trucks) can significantly enhance efficiency and reduce costs. The VNTS algorithm is designed to find the most optimal routes by considering various factors such as delivery time, energy consumption, and the capacity of both trucks and drones.

The algorithm utilizes a Tabu search approach combined with different neighborhood structures to explore different solutions. By applying the VNTS algorithm, the researchers were able to find better routes that minimized the total delivery time and energy consumption while satisfying the constraints of the delivery system.

Overall, the paper approaches to optimize route planning for truck-drone delivery systems, offering potential benefits in terms of improved efficiency and reduced costs.

Paper 20: Outdoor Flocking of Drones with Decentralized Model Predictive Control

Yuan et al. [4] investigate the use of decentralized model predictive control (MPC) for achieving coordinated flocking behavior in outdoor quadcopter drones. The objective of their study is to design a control strategy that allows multiple drones to navigate and move together as a flock without relying on a centralized control system.

The authors propose a decentralized MPC approach where each drone makes autonomous decisions based on its local information and the predicted behavior of its neighboring drones. This approach eliminates the need for explicit communication or coordination between the drones, making it suitable for outdoor environments where communication constraints may exist.

The decentralized MPC framework consists of two key components: a local model predictive controller and a consensus-based algorithm. The local controller generates optimal control inputs for each individual drone based on its own dynamics and local objectives. The consensus algorithm ensures that all drones converge to a common velocity vector while maintaining a desired inter-drone spacing.

The authors conducted experiments to validate the effectiveness of their approach. The results demonstrated that the decentralized MPC strategy allowed the quadcopter drones to achieve coordinated flocking behavior in outdoor scenarios.

The drones successfully maintained a desired formation while avoiding collisions and adapting to environmental changes.

In conclusion, the paper presents a decentralized model predictive control approach for outdoor flocking of quadcopter drones.

Paper 21: Combining Stigmergic and Flocking Behaviours to Coordinate Swarms of Drones Performing Target Search

Cimino et al. [20] explores the integration of two techniques, stigmergy and flocking, to effectively coordinate a group of drones in searching for targets. Stigmergy refers to a type of indirect communication where individuals interact with their environment by leaving traces or markers that influence the behavior of others.

Flocking behavior, on the other hand, is a collective motion observed in bird flocks or fish schools, where individuals align their movement with neighbors and maintain cohesion. In this study, the authors propose combining these two behaviors to enhance the coordination and efficiency of swarms of drones during target search operations.

By using stigmergic communication, drones can leave markers or information in the environment, such as the presence of a target or obstacles. Other drones can then detect and interpret these markers to make informed decisions.

The flocking behavior is incorporated to ensure that drones maintain a certain degree of cohesion and alignment in their movement. This helps in avoiding collisions and enables efficient exploration of the search area.

The authors conducted experiments to validate their approach, and the results showed improved coordination and search performance compared to individual drone search strategies. The combination of stigmergy and flocking allowed the drones to effectively communicate and adapt their search patterns based on the information shared among them.

Paper 22: Towards Drone Flocking Using Relative Distance Measurements

Brandstätter et al. [21] explore the concept of drone flocking by utilizing relative distance measurements. Flocking refers to the behavior of a group of drones that move in a coordinated manner, similar to how birds fly together.

The development of a flocking algorithm allows drones to maintain a specific formation based on relative distance measurements between neighboring drones. By considering the distances and positions of nearby drones, the algorithm enables drones to adjust their flight trajectories to maintain a cohesive flock.

The proposed algorithm utilizes a combination of artificial potential fields and a coordination mechanism based on relative distance measurements. Artificial potential fields guide the drones toward desired locations while avoiding collisions. The coordination mechanism ensures that drones maintain a certain distance from their neighbors, leading to a cohesive formation.

To evaluate the effectiveness of their approach, the authors conduct experiments using a set of simulated drones. The experiments demonstrate that the proposed algorithm successfully enables drones to flock and maintain formation while

avoiding collisions. The algorithm also exhibits robustness in the face of dynamic scenarios, such as drones entering or leaving the flock.

The authors provide valuable insights into the development of drone flocking systems using relative distance measurements. Their proposed algorithm shows promise in achieving coordinated flight and maintaining formation among a group of drones.

Paper 23: Multi-Agent Spatial Predictive Control with Application to Drone Flocking

Brandstätter et al. [22] developed the concept of multi-agent spatial predictive control and its application to drone flocking.

The authors present an extended version of their research, which focuses on developing a control system for coordinating the movements of multiple drones in a flock. The key objective is to enable the drones to navigate and maintain a desired formation while avoiding collisions and dynamically responding to changes in the environment.

The proposed control system incorporates spatial predictive control, which involves predicting the future positions and trajectories of the drones based on their current states and dynamics. By considering these predictions, the control system can make proactive adjustments to the drones' flight paths, ensuring smoother and more coordinated movements within the flock.

To validate their approach, the authors conduct simulations and experiments using a set of drones. The results demonstrate that the multi-agent spatial predictive control system effectively enables the drones to flock, maintain formation, and adapt to changing circumstances. The system also showcases robustness in the face of disturbances or perturbations in the environment.

This research contributes to the advancement of drone flocking by introducing a multi-agent spatial predictive control system. By leveraging predictive modeling and proactive adjustments, the system enhances the coordination and responsiveness of drones in a flock.

Paper 24: Distributed Three-Dimensional Flocking of Autonomous Drones

Albani et al. [23] explore the concept of flocking behavior in autonomous drones and propose a distributed approach to achieve three-dimensional flocking.

This paper focuses on developing a system that enables autonomous drones to flock together and move in a coordinated manner. Flocking refers to the collective behavior of a group of drones, similar to how birds fly in a flock. The proposed approach utilizes a distributed control system, where each drone operates autonomously and communicates with neighboring drones to maintain flocking behavior. By exchanging information about their positions and velocities, the drones coordinate their movements to achieve a cohesive flock. They demonstrate their approach through simulations and experiments. The results show that the distributed control system effectively enables the drones to achieve three-dimensional flocking while avoiding collisions and maintaining a desired formation. The system also exhibits robustness in the presence of perturbations or changes in the environment.

This approach allows drones to exhibit flocking behavior in three-dimensional space, enhancing their coordination and collective movements. The findings have potential applications in various domains, including swarm robotics, aerial surveillance, and collaborative tasks requiring multiple drones to work together.

Paper 25: An Optimized Flocking Starling Algorithm for Autonomous Drones
Chen et al. [24] present an innovative approach for achieving flocking behavior in autonomous drones using an optimized algorithm inspired by the behavior of starlings.

This paper focuses on developing a system that enables drones to mimic the collective behavior observed in flocks of starlings. Flocking behavior involves drones moving in a coordinated manner, maintaining formation, and responding to environmental changes as a cohesive unit.

The proposed approach introduces an optimized version of the Starling Algorithm specifically tailored for autonomous drones. The algorithm utilizes principles derived from the behavior of starlings, such as alignment, cohesion, and separation, to guide the drones' movements and achieve flocking behavior.

To evaluate the performance of the optimized algorithm, the authors conduct experiments using autonomous drones. The results demonstrate that the algorithm effectively enables the drones to exhibit flocking behavior while maintaining a desired formation, avoiding collisions, and adapting to dynamic scenarios.

They provide valuable insights into achieving flocking behavior in autonomous drones through an optimized algorithm inspired by the natural behavior of starlings. This approach contributes to the advancement of swarm intelligence and has practical applications in various fields, including search and rescue operations, surveillance, and coordinated tasks requiring multiple drones to work together.

Paper 26: Autonomous Cooperative Flocking for Heterogeneous Unmanned Aerial Vehicle Group
Wu et al. [25] present a study on achieving autonomous cooperative flocking behavior in a group of heterogeneous unmanned aerial vehicles (UAVs).

The authors propose a novel framework that combines a virtual leader-based approach with a potential field method to enable effective coordination and navigation within the UAV group. The system utilizes a distributed control scheme that allows each UAV to adjust its flight parameters based on the positions and velocities of nearby neighbors.

Through extensive simulations and experiments, the proposed framework demonstrates improved flocking performance and robustness in scenarios involving heterogeneous UAVs. The findings of this research contribute to the development of cooperative control strategies for UAV swarms and have potential applications in various domains, including surveillance, search and rescue, and environmental monitoring.

Paper 27: Learning Vision-Based Cohesive Flight in Drone Swarms
Schilling et al. [30] focus on the concept of cohesive flight in drone swarms and propose a learning-based approach to achieve it using vision-based techniques.

They address the challenge of coordinating multiple drones in a swarm to fly in a cohesive manner, where they maintain a desired formation and avoid collisions with each other. They argue that vision-based methods can provide rich sensory information for swarm coordination, allowing the drones to perceive their environment and make informed decisions.

To tackle this problem, the authors propose a learning-based approach that combines deep reinforcement learning and computer vision techniques. They train a neural network to predict the optimal control actions for each drone based on visual inputs. The network takes raw image frames captured by the drones' onboard cameras as input.

The experimental setup used for training and evaluation involves a simulated environment with multiple drones and various visual cues. They employ a deep Q-network (DQN) algorithm to train the neural network, using a reward function that encourages cohesive flight behavior and penalizes collisions.

The results of their experiments demonstrate the effectiveness of the proposed approach. The trained drones successfully learn to fly in a cohesive manner, maintaining a desired formation while avoiding collisions. The authors compare their approach to other baseline methods and show that it outperforms them in terms of both formation maintenance and collision avoidance.

Overall, the paper presents a learning-based approach to achieve cohesive flight in drone swarms using vision-based techniques. The authors demonstrate the effectiveness of their method through experiments and comparisons with baseline methods. The work contributes to the field of swarm robotics and provides insights into the use of deep reinforcement learning and computer vision for coordinated drone flight.

Paper 28: Robust Aerial Robot Swarms Without Collision Avoidance

Mulgaonkar et al. [26] explore the concept of aerial robot swarms and propose an approach that does not rely on traditional collision avoidance techniques.

