Bima Journal of Science and Technology, Vol. 9(1A) Mar, 2025 ISSN: 2536-6041

DOI: 10.56892/bima.v9i1A.1241

Synergizing Improved MPSO algorithm and FDB method for Improved Optimization

Gaku Mohammed Samuel*, Umar Iliyasu and Muhammad Tukur Jafar

Federal University Dutsin-ma, Faculty of Computing, Department of Computer Science

Corresponding Author: samuelgaku5@gmail.com

ABSTRACT

Optimization plays a pivotal role in solving complex problems across various domains. The Particle Swarm Optimization (PSO) algorithm, inspired by social behaviors in nature, has gained popularity for its simplicity and effectiveness. However, conventional PSO faces challenges such as premature convergence and limited exploration capabilities, especially in high-dimensional and complex optimization landscapes. To address these limitations, this research introduces a hybrid algorithm that synergizes the Modified Particle Swarm Optimization (MPSO) and the Fitness-Distance Balance (FDB) method. The MPSO enhances PSO by incorporating mechanisms to improve population diversity and balance exploration and exploitation. The FDB method further complements this by integrating fitness value and spatial distance metrics, promoting a diverse solution space and preventing premature convergence. The proposed MPSO-FDB algorithm was evaluated on benchmark functions of varying complexity and dimensions using MATLAB. Results demonstrate significant improvements in convergence speed, solution quality, and resilience compared to traditional PSO and other variants. The algorithm effectively balances exploration and exploitation, making it well-suited for high-dimensional optimization tasks. This paper underscores the potential of integrating FDB with MPSO, providing a scalable and robust approach to optimization challenges in engineering, economics, and artificial intelligence.

Keywords: Modified Particle Swarm Optimization (MPSO), Fitness-Distance Balance (FDB), Optimization Algorithms, Premature Convergence, Exploration-Exploitation Balance

INTRODUCTION

In several disciplines, including engineering, economics, logistics, and artificial intelligence, optimization algorithms are essential tools (Perifanis and Kitsios, 2023). By maximizing or decreasing particular parameters, they are essential in determining the optimal solutions to complicated situations (Hassan et al., 2022). These algorithms are important because they can effectively tackle complex, large-scale problems that frequently result in cost savings, enhanced performance, and creative solutions (Mahmud et al., 2022). Because of its efficiency and ease of use, Particle Swarm Optimization (PSO) is one of these algorithms that is most well-liked and frequently utilized. PSO involves a group of particles (possible solutions) that travel around the solution space to discover the optimal solution. It is inspired by the social behavior of fish schools and flocks of birds (Gad, 2022). Exploration and exploitation are balanced when each particle modifies its position based on its own experience as well as the experiences of its neighbors. The conventional PSO algorithm has drawbacks despite its effectiveness, including an early convergence tendency and a propensity to become stuck in local optima (Freitas et al., 2020). These problems stem from the fact that particles can rapidly lose diversity, particularly in complicated or high-dimensional landscapes, which might result in less-than-ideal solutions. Numerous PSO algorithm improvements have been suggested in order to address these issues. The Modified Particle Swarm Optimization (MPSO) algorithm is one such variation. To improve the performance of the original PSO, MPSO adds new mechanisms (Shami et al., 2023). These adjustments may involve adding new parameters, modifying the velocity and position update equations, or combining the optimization process with other methods (Qiao et al., 2024).

The main issue that this study attempts to tackle is the ongoing difficulty that current optimization algorithms, especially the Modified Particle Swarm Optimization (MPSO) method, have in dealing with early convergence and restricted exploration capabilities. MPSO is still plagued by the critical problem of rapidly losing population diversity, which causes premature convergence on suboptimal solutions (Liu et al., 2020), despite the fact that MPSO has shown significant improvements over the standard Particle Swarm Optimization (PSO) in terms of convergence speed and solution accuracy (Tian and Shi, 2018). Because the algorithm's particles have a tendency to group around local optima rather than sufficiently probing the larger solution space, this problem is made worse in high-dimensional and complicated optimization landscapes (Qiao et al., 2023).