The authors argue that existing collision avoidance methods often result in conservative behavior, limiting the efficiency and agility of aerial robot swarms. Instead, they propose a robust swarm control framework that focuses on system-level performance and resilience.

In their approach, they leverage the decentralized nature of swarm systems by allowing individual robots to rely on local sensing and communication. By maintaining a specific distance threshold between neighboring robots, they achieve a form of implicit collision avoidance. This strategy enables the swarm to maintain a cohesive structure while avoiding direct collisions.

To demonstrate the effectiveness of their approach, the conducted experiments using a swarm of quadrotor robots evaluated the swarm's performance in various scenarios, including obstacle avoidance and dynamic environmental conditions. The results showed that the proposed framework successfully enabled the swarm to navigate and accomplish complex tasks without explicit collision avoidance.

In conclusion, the paper presents a robust swarm control framework for aerial robot swarms that does not rely on traditional collision avoidance methods.

Paper 29: Self-Organized UAV Traffic in Realistic Environments
Vásárhelyi et al. [2] explore the concept of self-organized traffic for unmanned aerial vehicles (UAVs) in realistic environments.

The authors propose an approach to coordinate UAVs without relying on centralized control or predefined routes. They argue that traditional methods based on centralized control or predefined routes can be impractical and limiting in complex environments. Instead, they propose a self-organized traffic system where UAVs can adapt their flight paths based on local interactions. The authors utilize a decentralized control algorithm inspired by swarm intelligence. Each UAV in the system is considered an autonomous agent that perceives its environment and makes decisions based on local information. By considering the positions and velocities of neighboring UAVs, each agent adjusts its flight path to avoid collisions while maintaining a smooth and efficient flow of traffic.

To evaluate their approach, the authors conducted simulations in realistic environments. They compared the performance of their self-organized traffic system with a centralized control approach. The results demonstrated that the self-organized system was capable of efficiently managing UAV traffic, adapting to dynamic situations, and effectively avoiding collisions.

Overall, the paper presents a self-organized traffic system for UAVs in realistic environments by leveraging decentralized control and local interactions.

Paper 30: Survey on Unmanned Aerial Vehicle Networks for Civil Applications
Hayat et al. [27] provide a comprehensive survey of unmanned aerial vehicle (UAV) networks from a communications perspective, focusing on their civil applications.

The increasing use of UAVs in various civil domains such as disaster management, surveillance, infrastructure inspection, and communication relay highlights the importance of reliable and efficient communication systems to enable the successful deployment and operation of UAV networks in these applications.

The survey covers several key aspects related to UAV networks, including communication architectures, channel models, spectrum management, multiple access techniques, and networking protocols. The authors discuss the unique challenges and requirements posed by UAV communications such as mobility, limited energy resources, dynamic network topologies, and quality-of-service considerations. Furthermore, the paper examines various communication technologies and their suitability for UAV networks, including cellular networks, ad hoc networks, wireless sensor networks, and satellite communication systems. The authors analyze the advantages, limitations, and potential integration strategies of these technologies in UAV networks. Additionally, the survey discusses the state-of-the-art research and development efforts in UAV communications, including ongoing standardization activities and experimental testbeds. The authors identify open research challenges and provide insights into future directions for the design and optimization of UAV communication systems.

In conclusion, it provides a valuable resource for researchers, practitioners, and policymakers interested in understanding the communication requirements, challenges, and technologies associated with UAV networks.

Paper 31: A System for the Design and Development of Vision-Based Multi-Robot Quadrotor swarms

Sanchez-Lopez et al. [28] focus on the development of a system that facilitates the design and implementation of vision-based multi-robot quad-rotor swarms.

They recognize the growing interest in multi-robot systems, particularly quad-rotor swarms, for various applications such as surveillance, exploration, and monitoring. However, designing and developing these swarms can be challenging due to the complex coordination and control requirements.

To address these challenges, the authors propose a system that integrates several components for the design and development of vision-based quad-rotor swarms. The system consists of hardware, software, and communication modules that work together to enable coordinated flight and interaction among the quad-rotors. The hardware component includes quad-rotor platforms equipped with cameras for vision-based sensing. The software component comprises modules for perception, control, planning, and communication. These modules enable the quad-rotors to sense their environment, make decisions, and communicate with each other to achieve coordinated behavior. The authors also describe a centralized ground station that serves as the control interface for the quad-rotor swarms. The ground station provides a user-friendly graphical interface for mission planning, real-time monitoring, and data visualization. To validate the effectiveness of their system, the authors conducted experiments with a quad-rotor swarm. The experiments involved tasks such as formation flying, target tracking, and obstacle avoidance. The results demonstrated the system's capability to enable coordinated flight and efficient execution of complex missions.

In conclusion, the integrated hardware, software, and communication modules facilitate coordinated flight and interaction among the quad-rotors.

Paper 32: Flocking of Multi-Agents with Time Delay

Yang et al. [29] focus on the study of flocking behavior in multi-agent systems that experience time delays.

Investigate the flocking phenomenon, which refers to the coordinated motion of a group of agents, such as birds or robots, to achieve a common goal. They specifically address the impact of time delays in the communication between agents and how it affects the flocking behavior.

The paper proposes a control framework for achieving flocking behavior in the presence of time delays. The control algorithm takes into account the local information and the delayed information from neighboring agents to guide the agents' motion. They analyze the stability of the flocking system under different delay conditions and provide mathematical proofs to support their findings. Furthermore, the paper discusses the application of the proposed control framework to practical scenarios. It explores the use of the flocking behavior in tasks such as formation control, cooperative target tracking, and obstacle avoidance. The authors provide simulations and experimental results to demonstrate the effectiveness of the control algorithm in achieving desired flocking behaviors in real-world scenarios.

In conclusion, the proposed algorithm incorporates local and delayed information to guide the agents' motion and ensures the stability of the flocking system.

8 Conclusion

After studying the papers related to drone optimization, we have to conclude that drone optimization is one of latest research paths for various researchers and academician. There are numerous tasks which can be executed using drone technology. Despite all the uses, it should be noted that some limitations of drones needs to be addressed. These limitations are formulated as multi-objective optimization problems.

There are many nature-inspired optimization techniques, which in future can be applied to solve these optimization problems.

Bibliogrphy

1. Bansal, J. C., et al. (2022). *Drone flocking optimization using NSGA-II and principal component analysis*. arXiv preprint arXiv:2205.00432.
2. Vásárhelyi, G., et al. (2018). Optimized flocking of autonomous drones in confined environments. *Science Robotics, 3*(20), eaat3536.
3. Chung, S. H., Sah, B., & Lee, J. (2020). Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research, 123*, 105004.
4. Yuan, Q., Zhan, J., & Li, X. (2022). Outdoor flocking of quadcopter drones with decentralized model predictive control. *ISA Transactions, 71*, 84–92.
5. Schilling, F., Schiano, F., & Floreano, D. (2021). Vision-based drone flocking in outdoor environments. *IEEE Robotics and Automation Letters, 6*(2), 2954–2961.
6. Otto, A., et al. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks, 72*(4), 411–458.
7. Nonami, K. (2007). Prospect and recent research & development for civil use autonomous unmanned aircraft as UAV and MAV. *Journal of system Design and Dynamics, 1*(2), 120–128.
8. Murray, C. C., & Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies, 54*, 86–109.
9. Chiang, W.-C., et al. (2019). Impact of drone delivery on sustainability and cost: Realizing the UAV potential through vehicle routing optimization. *Applied Energy, 242*, 1164–1175.
10. Xiao, Y., et al. (2012). Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Computers & Operations Research, 39*(7), 1419–1431.
11. Zhang, J., et al. (2015). Vehicle routing problem with fuel consumption and carbon emission. *International Journal of Production Economics, 170*, 234–242.
12. Campbell, J. F., et al. (2016). Strategic design for delivery with drones and trucks. In *Presented at the 2016 INFORMS conference*, Nashville, Tennessee, 14 November 2016.
13. Guerriero, F., Surace, R., Loscrí, V., & Natalizio, E. (2014). A multi-objective approach for unmanned aerial vehicle routing problem with soft time windows constraints. *Applied Mathematical Modelling, 38*(3), 839–852.

14. Jabir, E., Panicker, V. V., & Sridharan, R. (2015). Multi-objective optimization model for a green vehicle routing problem. *Procedia - Social and Behavioral Sciences, 189*, 33–39.
15. Saffre, F., Hildmann, H., & Karvonen, H. (2021). *The design challenges of drone swarm control* (pp. 408–426). Springer.
16. Virágh, C., et al. (2014). Flocking algorithm for autonomous flying robots. *Bioinspiration& Biomimetics, 9*(2), 025012.
17. Saeed, R. A., Omri, M., Abdel-Khalek, S., et al. (2022). Optimal path planning for drones based on swarm intelligence algorithm. *Neural Computing and Applications, 34*, 10133–10155.
18. Kung, C.-m., et al. (2020). The fast flight trajectory verification algorithm for drone dance system. In *2020 IEEE international conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*. IEEE.
19. Tong, B., et al. (2022). Optimal route planning for truck–drone delivery using Variable Neighborhood Tabu Search Algorithm. *Applied Sciences, 12*(1), 529.
20. Cimino, M. G. C. A., Lazzeri, A., & GigliolaVaglini. (2015). Combining stigmergic and flocking behaviors to coordinate swarms of drones performing target search. In *2015 6th international conference on information, intelligence, systems and applications (IISA)*. IEEE.
21. Brandstätter, A., et al. (2022a). *Towards drone flocking using relative distance measurements*. Leveraging applications of formal methods, verification and validation. Adaptation and learning: 11th international symposium, ISoLA 2022, Rhodes, Greece, October 22–30, 2022, proceedings, part III. Cham, Springer Nature Switzerland.
22. Brandstätter, A., et al. (2022b). *Multi-agent spatial predictive control with application to drone flocking (extended version)*. arXiv preprint arXiv:2203.16960.
23. Albani, D., et al. (2022). Distributed three dimensional flocking of autonomous drones. In *2022 international conference on robotics and automation (ICRA)*. IEEE.
24. Chen, F., et al. (2023). An optimized flocking starling algorithm for autonomous drones. In *Proceedings of 2022 international conference on autonomous unmanned systems (ICAUS 2022)*. Springer Nature Singapore.
26. Wu, J., et al. (2021). Autonomous cooperative flocking for heterogeneous unmanned aerial vehicle group. *IEEE Transactions on Vehicular Technology, 70*(12), 12477–12490.
26. Mulgaonkar, Y., et al. (2018). Robust aerial robot swarms without collision avoidance. *IEEE Robotics and Automation Letters, 3*(1), 596–603.
27. Hayat, S., Yanmaz, E., & Muzaffar, R. (2016). Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. *IEEE Communications Surveys & Tutorials, 18*, 2624–2661.
28. Sanchez-Lopez, J. L., Pestana, J., de la Puente, P., Suarez-Fernandez, R., & Campoy, P. (2014). A system for the design and development of vision-based multi-robot quadrotor swarms. In *2014 international conference on unmanned aircraft systems (ICUAS)* (pp. 640–648). IEEE.
29. Yang, Z., et al. (2012). Flocking of multi-agents with time delay. *International Journal of Systems Science, 43*(11), 2125–2134.
30. Schilling, F., et al. (2019). Learning vision-based flight in drone swarms by imitation. *IEEE Robotics and Automation Letters, 4*(4), 4523–4530.