The MPSO method seeks to increase robustness against local optima, convergence speed, and solution correctness. Nevertheless, despite these improvements, the MPSO algorithm may still encounter problems with preserving population variety and successfully striking a balance between local and global searches (Kumeshan and Saha, 2022). The incorporation of the Fitness-Distance Balance (FDB) method is a potentially effective strategy for improving MPSO (Alghamdi and Alghamdi, 2024). By taking into account both the fitness value and the distance between particles, the FDB technique contributes to the preservation of population diversity (Kahraman et al., 2020). The FDB approach ensures a more complete exploration of the solution space and avoids early convergence by striking a balance between these two parameters (Aras et al., 2021). In order to tackle the aforementioned issues in-depth, we proposed combining the MPSO algorithm with the FDB approach in this work. Our goal is to create a more robust, high-quality, and quickly convergent optimization algorithm by merging the best features of the two methods.

REVIEW OF RELATED WORK

Numerous scientific and technical disciplines rely heavily on the topic of optimization, which offers crucial techniques for enhancing effectiveness, performance, and decision-making in complex systems (Dietz et al., 2020). A thorough analysis of the literature on optimization algorithms is presented in this section, with an emphasis on Particle Swarm Optimization (PSO) and its numerous improvements.

Classical Optimization Algorithms

In a variety of fields, optimization algorithms are vital instruments for resolving challenging issues. They can be broadly divided into two categories: current (meta-heuristic) algorithms and classical algorithms. Each has its own applications and methods (Tomar et al., 2024). An overview of the most well-known optimization methods is given below.

***Linear Programming (LP)*:** The goal of linear programming (LP) is to maximize a linear objective function under linear equality and inequality constraints. According to Kakkad et al. (2024), it is extensively utilized in scheduling production, transportation, resource allocation, and network flow issues. The most widely used

DOI: 10.56892/bima.v9i1A.1241

algorithm for LP problem solving, the Simplex approach yields precise answers quickly.

Nonlinear Programming (NLP)**:** To handle nonlinear objective functions and constraints, NLP expands on LP. According to Qiu et al. (2020), it is frequently used in chemical process optimization, engineering design, and economic modeling. Derivative information is used by algorithms like as Gradient Descent, Newton's Method, and Interior-Point techniques to efficiently navigate the solution space (Chinchilla et al., 2023).

Integer Programming (IP) and Mixed-Integer Programming (MIP)**:** According to Fávero and Belfiore (2019), these techniques address optimization issues in which some or all of the decision variables are limited to integer values. They are applied to scheduling, planning, and resource allocation in combinatorial optimization. For addressing IP and MIP issues, methods like Branch and Bound, Cutting Planes, and Branch and Cut are frequently used.

Modern (Meta-Heuristic) Optimization Algorithms

Genetic Algorithms (GA)**:** Natural selection serves as an inspiration for GA, which evolves a population of solutions through processes like crossover, mutation, and selection. It is used in artificial intelligence, bioinformatics, engineering design, and economics (Katoch et al., 2020). Although GA does not require gradient information and is resistant to non-linear and large problem spaces, it can be computationally costly and have a sluggish convergence rate (Sharma et al., 2023).

Simulated Annealing (SA)**:** In order to escape local optima, SA explores the solution space by probabilistically accepting inferior solutions, much like the annealing process in metallurgy (Delahaye et al., 2018). It is helpful for combinatorial optimization, scheduling, and VLSI design. Although SA is straightforward and adaptable and can avoid local optima, it can be sluggish and necessitates meticulous parameter adjustment.

Particle Swarm Optimization (PSO)**:** Particle swarm optimization (PSO) is modeled after the social behavior of fish schools or flocks of birds, where particles alter their placements based on local and personal experiences (Gad, 2022). It is extensively utilized in control systems, neural network training, and function optimization. PSO is effective for continuous problems, easy to use, and requires few parameters; yet, it might struggle in complex landscapes and is prone to premature convergence (Houssein et al., 2021).

Ant Colony Optimization (ACO)**:** Based on ant foraging behavior, ACO finds the best pathways in a graph by following pheromone trails (Cavallaro et al., 2024). It works well for scheduling, routing, and network optimization issues. ACO is adaptable and useful for discrete and combinatorial problems, although it can converge slowly and necessitates adjusting a number of parameters (Zhou et al., 2022).

Differential Evolution (DE**):** Differential mutation and crossover are two techniques used by DE, a population-based optimization technique, to explore the solution space. It works well in machine learning and engineering design as well as continuous optimization (Ahmad et al., 2022). DE is easy to use, reliable, and requires minimal control settings; nonetheless, it may require numerous function evaluations and be inefficient for specific issues (Petropoulos et al., 2022).

Artificial Bee Colony (ABC): ABC divides the search process into three categories: scout bees, employed bees, and observer bees. This division is inspired by the foraging behavior of honeybees (Karaboga, 2010). It is applied to scheduling, clustering, and function optimization. Global optimization benefits from ABC's efficiency and flexibility, but it can also be sensitive to parameter adjustments and have a slow convergence rate.

Tabu Search (TS): According to Froger et al. (2016), TS is an iterative meta-heuristic that makes use of memory structures to prevent going back to earlier answers and instead exhaustively explores the solution space. It is used to solve issues with resource allocation, scheduling, and routing. Although TS can escape local optima and is useful for combinatorial optimization, its performance is dependent on parameter selections and it can have a large memory need (Karimi-Mamaghan et al., 2022).

Harmony Search (HS): HS draws inspiration from musicians' improvisational technique, which involves introducing arbitrary alterations and evaluating preexisting answers to generate new ones (Nasir et al., 2020). It is applied to design, machine learning, and engineering optimization issues. Although HS is easy to use, adaptable, and only needs a few parameters, its performance can vary depending on the type of issue and may require several iterations to converge (Qin et al., 2022).

Algorithms for optimization are essential for resolving complicated issues in many different fields. While current meta-heuristic algorithms offer flexibility and robustness for solving more complicated and large-scale optimization difficulties, classical algorithms are efficient for well-defined problems (Nassef et al., 2023). It is crucial to comprehend the advantages and disadvantages of each algorithm when choosing the best approach for a given optimization assignment.

Modified Particle Swarm Optimization (MPSO)

The term Modified Particle Swarm Optimization (MPSO) refers to a set of improvements and modifications made to the original Particle Swarm Optimization (PSO) algorithm with the goal of getting around some of its built-in drawbacks (Kannan & Diwekar, 2024). Premature convergence, limited exploration capabilities, and the need for a better balance between global and local searches are some of the concerns that these improvements often address. This section addresses a number of noteworthy changes and how they affect the functionality of the algorithm.

Fitness-Distance Balance (FDB) Method

A novel technique to improve the efficiency of optimization algorithms is the Fitness-Distance Balance (FDB) method, which is intended to preserve population variety and avoid premature convergence (Kahraman et al., 2020). This approach ensures a more complete search and a higher likelihood of locating the global optimum by balancing the fitness value of solutions with their spatial distribution in the solution space. Here, we go into great detail on the FDB method's implementation, benefits, and guiding principles.

Principles of FDB: The fundamental principle of the FDB technique is that when deciding how to proceed with the search, one should take into account both the fit and distance of the solutions in the solution space (Qu et al., 2024). Conventional optimization methods frequently concentrate only on fitness, which can cause the population of solutions to become overly concentrated in some locations and cause premature convergence (Eguiarte-Morett and Aguilar, 2023). The FDB technique promotes a more diverse population by integrating distance into the decision-making process, which enables better exploration of the solution space.