Metaheuristics Algorithm for Search Result Clustering

Sushil Kumar, Sunny Parihar, and Vanita Garg

1 Introduction

Search Result Clustering (SRC) is the technique that requires the grouping of the results into a cluster and labels the clusters with some meaningful sentence which can be described as the results included in a cluster. Since, SRC is the most known problem of retrieving information and clustering into the groups based on some measure which gives the similarity among the web snippets in a cluster. In the field of computer science and mathematical optimization, a Metaheuristics is a higher level of heuristic designed to find, generate, or select a heuristic that is used to get the good solution of the optimization problems such as clustering, analysis etc. Metaheuristics Algorithms are also known as optimization methods designed according to the strategies laid out in a Metaheuristics framework. These heuristics are used in order to find a good solution in efficient computing time.

National Institute of Technology Kurukshetra.

S. Kumar
Department of Computer Engineering, National Institute of Technology Kurukshetra, Kurukshetra, India

Department of Computer Science and Engineering, National Institute of Technology Warangal, Warangal (TS), India
e-mail: skumar@nitkkr.ac.in

S. Parihar
Department of Computer Science and Engineering, National Institute of Technology Warangal, Warangal (TS), India

V. Garg (✉)
Department of Mathematics, Galgotias University, Greater Noida, India

V. Garg et al. (eds.), *Women in Soft Computing*, Women in Engineering and Science,
https://doi.org/10.1007/978-3-031-44706-8_7

1.1 Data Clustering

Data Clustering is a technique which is used to cluster the data points or population into a particular group or cluster according to some techniques in such a way that data points or populations that belong to a particular group have similar properties or nature. In other words, segregation of the similar items into one cluster is known as clustering. There are various clustering algorithms that are used to perform clustering of data. In Metaheuristics, there are certain algorithms that are used to perform clustering, and these algorithms mainly improve the clustering accuracy in an efficient way. Similarly, Particle Swarm Optimization is used for data clustering because PSO optimizes the problem by iterative in nature to improve the solution with respect to the measured quality.

2 Literature Review

SRC is a most popular problem in the fields of information retrieval method. A Multi-Objective Binary Differential Evolution (MBDE) framework is proposed by Van der Merwe and Engelbrecht [1]. Syntactic and semantic measures are taken into consideration for performing clustering. An experimental analysis is done on three datasets and computed the F-score value, and the results show that the MBDE performs better. The main limitation of multi objective optimization approach is that the number of queries increases the length of solution.

DE algorithm was proposed [2] based on some self-adaptation of adjustment mechanism. The proposed algorithm applies DE to adjust the mutation strategy during the evolution stage.

EWMA-DE [3] is basically a mutation adapting mechanism for DE. In this proposed algorithm Mutation Scale Factor was used for modification, while the other remains fixed. This algorithm performs better in the majority of the test cases.

EWMA-DECrF [4] uses two parameters that are Mutation Scale Factor and Crossover Factor. Both the parameters are used to adapt the modification based on some exponential weighting moving average. The third parameter is fixed, the same as classical DE, i.e., keeping population size constant. This algorithm is compared with classical DE, and the result shows that EWMA-DECrF performs better than classical DE.

The proposed method by Yang et al. [5] checks the population adaptation regarding population diversity and proposed a method named auto-enhanced population diversity (AEPD) which is used to increase the number of population diversity automatically. The result shows an improvement compared to the original algorithms and differential evolution has a superior performance over the other algorithms. This proposed approach [6] is compared with DErand and compared with other algorithms as well, to study the clustering ability of each algorithm. Experiments are done on the five different datasets and the result shows that the DE with a modified

mutation approach is efficient. A hierarchical DE algorithm [7] suggested for unsupervised algorithm. This approach can identify the number of clusters as well as their centers. Results show that this algorithm is feasible. A modified threshold ridge regression-based subspace clustering method was proposed [8] using K-means algorithm and differential evolution. This proposed method is compared with six different techniques, and the experimental data results show that this method performs better in terms of accuracy as well as normalized mutual information. A modification of Differential Evolution [9] for automatic clustering using large unlabeled datasets. This proposed method does not need any information regarding the data which is to be classified. This performs better than two algorithms and can be applied to real-world application like image segmentation.

A DE-based algorithm with cluster number oscillation for automatic crisp clustering ACDE-O was proposed [10]. This algorithm does not require information regarding the number of clusters to be formed in the dataset, since it finds the number of clusters with stable and fast convergence. This algorithm performs better compared to the partitional clustering algorithm. Automatic clustering using DE is found to be the most efficient considering all the features of DE. The work [11] proposes a novel (ACFSDE), which is an improved method of ACDE, which defines the structures for finding the optimal clusters and selecting optimal features. Cluster identification and feature selection [12] are done at the same time. Experimental results are compared with the various clustering approaches using the real-world benchmarks datasets from UCI. Improved DE [13] to determine similarity based clustering. The main objective of this method is to find an optimal number of clusters by decreasing the similarity matrix's uncertainty. An automatic clustering using DE (ACDE) [14] proposed to an optimal number of clusters is determined by encoding the activation value of each cluster. This proposed method activates the clusters into the chromosomes. Clustering Algorithm in Wireless Sensor Networks Based on Differential Evolution Algorithm was proposed [15], and clustering is done on the wireless sensor networks using DE. This method improves the performance by decreasing the amount of calculation and energy consumption.

Particle Swarm Optimization based on the initial population of clustering was proposed [16]. A method is proposed in which Particle Swarm Optimization is done by initializing the population for clustering purpose. Initial population and other parameters are set in such a way that it performs the optimized calculation. The simulation results show that this method effective and feasible. A new fitness function is used in the standard Particle Swarm Optimization methods. The work [17] was proposed with a new fitness function since this fitness function can further be improved by using an improved FCM function. An Improved version of Particle Swarm Optimization [18] is used for fast clustering research. Since the traditional clustering algorithms converge readily which falls into the local optimum in solution space and also efficiency is low as well hence a fast improved PSO method is used in this chapter.

A combined version of K-means and combinatorial Particle Swarm Optimization known as (KCPSO) was proposed in [19]. This approach does not require any knowledge on the clusters' counts, i.e., how many clusters should be formed.

Experimental results show that the KCPSO algorithm is effective as well as efficient for dynamic clustering problems. The proposed PSO [20] takes more time than the basic Particle Swarm Optimization and K-means, but it gives better results in the form of mean square error, entropy, and purity but takes more average CPU time. A simple version of Particle Swarm Optimization [1] is used. This chapter uses two methods to perform the clustering of the data since it is known that Particle Swarm Optimization can be used to get the centroids for each clusters formed from the dataset, but the number of clusters must be given to the user.

Research article [21] proposed a novel Particle Swarm Optimization approach. In this chapter randomly occurring distributed delayed PSO is used, and in each iteration of this method an evolutionary factor is calculated based on some various factors such as velocity. Shen et al. [22] proposed multi-swarm particle swarm optimization algorithm based on clustering dynamic grouping (DGMSPSO) which solves the problem of readily getting struck at local optima in optimization process and very low precision value as well. Hence, by the help of clustering grouping strategy local information of particle to each in search process can be enhanced which helps in improving the diversity of algorithm.

A combined version of hybrid firefly and Particle Swarm Optimization (PSO) in research article [23]. This firefly algorithm is the important regarding the solution of most difficult optimization problems in the field of global optimization. Its performance is based on the parameter tuning.

An hybrid firefly optimization algorithm is proposed in research article [24]. Since the algorithm is very powerful as well as effective swarm intelligence method when it comes to solving the optimization problems.

For high-dimensional data clustering, one approach is proposed [25] which is better than other algorithms when it comes to find the solutions as early as possible. The result is calculated using the modified version of three algorithms and compared with a simplified version, and the modified one performs significantly better.

A hybrid discrete artificial bee colony—GRASP algorithm [26] for clustering. The author used for clustering N objects into K clusters. This proposed method has two phases in which artificial bee colony is used for feature selection and GRASP is used for clustering the data. This proposed method gives the accuracy greater than 98%.

The hybrid approach [27] is the high level of hybridization where genetic algorithms represents the initialization phase (Global Search) while the Black Hole algorithm represents the searching phase (Local Search). This combination GA-BH is examined based on the various optimization problems, and hence it performs better than individual BH and GA algorithm. This proposed method gives the better global search space only because of GA as GA is capable of exploring search space.