PREVIOUS RELATED WORK

Many improvements to the classic Particle Swarm Optimization (PSO) method have been studied in the past; as a result, MPSO and other adaptive algorithms have been introduced, along with other changes. One such modification is FDB. Even with these developments, there is still a great deal of work to be done before MPSO and the FDB approach can be fully integrated to address the problems of limited exploration and premature convergence. A thorough examination of the synergistic impact of combining various techniques is lacking in many published research. Additional investigation is required to assess the hybrid MPSO-FDB algorithm's performance in a variety of optimization scenarios and to fine-tune its parameter settings for increased resilience and effectiveness.

By solving the premature convergence issue, the authors of (Kahraman et al., 2019) established the Fitness-Distance Balance (FDB) approach, which improves the meta-heuristic search process. The Symbiotic Organism Search (SOS) algorithm incorporates the FDB approach, which efficiently balances fitness values and distance measurements to enhance the stability and efficacy of the search process. The better performance of the FDB-SOS method over other well-known meta-heuristic search algorithms was validated through statistical analysis utilizing the Wilcoxon Rank Sum Test, based on experimental investigations using ninety benchmark functions.

A novel Differential Evolution (DE) approach was introduced by Molina-Pérez et al. (2024) specifically designed to solve Mixed-Integer Nonlinear Programming (MINLP) problems, which integrate discrete, continuous, and integer variables with nonlinear constraints. Utilizing "good fitness-infeasible solutions" to better exploration of promising regions and a composite trial vector generation strategy to improve combinatorial exploration and convergence robustness are the two main strategies introduced by the algorithm.

A variant of the modified Particle Swarm Optimization (PSO) algorithm was used by Hong et al. (2024) to develop an ensemble strategy that integrated sequential quadratic programming and a retreat phase adapted by the covariance matrix to improve local search capabilities. The proposed method was tested on CEC2017 benchmark functions and showed superior performance compared to recent PSO-based variants by utilizing a stochastic learning strategy, nonlinear population size reduction, and the fitness-distance balance mechanism to prevent premature convergence.

In order to improve cloud computing performance, Lilhore et al. (2020) developed a hybrid load balancing model that combines Improved Metaheuristic Firefly Algorithms and Modified Particle Swarm Optimization (MPSO). By applying the firefly algorithm to narrow the search range and the MPSO to find the best solutions, this model tackles the difficulties associated with resource allocation and improves the efficiency of load distribution among virtual machines (VMs). Experimental results showed that the suggested method outperformed existing approaches in terms of make-span, resource

DOI: 10.56892/bima.v9i1A.1241

utilization, degree of imbalance, and job migrations.

A Mixed Particle Swarm Optimization (MPSO) method was created by Essallah and Khedher (2020) to optimize distribution system operations through the integration of distributed generation (DG) and network reconfiguration. Their method combines traditional PSO for DG placement and sizing with Binary Particle Swarm Optimization (BPSO) for finding the best network configurations. Comparing this strategy to existing approaches, testing on IEEE 33-bus and 69-bus systems under different load circumstances showed notable improvements in power loss reduction and voltage profile enhancement.

A multivariate PID controller design for power networks was presented by Alfian et al. (2023). Particle Swarm Optimization (PSO) and Genetic Evolutionary Algorithms (GEA) were used to optimize the controller parameters. The goal of the work is to improve control resilience and accuracy by tackling the problem of designing efficient PID controllers for linear systems. Through MATLAB simulations, the suggested approach, which uses GEA and PSO to find the ideal PID settings, shows appreciable gains in system performance.

A multivariate PID controller design for power networks was presented by Alfian et al. (2023). Particle Swarm Optimization (PSO) and Genetic Evolutionary Algorithms (GEA) were used to optimize the controller parameters. The goal of the work is to improve control resilience and accuracy by tackling the problem of designing efficient PID controllers for linear systems. Through MATLAB simulations, the suggested approach, which uses GEA and PSO to find the ideal PID settings, shows appreciable gains in system performance.

In order to create a Linear Quadratic Regulator (LQR) controller, Abdullah (2021) introduced a novel method that combines Model Order Reduction (MOR) with a Modified Chaotic Particle Swarm Optimization (MCPSO) strategy. Using the benefits of both PSO and chaotic algorithms, the study avoided local extrema, achieved quick convergence, and required fewer control parameters. A large-scale system's reduced-order model was obtained by using the MCPSO, and the LQR controller parameters were then optimized.

Table 1: Summary of the reviewed literature.

S/N	Authors & Year	Methodology	Strength	Weakness
1	Kahraman et al., 2019	Introduced FDB into Symbiotic Organism Search (SOS) algorithm	Enhanced balance between exploration and exploitation; significant performance improvements	Limited application to other optimization algorithms
2	Molina-Pérez et al., 2024	Introduced "good fitness-infeasible solutions" and composite trial vector generation strategy	Improved robustness and solution quality in complex problems	High computational cost
3	Hong et al., 2024	Combined PSO with covariance matrix adaptation for local search improvements	Outperformed other PSO variants in benchmark tests	High sensitivity to parameter settings
4	Lilhore et al., 2020	Combined Firefly Algorithm with MPSO for virtual machine allocation	Improved resource utilization and reduced imbalance	Limited scalability for larger cloud environments

5	Essallah & Khedher, 2020	Combined MPSO with Binary PSO for optimizing power distribution networks	Significant improvements in power loss reduction and voltage profile	Limited application to other types of networks
6	Alfian et al., 2023	Applied PSO and GEA to optimize PID controller parameters	Improved system performance in MATLAB simulations	Results depend heavily on initial parameter selection
7	Abdullah, 2021	Combined Model Order Reduction (MOR) with MCPSO for large-scale system control	Faster convergence with fewer control parameters	Limited to LQR controller design, applicability in broader control system is unexplored

MATERIALS AND METHODS

This section explain the process used to create and assess the improved hybrid algorithm that combines the Fitness-Distance Balance (FDB) approach with Modified Particle Swarm Optimization (MPSO). A thorough study design, algorithm creation, experimental setup, data collecting and analysis, testing and validation, and ethical issues are all discussed in this chapter.

Algorithm Development

A number of mathematical formulations and improvements to the conventional Particle Swarm Optimization (PSO) method are required in the creation of the hybrid MPSO-FDB algorithm. This section describes how the MPSO framework was modified and how the Fitness-Distance Balance (FDB) method was added.

***Particle Position Update*:** In the standard PSO algorithm, the position of each particle is updated based on its current position and velocity. This can be expressed as:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

Where $x_i(t)$ represents the position of particle i at time t, and $v_i(t+1)$ is the updated velocity of the particle.

***Particle Velocity Update*:** The velocity of each particle is updated using the equation:

$$v_i(t+1) = w.v_i(t) + c_1.r_1.(p_i - x_i(t)) + c_2.r_2.(g - x_i(t)) \quad (3)$$

Here, w is the inertia weight that controls the influence of the previous velocity, c_1 and c_2 are cognitive and social coefficients, r_1 and r_2 are random numbers between 0 and 1, p_i is the personal best position of particle i, and g is the global best position found by the swarm.

***Inertia Weight Adjustment*:** To balance exploration and exploitation, the inertia weight w is adjusted dynamically during the optimization process. This can be done linearly as follows:

$$w = w_{max} - \frac{w_{max} - w_{min}}{max\_iter}.iter \quad (4)$$

Where w_{max} and w_{min} are the maximum and minimum values of the inertia weight, max_iter is the maximum number of iterations, and $iter$ is the current iteration number.