An algorithm [28] is used for the multi-objective clustering problem. This proposed method is basically used for coupling and cohesion. It is the fast and effective algorithm for the search for a cluster which maximizes cohesion and minimizes the coupling of the software modules. The experimental results show that the proposed method is better than other methods in terms of the solution's quality and better computation time.

A modified version of Black Hole and Levy flight technique in research article [29]. The proposed method the black hole optimization method simulates the black hole phenomenon of the universe. At every iteration in the solution space can be seen as an individual star. Since simple BlackHole lacks in exploration in some datasets so to overcome this problem, this method includes levy flight into BlackHole named as Levy Flight Black Hole (LBH). This approach is tested on six datasets from UCI machine learning. This algorithm shows effectiveness and robustness and this is suitable for data clustering.

3 Proposed Methodology

Particle Swarm Optimization is the Metaheuristics mature inspired swarm-based algorithm which optimizes the problem. In each iteration of this algorithm, it tries to improves the solution search space in terms of measured quality. This algorithm solves the problem with the help of population of the candidate solutions and such particles move around the search space with respect to its velocity and position. Every particle present in the swarm is influenced by its local best position which helps in guiding to best known location in the solution search space, which is being updated as a better solution/position found by any other particle in the solution space. Since this algorithm requires a population of particles also known as swarm of particles, and these particles in the algorithm are a potential candidate to the solution of an optimization problem. The main motive of this PSO algorithm is to find those positions of particles, which gives best result with respect to the given fitness function. Here, swarm represents a number of solutions which has the potential to become a solution in an optimization problem. Each potential solution here represents a particle (Fig. 1).

Let us take a number of particles as N, and they are all move in a Q-Dimensional solution space. Each particle is considered as a Q-Dimension $q = (q_1, q_2, \ldots, q_d)$ in location of $S = (s_1, s_2, \ldots, s_q)$ and each particle moves with the velocity $Vel =$

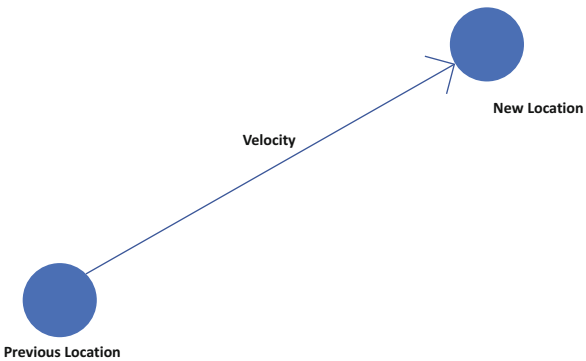


Fig. 1 Location of particle changed in solution space due to velocity

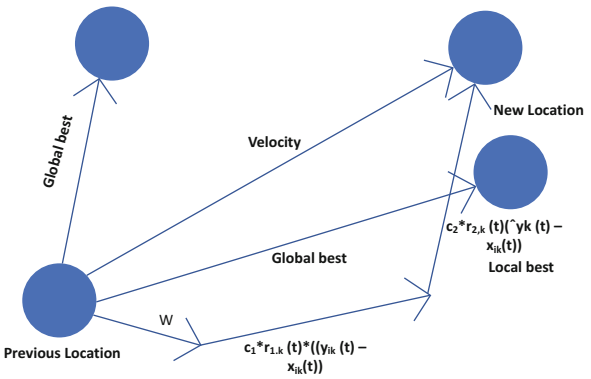


Fig. 2 Detailed movement of particle in solution space

$(vel_1, vel_2, \ldots, vel_q)$ in the solution space. The velocity is bounded by $Vel_{max} = (Vel_{max1}, Vel_{max2} \ldots Vel_{maxd})$, and if any particle is found with velocity greater than Vel_{max}, then its velocity is being replaced with Vel_{max}. Each particle p has the following properties (Fig. 2):

- x_i = Position of a particle p currently.
- vel_i = Velocity of the particle p currently.
- $y_i = Per_{best}$ position of the particle p.
- $\hat{y}$ = Global best position found so far.

Using these notations, any particle position can be found using Eqs. 2 and 3, where u is the inertial weight and c_1 and c_2 are acceleration constants. The velocity is calculated according to three factors which are:

1. Previous Velocity
2. Distance between the particle p's personal best position and its current position is known as cognitive component
3. Distance between the particle p's current position to the best position found thus far known as social components.

Equation (1) represents the variable inertia weight where two maximum w_2 and minimum w_1 are used to calculate the variable inertia weight at each iteration i where itr is the total number of iterations used in algorithm.

$$w = w_2 - w_1 * \frac{(itr - i + 1)}{itr} + w_1 \tag{1}$$

$$F_1 = J_e = \frac{\sum_{j=1}^{N_e} \left[\sum_{\forall \mathbf{z}_p \in C_{ij}}, d(\mathbf{z}_p.\mathbf{m}_j)/|C_{ij}| \right]}{N_c} \tag{2}$$

$$vel_{i,k}(T+1) = wvel_{i.k}(T) + c_1 r_{1.k}(T)(y_{ik}(T) - x_{ik}(T))$$

$$+ c_2 r_{2,k}(T)(\hat{y}_k(T) - x_{ik}(T) \tag{3}$$

$$\mathrm{x}_i(T+1) = \mathrm{x}_i(T) + \mathrm{vel}_i(T+1) \tag{4}$$

where Z_p shows the p_{th} data vector, C_i is the number of data vectors belonging to cluster i, and d is Euclidean distance between Z_p and m_j. Equation (2) fitness function shows good and approving results, but the results were not good enough, and also it was time consuming, since a standard fitness function has the following problems: Sometimes a particle that is not a good solution is mistakenly found as a good cluster. In order to solve this, fitness function is updated to solve the problem.

$$\begin{aligned} F_2 &= F_1 * (|C_{ik}| - |C_{il}| + 1) \\ |C_{ik}| &= \max_{\forall j=1\ldots,N_c} \{|C_{il}|\}, \{|C_{il}|\} = \min_{\forall j=1,N_c} \{|C_{ij}|\} \end{aligned} \tag{5}$$

4 Methodology

4.1 Initialization

In the first step, Initialization of each of the particle's velocity as well as the particle's position and then calculate the fitness value for of the each particle which give p_{id} and p_{gd}. Then the updated value of each particle's velocity as well as position with respect to fitness function value.

4.2 Calculation

Map each particle location into the clustering position and then calculation of the fitness value for each of the particle p. Use max as the maximum number of iteration in a loop to calculate the global best position and update it.

4.3 Updation

If the cost is less than pbest, then update the pid and pbest, else check if the personal best of a particle is less than the best position (globally in solution space) of overall all particles, then update this best position. Do this process until the maximum iteration is reached, else check again from step 3 via updating velocity and position of particle (Fig. 3).

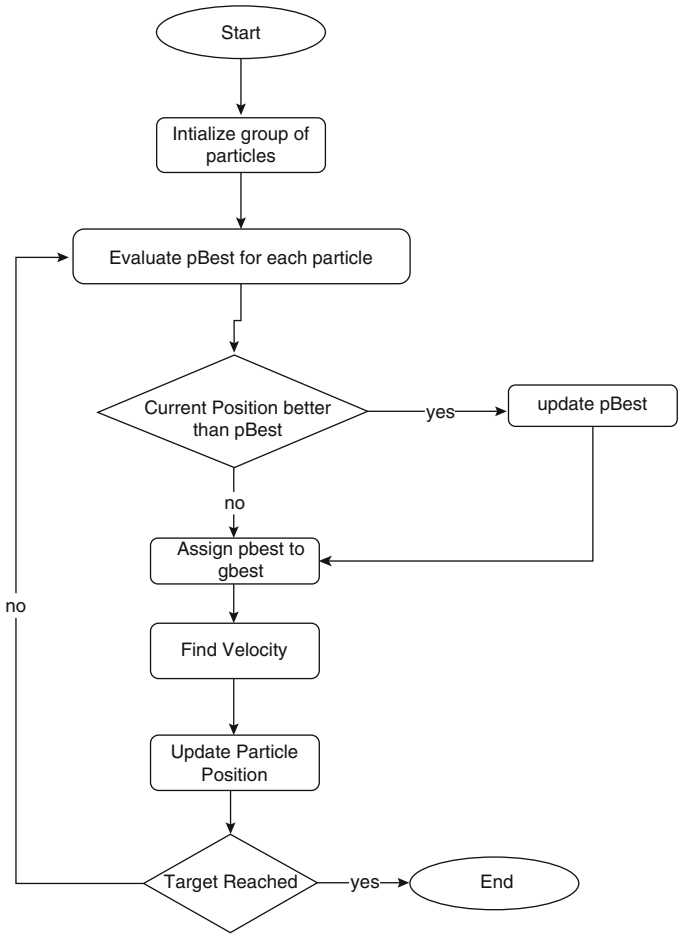


Fig. 3 Flow chart diagram

4.4 Flow Chart Diagram

1. Initialize a group of particles in the solution space with some random value which must be within the given required range.
2. Evaluate the personal best position for each particle via updating the location which can be done by change in the velocity of the particle.
3. Check if the new personal best is better than the global best, i.e., a position that is the best among all the particles in the solution space. If better, then assign pbest to the gbest.
4. Now calculate the new velocity of the particle which depends upon the three values which are velocity of that particle earlier, social, and cognitive component.
5. Use the velocity calculated above, and find the new position of the particle.
6. Go to step no. 2 until all iterations are finished.