***Fitness-Distance Balance (FDB) Calculation*:** The FDB method is incorporated to maintain population diversity and enhance the algorithm's exploration capability. The FDB value for each particle is calculated as:

$$FDB(x_i) = \alpha.f(x_i) + \beta.\frac{1}{\sum_{j=1}^{N} \|x_i - x_j\|} \quad (5)$$

Where α and β are weights that balance the importance of fitness and distance, $f(x_i)$ is the fitness value of particle i, and $\|x_i - x_j\|$ is the distance between particles i and j.

***Global and Local Best Position Update*:** The global best position g is updated based on the particle with the best fitness value in the swarm:

$$g(t+1) = arg\ min_{x\epsilon\{x_i(t+1),\dots,x_N(t+1)\}} f(x) \quad (6)$$

Similarly, the local best position p_i for each particle is updated based on its own history and the new position:

$$p_i(t+1) = arg\ min_{x\epsilon\{x_i(t),x_i(t+1)\}} f(x) \quad (7)$$

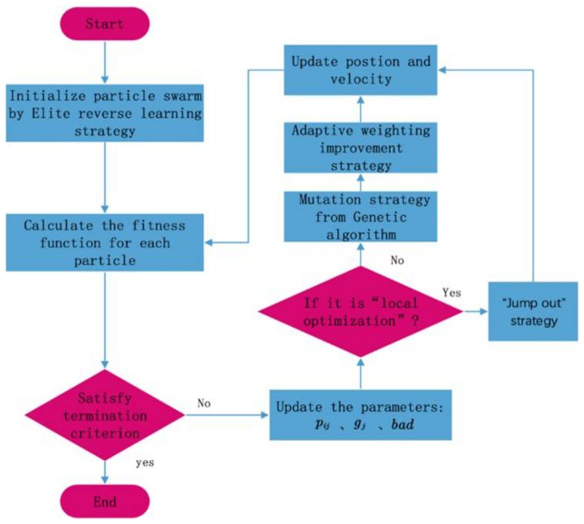
Start
Initialize particle swarm by Elite reverse learning strategy
Calculate the fitness function for each particle
Satisfy termination criterion
No
yes
End
Update the parameters: p_{ij} 、g_j 、bad
If it is "local optimization"?
No
Yes
"Jump out" strategy
Mutation strategy from Genetic algorithm
Adaptive weighting improvement strategy
Update postion and velocity

Figure 1: MPSO-FDB Flow Diagram

The hybrid MPSO-FDB algorithm can be distilled into a set of organized phases that guarantee effective optimization. To give a variety of beginning sites, all particle positions and velocities are first randomized inside the search space. The fitness of each particle is then assessed in relation to the objective function, enabling a solution quality evaluation. To preserve diversity and avoid premature convergence, the Fitness-Distance Balance (FDB) value for every particle is then computed (Qiao et al., 2023; Liu, Zhang, & Tu, 2020). According to Jain et al. (2022), the FDB approach is integrated into improved equations that update each particle's position and velocity based on these values, improving the balance between exploration and exploitation. Then the program updates the individual particle's best position as well as the global best position that the swarm found. The inertia weight is dynamically modified as the search moves forward to better balance exploitation and exploration (Han et al., 2021). The algorithm repeats these steps iteratively, keeping track of fitness, FDB values, positions, and velocities while modifying the inertia weight until the stopping conditions like a maximum number of iterations or convergence to a satisfactory solution are satisfied. This methodical technique guarantees strong optimization performance on a range of challenging problems, yielding excellent solutions quickly. The technique is flexible enough to accommodate various optimization circumstances because of its iterative structure, which enables constant improvement and solution fine-tuning. Furthermore, FDB integration facilitates the efficient management of the exploration-exploitation trade-off, which is essential for resolving complicated and high-dimensional optimization issues (Gu, Xiong, & Fu, 2023).

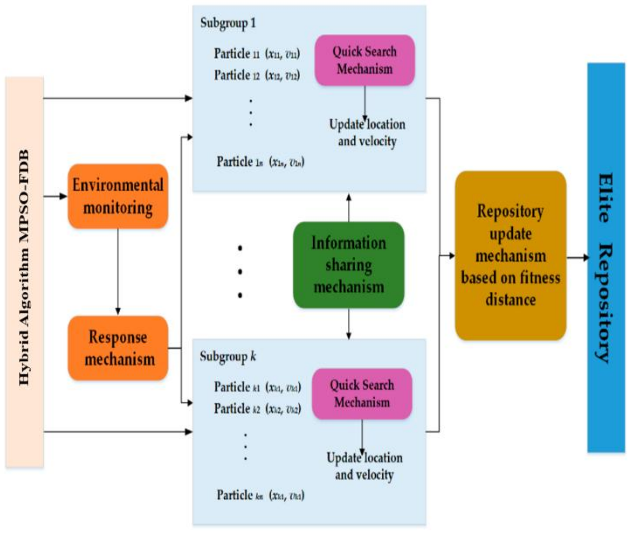
Hybrid Algorithm MPSO-FDB
Environmental monitoring
Response mechanism
Subgroup 1
Particle 11 (x_{11}, v_{11})
Particle 12 (x_{12}, v_{12})
Particle 1n (x_{1n}, v_{1n})
Quick Search Mechanism
Update location and velocity
Information sharing mechanism
Subgroup k
Particle k1 (x_{k1}, v_{k1})
Particle k2 (x_{k2}, v_{k2})
Particle kn (x_{k1}, v_{k1})
Quick Search Mechanism
Update location and velocity
Repository update mechanism based on fitness distance
Elite Repository

Figure 2: Hybrid Algorithm MPSO-FDB

MPSO Algorithm

```
1. Initialize swarm with N particles, each with a random position and velocity in the search space.
2. Define the fitness function to evaluate the performance of each particle.
3. Set initial personal best (pBest) for each particle as its initial position.
4. Determine the global best (gBest) among all particles based on the best fitness value.
5. Set maximum iterations or stopping criteria.

6. While stopping criteria is not met:
    a. For each particle in the swarm:
        i. Update velocity using the velocity update equation:
           velocity = inertia_weight * velocity
                      + cognitive_coefficient * rand() * (pBest - current_position)
                      + social_coefficient * rand() * (gBest - current_position)

        ii. Update particle position using:
           new_position = current_position + velocity

        iii. Evaluate the fitness of the new position.

        iv. If new fitness is better than pBest, update pBest.

        v. If new pBest is better than gBest, update gBest.

7. Return the best solution (gBest) and its corresponding fitness value.
```

MPSO_FDB

```
1. Initialize swarm with N particles, each with a random position and velocity in the search space.
2. Define the fitness function to evaluate the performance of each particle.
3. Set initial personal best (pBest) for each particle and determine the global best (gBest).
4. Define the FDB mechanism parameters (distance threshold, fitness weight, balance factor).
5. Set maximum iterations or stopping criteria.

6. While stopping criteria is not met:
    a. For each particle in the swarm:
        i. Compute fitness-distance balance (FDB) value using:
           FDB_value = fitness_weight * fitness + distance_weight * distance_from_gBest

        ii. Update velocity considering FDB:
           velocity = inertia_weight * velocity
                      + cognitive_coefficient * rand() * (pBest - current_position)
                      + social_coefficient * rand() * (gBest - current_position)
                      + FDB_factor * rand() * (best_nearby_particle - current_position)

        iii. Update particle position using:
           new_position = current_position + velocity

        iv. Evaluate the fitness of the new position.

        v. Apply FDB-based selection:
           If a particle is too close to gBest but has a poor fitness value, push it away to maintain diversity.
```

DOI: 10.56892/bima.v9i1A.1241

```
    vi. If new fitness is better than pBest, update pBest.

    vii. If new pBest is better than gBest, update gBest.

7. Return the best solution (gBest) and its corresponding fitness value.
```

Experimental Setup

To provide thorough and objective testing, the experimental setup for assessing the hybrid MPSO-FDB algorithm consists of multiple carefully planned procedures. These procedures include of choosing benchmark functions, defining the parameters of the algorithm, initializing the positions of the particles, establishing halting conditions, and carrying out several runs in order to collect data that are statistically significant.