5 Algorithms

5.1 PSO Algorithm

Algorithm 1 PSO

for Each Particle p **do**
  Initialization of each particle p_i with some random value within required range.
end for
for q = 1 to itr_{max} **do**

  for Each particle a **do**
    Compute the fitness function value.
  end for
  Select the Best Particle as $Global_{Best}$.

  for Each particle p **do**
    Updating the value of the Velocity for each particle p using equation no. (3).
    Updating the value of the Position for each particle p using equation no. (4).
  end for
end for

5.2 PSO Data Clustering

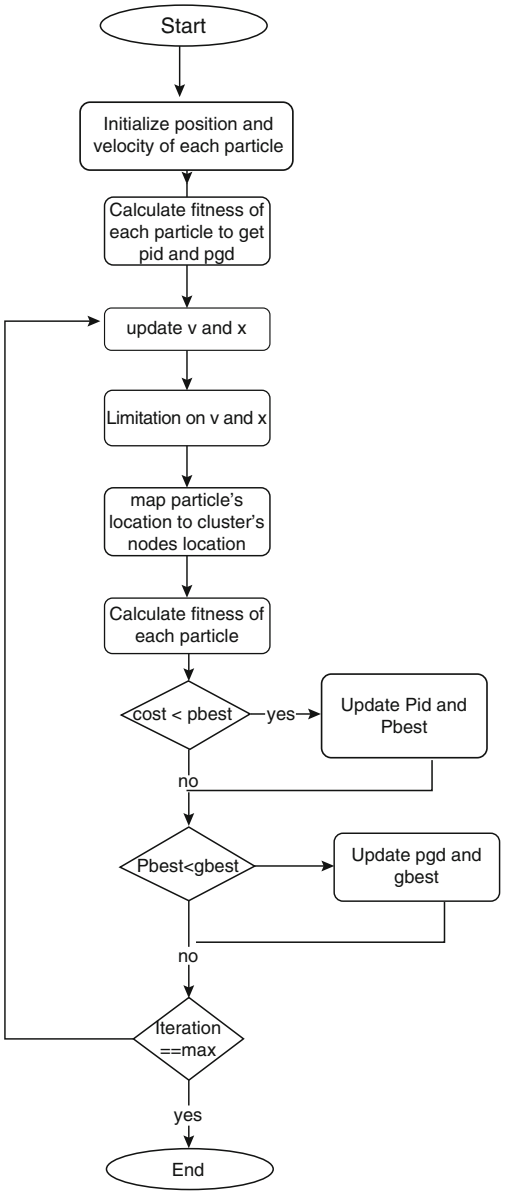
Algorithm 2 PSO clustering

for Each particle p in solution space **do**
  Initialize particle p to contain Wc randomly selected number of cluster centroids.
end for
for $itr \leftarrow$ 0 MaxIteration - 1 **do**
  for every particle p **do**
    for every data vector z_p **do**
      calculation of the value of Euclidean distance dist(z_p,m_{pb}) with respect to the all cluster centroids C_{pb}
      assign z_p to cluster c such that $dist(z_a, m_{ab}) = min(forallc = 1 toW, dist(z_a, m_{ac}))$
      calculate the fitness function value F_2 using Eq. (5)
    end for
  end for
end for
Updating the value of global and local best positions by best values found so far respectively
Updating the value of each cluster by using Eqs. (3) and (4).

5.3 Clustering Methodology

1. Let the number of particles present in the solution space be P (Fig. 4).
2. Initialize the position and the velocity of each particle in the solution space.
3. Calculate the value of fitness function F1 as well as F2 (both in different executions) to get the value of the pid and pgd.

Fig. 4 Methodology



4. Update the value of velocity and location in the solution space for each particle, and check if both values are within the given required range.
5. Now map the particle's location to the clusters' nodes location. And calculate the fitness value for each particle p.
6. Update the value of pBest as well as gBest when a better value is encountered.
7. Repeat this process starting from step no. (4) until the number of iterations is reached the maximum given value.

5.4 Text Data Clustering Using TF-IDF Vectorization

TF-IDF is Term Frequency-Inverse Document Frequency which is a numerical value of a word that shows how important the word is to corpus. Here, Frequency refers to the number of current words to the total number of words in a document.

$$tf(t, d) = \frac{N_t}{\sum_k N_k} \tag{6}$$

where N_t is the count of current words in the document/string, and the sum of N_k is the sum of count of all words in the document/string. Inverse Document Frequency is taking log of the ratio of the total number of documents present in the corpus to number of documents/strings in which a particular term or word is present t_i.

$$idf(t, D) = \log \frac{D}{\{d_i \in D \; t \in d_i\}} \tag{7}$$

td-idf(t,d,D) is the product value of the above-mentioned two equations as

$$tf - idf(t, d, D) = tf(t, d) \times idf(t, D) \tag{8}$$

Steps to perform text clustering of data using tf-idf vector method are as follows:

1. Cleaning of dataset (removing numbers, stopwords, and punctuation)
2. Making bag of words using dataset for corpus as well for words
3. Calculation of the term frequency for each word in the corpus
4. Calculation of the Inverse Document Frequency which signifies that how much a particular word used in the string
5. Taking product of the above two values for each word in corpus as td-idf weight vector. This vector gives the significance about which word is important compared to the other words
6. Transforming the vector into the required form for the clustering using the fit-transform method
7. Using this method for each of the columns present in the dataset and vectorization gives an output

8. Sparse Concatenation of each vector produced by the using every column
9. Using the clustering algorithm to perform the clustering on the concatenated vector formed

5.5 Text Data Clustering Using Word2Vec Model

Word2Vec algorithm is an NLP invented at Google. The Word2Vec model is used for mapping of a word to a vector of a number (generating embeddings). The basic idea used for embeddings of word is that those words that occur in a similar context in a text are more likely to be closer to each other in a vector space. Since the vector representations of word which are generating using word2vec model must put semantically similar words closer to each other in a 2D space. Using a clustering algorithm, one can group similar words from the text and can form clusters.

Steps to perform text clustering of data using Particle Swarm Optimization Algorithm are as follows:

1. Cleaning of dataset (removing numbers, stopwords, and punctuation)
2. Training of Word2Vec Model
3. Dimensionality reduction using PCA (Principal Component Analysis)
4. Applying PSO Algorithm for the cleaned dataset
5. Comparing the of results using fitness function in Eqs. (2) and (5)

6 Results

The PSO algorithm is trained for 5 datasets in which there are 3 numerical datasets and 2 text datasets.

6.1 Dataset Description (Table 1)

These 3 datasets contain only numerical data, and the number of clusters to be formed is given as well (Table 2).

Table 1 Dataset description for numerical data

Dataset name	Number of rows	Number of columns	Number of clusters
Iris dataset	150	4	3
Wine dataset	178	13	3
Glass dataset	214	9	7

Table 2 Dataset description

Dataset	Queries	Snippets
AMBIENT	44	4400
ODP-239	239	25580

Here the number of queries represents the total number of clusters that can be formed. And the snippets are the texts that belong to the particle number of clusters to be formed.

6.2 Output

For the clustering using PSO with two different fitness functions F1 and F2, data points are plotted on the graph and different colors are used for each cluster type to distinguish between them. In these images of data points, dark gray circular points represent the cluster's centroid. This graph is plotted at each 200th iteration in the implementation, and final 1000th iteration plotting of data points is given below. This is done by using both fitness functions F1 and F2. So, for each dataset the first image of data points shows the 1000th iteration's final result using F1 and the second image for each dataset shows the 1000th iteration's final result using F2.

6.3 Iris Dataset Result

Initialize swarm with the 10 particles, and it is divided into 3 clusters with 1000 maximum iterations. Clustering diagrams show clustering results at fitness function F1 and fitness function F2, since Iris-setosa represents 0 in the cluster diagrams, Iris-versicolor represents 1 in the cluster diagrams, and Iris-virginca represents 2 in the clustering diagrams. The global best position for the data using fitness function F1 is given below:

1. At Iteration 0 global best = inf., no cluster at all.
2. At iteration number 200th global best position has changed to = 1.2607582879775.
3. At iteration number 400th global best position has changed to = 1.2516052460662213.
4. At iteration number 600th global best position has changed to = 1.2516033022038895.
5. At iteration number 800th global best position has changed to = 1.251603302200107.

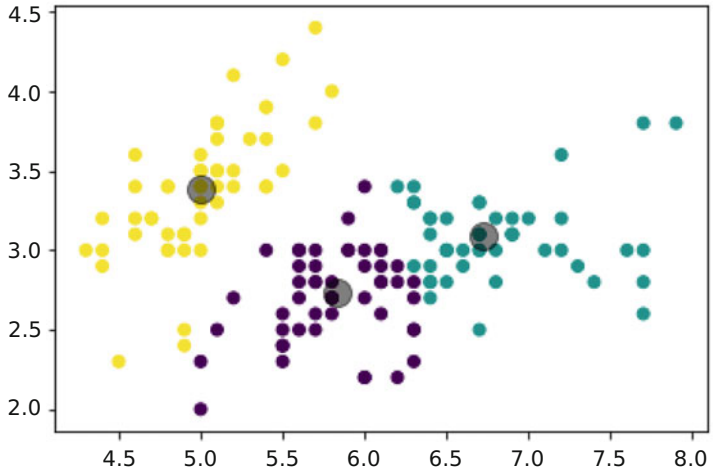


Fig. 5 Clustering found by Iris dataset with F1 fitness function

The global best position for the data using fitness function F2 is given below (Fig. 5):

1. At Iteration 0 global best = inf., no cluster at all.
2. At iteration number 200th global best position has changed to = 0.4284694709026384.
3. At iteration number 400th global best position has changed to = 0.4166974191684821.
4. At iteration number 600th global best position has changed to = 0.41667600698181073.
5. At iteration number 800th global best position has changed to = 0.4166634921453492 (Fig. 6).

6.4 Wine Dataset Result

Initialize swarm with 10 particles, and it is divided into 3 clusters with maximum 1000 iterations. Clustering diagrams show clustering results at fitness function F1 and fitness function F2. Clusters are named as 1, 2, and 3.

The global best position for the data using fitness function F1 is given below:

1. At Iteration 0 global best = inf., no cluster at all.
2. At iteration number 200th global best position has changed to = 1.5468292195294364.
3. At iteration number 400th global best position has changed to = 1.4363927211650254.

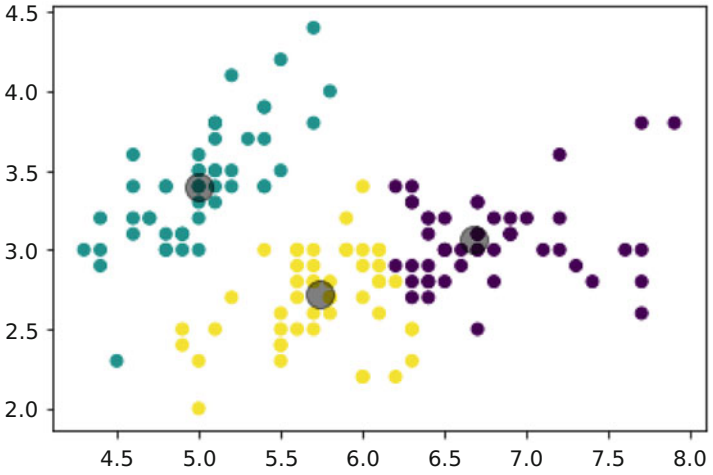


Fig. 6 Clustering found by Iris dataset with F2 fitness function

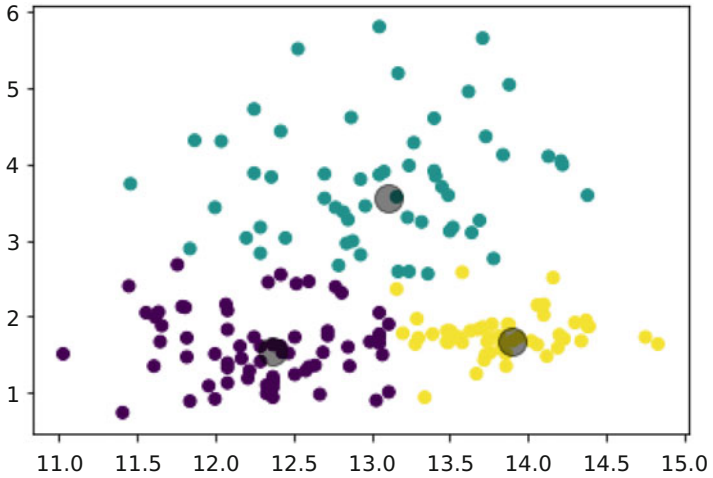


Fig. 7 Clustering found by wine dataset with F1 fitness function

4. At iteration number 600th global best position has changed to = 1.4280505080846657.
5. At iteration number 800th global best position has changed to = 1.4280480044212478 (Fig. 7).

The global best position for the data using fitness function F2 is given below:

1. At Iteration 0 global best = inf., no cluster at all.
2. At iteration number 200th global best position has changed to = 0.6440276048489663.

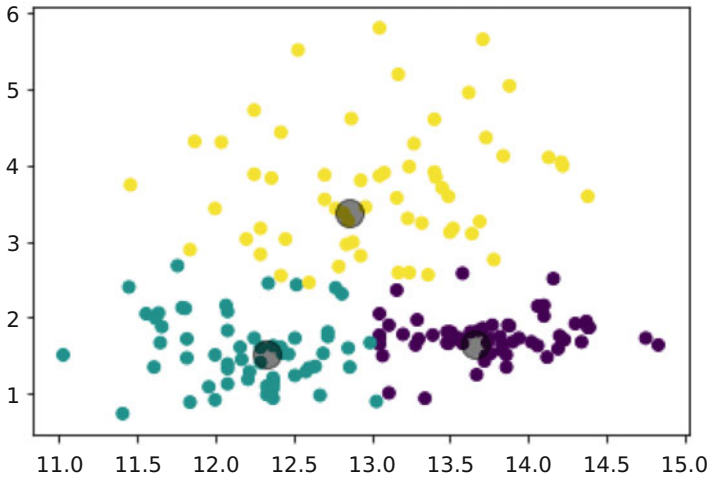


Fig. 8 Clustering found by wine dataset with F2 fitness function

3. At iteration number 400th global best position has changed to = 0.6396057502156508.
4. At iteration number 600th global best position has changed to = 0.6389733713896532.
5. At iteration number 800th global best position has changed to = 0.6389733713896532 (Fig. 8).

6.5 Glass Dataset Result

Initialize swarm with 10 particles, and it is divided into 7 clusters with maximum 1000 iterations. Clustering diagrams show clustering results at fitness function F1 and fitness function F2. Clusters are named as 1, 2, 3, 4, 5, 6, and 7.

The global best position for the data using fitness function F1 is given below (Fig. 9):

1. At Iteration 0 global best = inf., no cluster at all.
2. At iteration number 200th global best position has changed to = 1.0648539933120984.
3. At iteration number 400th global best position has changed to = 0.6817927458232369.
4. At iteration number 600th global best position has changed to = 0.6781102172151403.
5. At iteration number 800th global best position has changed to = 0.6764198237313302.

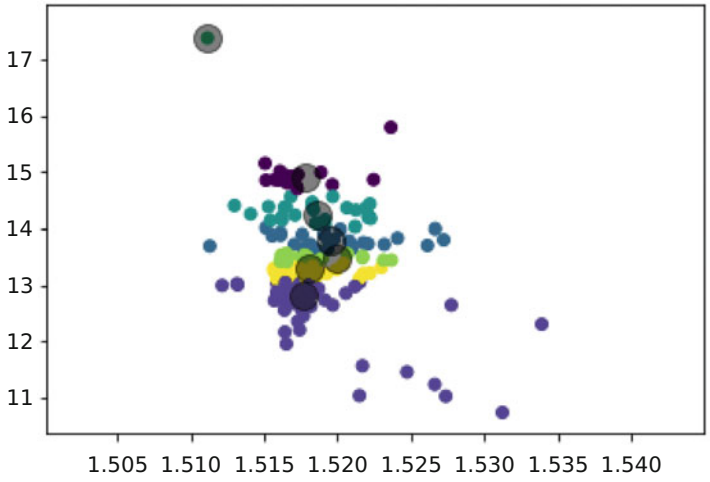


Fig. 9 Clustering found by glass dataset with F1 fitness function

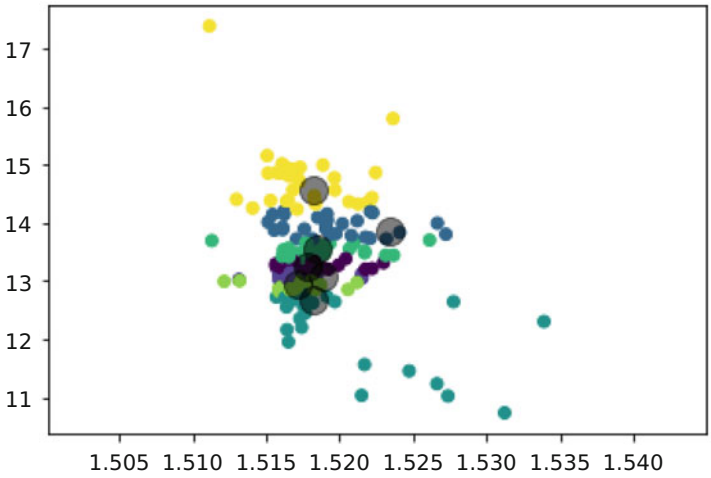


Fig. 10 Clustering found by glass dataset with F2 fitness function

The global best position for the data using fitness function F2 is given below (Fig. 10):

1. At Iteration 0 global best = inf., no cluster at all.
2. At iteration number 200th global best position has changed to = 2.3553043532612294.

3. At iteration number 400th global best position has changed to = 0.8487277072946797.
4. At iteration number 600th global best position has changed to = 0.8484994116264498.
5. At iteration number 800th global best position has changed to = 0.8484994116086169.

6.6 *Output of Ambient Dataset*

AMBIENT (AMBIgous ENTries) is a text dataset which is made for information retrieval process. It has 44 topics and each topics has its set of subtopics and a document list. It is one of the simplest datasets in which each query contains only single word.

6.6.1 Using TF-IDF (Figs. 11 and 12)

6.6.2 Using Word2Vec Model

The global best position for the data using fitness function F1 is given below (Fig. 13):

1. At Iteration 0 global best = inf., no cluster at all.
2. At iteration number 200th global best position has changed to = 0.006310132702471327.
3. At iteration number 400th global best position has changed to = 0.006310132702471327.

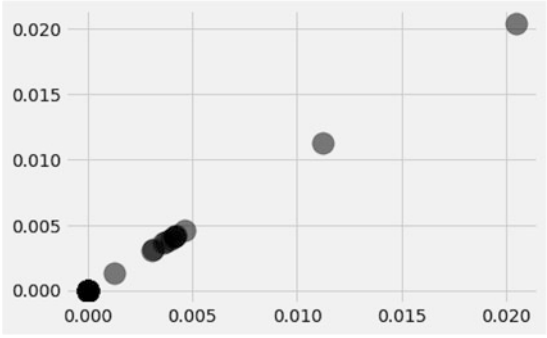


Fig. 11 Clusters centers' position for ambient dataset

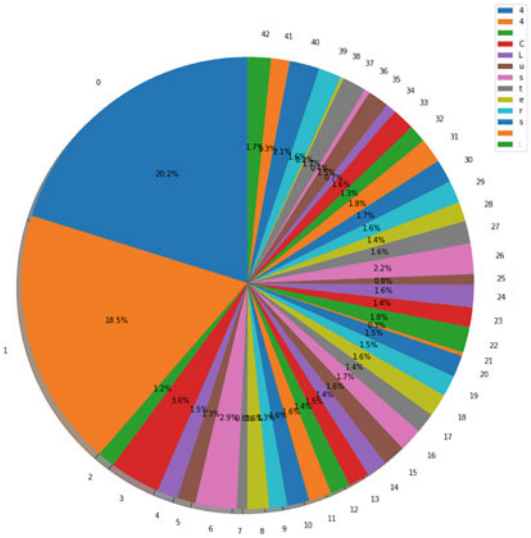


Fig. 12 Pie chart of ambient dataset which shows the percentage of snippets belonging to each queries

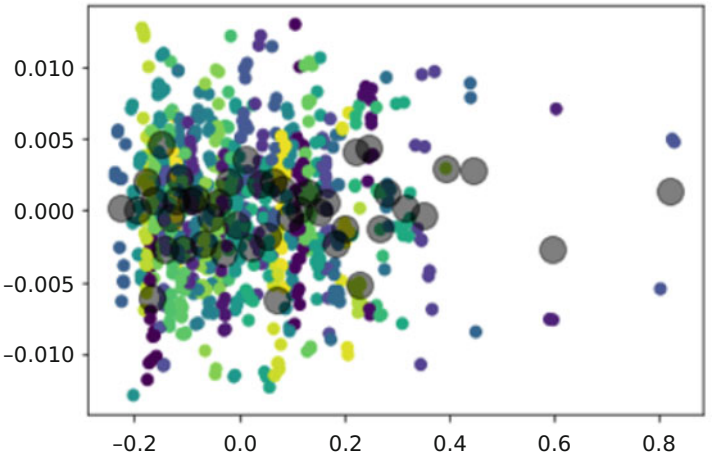


Fig. 13 Clustering found by ambient dataset with F1 fitness function

4. At iteration number 600th global best position has changed to = 0.006310132702471327.
5. At iteration number 800th global best position has changed to = 0.006310132702471327.

The global best position for the data using fitness function F2 is given below (Fig. 14):

1. At Iteration 0 global best = inf., no cluster at all.

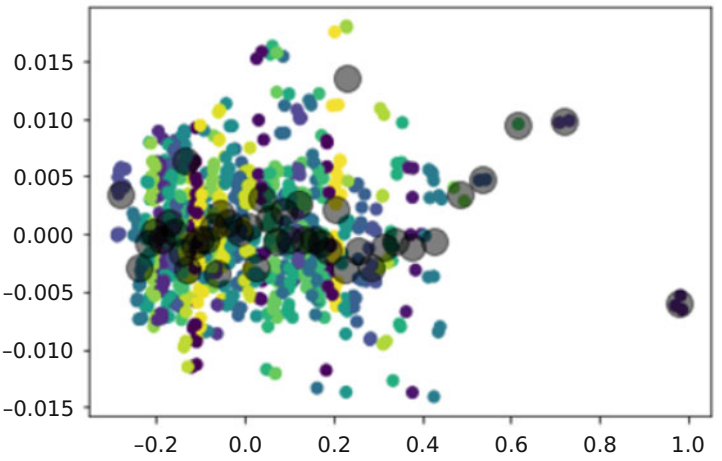


Fig. 14 Clustering found by ambient dataset with F2 fitness function

2. At iteration number 200th global best position has changed to = 0.003095565329400458.
3. At iteration number 400th global best position has changed to = 0.003095565329400458.
4. At iteration number 600th global best position has changed to = 0.003095565329400458.
5. At iteration number 800th global best position has changed to = 0.003095565329400458.

6.7 Output of ODP-239 Dataset

ODP is also a text data which is used for information retrieval as well. It has 239 topics and each topic has its set of subtopics and a document list. Each topic has 10 subtopics in dataset. In this dataset, subtopics are complex in nature and tough to distinguish as they have almost the same meaning. Thus, it is very difficult to cluster this dataset.

6.7.1 Using TF-IDF (Figs. 15 and 16)

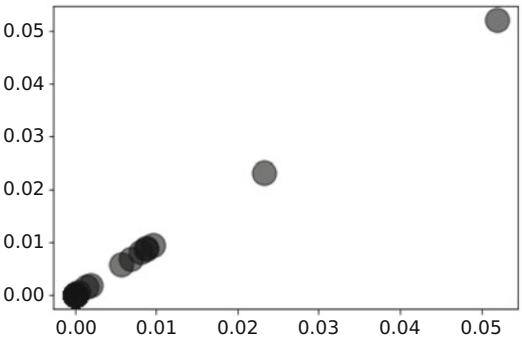


Fig. 15 Clusters centers' position for ODP-239 dataset

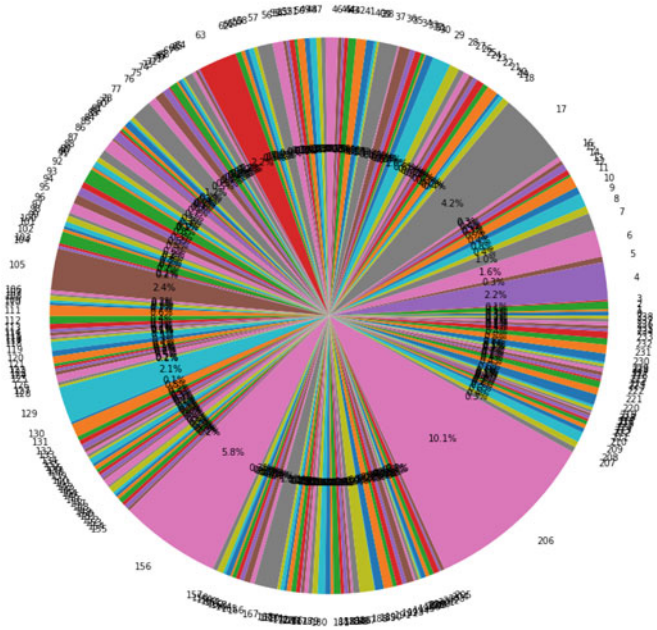


Fig. 16 Pie chart of ODP-239 dataset which shows the percentage of snippets belonging to each queries

6.7.2 Using Word2Vec

The global best position for the data using fitness function F1 is given below (Figs. 17 and 18):

At Iteration 0 global best = inf., no cluster at all.
At iteration number 200th global best position has changed to = 0.15706948111635832.
At iteration number 400th global best position has changed to = 0.15706948111635832.

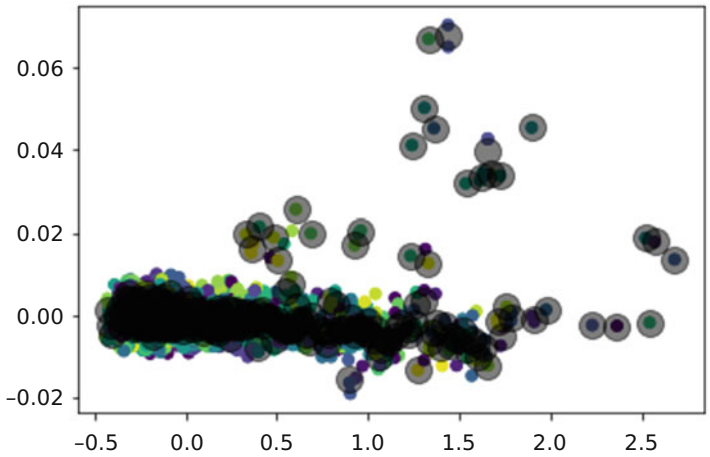


Fig. 17 Clustering found by ODP-239 dataset with F1 fitness function

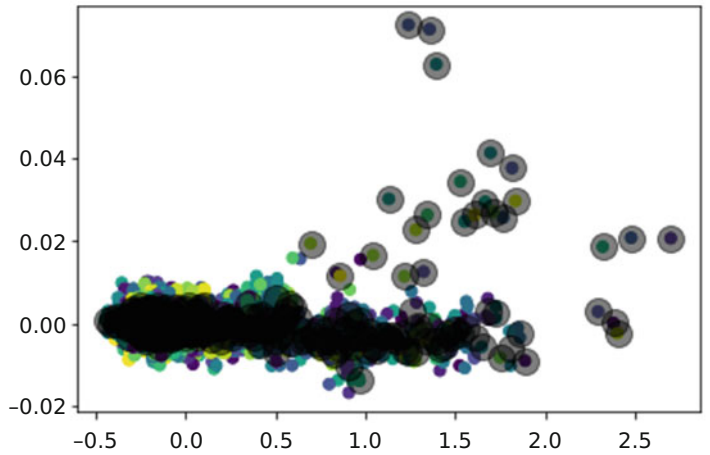


Fig. 18 Clustering found by ODP-239 dataset with F2 fitness function

At iteration number 600th global best position has changed to = 0.15706948111635832.
At iteration number 800th global best position has changed to = 0.15706948111635832.

The global best position for the data using fitness function F2 is given below:

1. At Iteration 0 global best = inf., no cluster at all.
2. At iteration number 200th global best position has changed to = 0.003095565329400458.
3. At iteration number 400th global best position has changed to = 0.003095565329400458.

4. At iteration number 600th global best position has changed to = 0.003095565329400458.
5. At iteration number 800th global best position has changed to = 0.003095565329400458.

In Figs. 1, 2, 3, 4, 5, 6, 9, 10, 13, 14 show the different clusters formed by the PSO algorithm using the mentioned datasets. Each sub-figure shows the graphical representation of the data points, and dark circle represents the cluster centroid. Data points having the same color show that these points belong to the same cluster. In each figure there are two sub-figures which show the cluster formation using PSO with fitness functions F1 and F2, respectively. It can be clearly seen from the figure that the clusters formed by modified fitness function which is F2 are better compared to the fitness function F1.

7 Comparison

7.1 Inter-cluster Distance (Table 3)

Table 3 Comparison of F1 and F2 for inter-cluster distance

	Inter-cluster distance	
Dataset	F1	F2
Iris	2.8135964819607278	1.029499701009877
Glass	12.912154766366982	12.021969574285691
Wine	10.77802997479919	10.75324414021391
Ambient	0.18278666080563996	0.3008418225727529
ODP-239	1.1388159506862974	0.7853967202005553

7.2 Intra-cluster Distance (Table 4)

Table 4 Comparison of F1 and F2 for intra-cluster distance

	Intra-cluster distance	
Dataset	F1	F2
Iris	1.029499701009877	1.0267426511778508
Glass	1.317904622674718	0.8242524627939393
Wine	1.570892216599712	1.568551430263879
Ambient	0.2181429533467094	0.3304259906933688
ODP-239	1.2188990146953118	0.8782585766503267

7.3 Global Best Position

7.4 Average Frequency of Clusters for Text Datasets Using TF-IDF

Since clustering is the most well-known problem in terms of optimization, several Metaheuristics algorithms (particle swarm optimization, Differential evolution, genetic algorithm, Cuckoo search, Black hole, hybrid algorithms, etc.) are applied to find optimal clusters numbers and how an algorithm can be used by modifying some parameters in order to improve the convergence speed and not to struck in local optima (Tables 5 and 6).

In Particle Swarm optimization inertial weight is added with 2 acceleration constants c_1 and c_2 in order to improve the PSO, and improved PSO shows a significant result compared to the standard PSO. It can also be improved by initial population clustering and other parameters as initial population is designed as a combination of fitness value and search space. Another improvement of PSO can be hybrid PSO in which FCM function can be improved, and this algorithm gives comparable results to the improvement of initialization of algorithm. PSO can be improved as multi-swarm optimization as well, this approach can improve the convergence speed compared to other PSO approaches, and it also enhance the diversity of the algorithm. Differential evolution framework has three parameters which can be modified to improve the optimization process by modifying the mutation factor, crossover constant, and population size. This (multi-objective multi view-based clustering using Differential Evolution Framework) approach performs better than the DE/rand/1. Here DE/rand/1 best denotes to the DE variant where rand denotes about the base vectors are chosen at random and 1 shows that the only one vector difference is used to form the mutated operation.

Table 5 Comparison of F1 and F2 for global best position

	Global best comparison	
Dataset	At F1	At F2
Iris	1.251603302200107	0.4166634921453492
Wine	1.4280480044212478	0.6389733713896532
Glass	0.6764198237313302	0.8484994116086169
Ambient	0.006310132702471327	0.005176546664728095
ODP-239	0.15706948111635832	0.003095565329400458

Table 6 Frequency of clusters average

Dataset	Average frequency per cluster	Average percentage per cluster
AMBIENT	98.1590909090909	2.230888429752066
ODP-239	107.0376569037657	0.4184427556832122

References

1. Van der Merwe, D. W., & Engelbrecht, A. P. (2003). Data clustering using particle swarm optimization. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03*. https://doi.org/10.1109/CEC.2003.1299577
2. Wang, X., Zhao, S., Jin, Y., & Zhang, L. (2013). Differential evolution algorithm based on self-adaptive adjustment mechanism. In *2013 25th Chinese Control and Decision Conference (CCDC)* (pp. 577–581). https://doi.org/10.1109/CCDC.2013.6560990
3. Aalto, J., & Lampinen, J. (2013). A mutation adaptation mechanism for differential evolution algorithm. In *2013 IEEE Congress on Evolutionary Computation* (pp. 55–62). https://doi.org/10.1109/CEC.2013.6557553
4. Aalto, J., & Lampinen, J. (2014). A mutation and crossover adaptation mechanism for differential evolution algorithm. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 451–458). https://doi.org/10.1109/CEC.2014.6900532
5. Yang, M., Li, C., Cai, Z., & Guan, J. (2015). Differential evolution with autoenhanced population diversity. *IEEE Transactions on Cybernetics, 45*(2), 302–315. https://doi.org/10.1109/TCYB.2014.2339495
6. Win Cho, P. P., & Thi Soe Nyunt, T. (2020). Data clustering based on differential evolution with modified mutation strategy. In *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)* (pp. 222–225). https://doi.org/10.1109/ECTI-CON49241.2020.9158243
7. Lai, C.-C., Lee, P.-F., Hsieh, P.-Y. (2008). Unsupervised clustering by means of hierarchical differential evolution algorithm. In *2008 Eighth International Conference on Intelligent Systems Design and Applications* (Vol. 2, pp. 297–301). https://doi.org/10.1109/ISDA.2008.173
8. Kulhari, A., Saraswat, M. (2017). Differential evolution-based subspace clustering via thresholding ridge regression. In *2017 Tenth International Conference on Contemporary Computing (IC3)* (pp. 1–3). https://doi.org/10.1109/IC3.2017.8284359
9. Das, S., Abraham, A., & Konar, A. (2007). Automatic clustering using an improved differential evolution algorithm. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 38*(1), 218–237. https://doi.org/10.1109/TSMCA.2007.909595
10. Lee, W.-P., Chen, S.-W. (2010). Automatic clustering with differential evolution using cluster number oscillation method. In *2010 2nd International Workshop on Intelligent Systems and Applications* (pp. 1–4). https://doi.org/10.1109/IWISA.2010.5473289
11. Srinivas, V.S., Srikrishna, A., Eswara Reddy, B. (2018). Automatic clustering simultaneous feature subset selection using differential evolution. In *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)* (pp. 468–473). https://doi.org/10.1109/SPIN.2018.8474233
12. Hancer, E. (2018). A differential evolution approach for simultaneous clustering and feature selection. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)* (pp. 1–7). https://doi.org/10.1109/IDAP.2018.8620878
13. Dong, C.-R., Yeung, D. S., & Wang, X.-Z. (2013). An improved differential evolution and its application to determining feature weights in similarity based clustering. In *2013 International Conference on Machine Learning and Cybernetics* (Vol. 02, pp. 831–838). https://doi.org/10.1109/ICMLC.2013.6890399
14. Tam, H.-H., Ng, S.-C., Lui, A. K., & Leung, M.-F. (2017). Improved activation schema on automatic clustering using differential evolution algorithm. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1749–1756). https://doi.org/10.1109/CEC.2017.7969513
15. Liu, X., Mei, K., & Yu, S. (2020). Clustering algorithm in wireless sensor networks based on differential evolution algorithm. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)* (Vol. 1, pp. 478–482). https://doi.org/10.1109/ITNEC48623.2020.9085089

16. He, D., Chang, H., Chang, Q., & Liu, Y. (2010). Particle swarm optimization based on the initial population of clustering. In *2010 Sixth International Conference on Natural Computation (ICNC 2010)*. https://doi.org/10.1109/ICNC.2010.5582936
17. Toreini, E., & Mehrnejad, M. (2011). Clustering data with particle swarm optimization using a new fitness. In *2011 3rd Conference on Data Mining and Optimization (DMO)* (pp. 266–270). https://doi.org/10.1109/DMO.2011.5976539
18. Hai-Long, S. (2014). Research on fast clustering algorithm based on improved particle swarm optimization. In *2014 Fifth International Conference on Intelligent Systems Design and Engineering Applications* (pp. 798–802). https://doi.org/10.1109/ISDEA.2014.180
19. Kao, Y., & Lee, S.-Y. (2009). Combining k-means and particle swarm optimization for dynamic data clustering problems. In *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems* (Vol. 1, pp. 757–761). https://doi.org/10.1109/ICICISYS.2009.5358020
20. Swetha, K. P., & Devi, V. S. (2012). Feature weighting for clustering by particle swarm optimization. In *2012 Sixth International Conference on Genetic and Evolutionary Computing*. https://doi.org/10.1109/ICGEC.2012.94
21. Liu, W., Wang, Z., Liu, X., Zeng, N., Bell, D. (2019). A novel particle swarm optimization approach for patient clustering from emergency departments. *IEEE Transactions on Evolutionary Computation, 23*(4), 632–644. https://doi.org/10.1109/TEVC.2018.2878536
22. Shen, Y., Li, Y., Kang, H., et al. (2018). Multi-swarm particle swarm optimization algorithm based on clustering dynamic grouping. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)* (pp. 2012–2016). https://doi.org/10.1109/CompComm.2018.8780896
23. Agbaje, M. B., Ezugwu, A. E., & Els, R. (2019). Automatic data clustering using hybrid firefly particle swarm optimization algorithm. *IEEE Access, 7*, 184963–184984. https://doi.org/10.1109/ACCESS.2019.2960925
24. Ezugwu, A. E.-S., Agbaje, M. B., Aljojo, N., Els, R., Chiroma, H., & Elaziz, M. A. (2020). A comparative performance study of hybrid firefly algorithms for automatic data clustering. *IEEE Access, 8*, 121089–121118. https://doi.org/10.1109/ACCESS.2020.3006173
25. Bejinariu, S.-I., Rotaru, F., Luca, R., & Costin, H. (2020). Nature-inspired metaheuristics for high-dimensional data clustering. In *11th International Conference and Exposition on Electrical and Power Engineering (EPE 2020)*. https://doi.org/10.1109/EPE50722.2020.9305585
26. Marinakis, Y., Marinaki, M., & Matsatsinis, N. (2009). A hybrid discrete artificial bee colony - grasp algorithm for clustering. In *2009 International Conference on Computers & Industrial Engineering*. https://doi.org/10.1109/ICCIE.2009.5223810
27. Mohammed, O. S., Sewisy, A. A. A. M., & Taloba, A. I. (2020). Solving optimization problems using hybrid metaheuristics: Genetic algorithm and black hole algorithm. In *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*. https://doi.org/10.1109/ICCIS49240.2020.9257717
28. Kumari, A. C., Srinivas, K., & Gupta, M. P. (2013). Software module clustering using a hyper-heuristic based multi-objective genetic algorithm. In *2013 3rd IEEE International Advance Computing Conference (IACC)*. https://doi.org/10.1109/IAdCC.2013.6514331
29. Abdulwahab, H. A., Noraziah, A., Alsewari, A. A., & Salih, S. Q. (2019). An enhanced version of black hole algorithm via Levy flight for optimization and data clustering problems. *IEEE Access, 7*, 142085–142096. https://doi.org/10.1109/ACCESS.2019.2937021

Index

V. Garg et al. (eds.), *Women in Soft Computing*, Women in Engineering and Science,
https://doi.org/10.1007/978-3-031-44706-8