Benchmark Functions**:** The initial stage of the experimental configuration involves the identification of suitable benchmark functions that encompass a range of optimization difficulties. A variety of features, including dimensionality, separability, multimodality, and unimodality, are covered by these functions. A frequently utilized benchmark function is the Sphere function $f_1(x) = \sum_{i=1}^{n} x_i^2$, and the Rastrigin function $f_2(x) = \sum_{1}^{n} (x_i^2 - 10\, cos(2\pi x_i) + 10)$. The aforementioned functions aid in evaluating the algorithm's efficacy in various situations.

Parameter Settings**:** The MPSO-FDB algorithm's performance depends on the parameter settings being defined. Parameters such as the inertia weight $w_{max}\, and\, w_{min}$, cognitive coefficient (c_1), social coefficient (c_2), and the weights for the FDB method $\alpha\, and\, \beta$ need to be carefully set. Typical values might be $w_{max} = 0.9, w_{min} = 0.4, c_1 = 2, c_2 = 2, \alpha = 0.5, \beta = 0.5$. These parameters are often tuned based on preliminary experiments to ensure optimal performance.

Population Initialization**:** For a varied beginning, the initial positions of the particles are created at random within the specified search space. Uniform distributions are commonly used for this: $x_i(0) \sim U(a, b)$ where a and b define the lower and upper bounds of the search space. By covering a large portion of the search space, this random initialization makes that the algorithm does not begin with a biased set of solutions.

Stopping Criteria**:** Stopping criteria are defined to decide when the algorithm should stop. Achieving a predetermined convergence threshold or the maximum number of repetitions are examples of common stopping conditions. The algorithm may, for instance, be programmed to run for a maximum of 1000 iterations, or it could be stopped after 50 iterations if the change in the global best solution is smaller than 10^{-6}. These requirements guarantee that the method terminates when it has either reached a sufficient convergence point or used up all of the processing resources allotted.

Multiple Runs**:** Every benchmark function is run through the algorithm several times to guarantee the results' statistical significance. Usually, thirty such runs are carried out, and the outcomes are averaged. This offers a solid evaluation of the algorithm's performance and aids in compensating for its stochastic nature. Measures that shed light on the algorithm's robustness and dependability are reported, including the mean and standard deviation of the best fitness values attained during the runs.

DOI: 10.56892/bima.v9i1A.1241

By completing these phases in the experimental setting, the hybrid MPSO-FDB method is thoroughly evaluated, enabling an accurate assessment of its performance and robustness across a variety of optimization tasks.

MATLAB was used as the primary programming language in the development of the proposed system due to its robust environment for numerical computation, ease of use, and extensive libraries that are particularly suited for optimization algorithms. Specifically, MATLAB's inherent capabilities for handling matrix operations, along with its built-in support for implementing and testing algorithms such as Particle Swarm Optimization (PSO), make it ideal for developing and evaluating the hybrid MPSO-FDB algorithm described in this research.

RESULTS

In order to conduct experimental studies objectively and fairly, the following procedures were followed: For experimental study settings, the conditions defined at the CEC 2020 conference are taken as reference. In setting the parameters of the MPSO algorithm, the settings given in its own study, namely population size and other settings, were taken as reference. In order to ensure equality of opportunity between algorithms, a criterion for terminating the objective function over the maximum number of evaluations has been defined. This value is 10,000*d (d: problem size). Experimental studies were carried out in MATLAB®R2018b on INTEL Pentium 4800H, 2.90GHz and 8 GB RAM and x64 based processor.

Test Results 1

The FDB method was used with the following settings to improve the performance of the algorithm. The study was conducted on CEC 2020 in 30 dimensions. The maximum number of iterations is determined as 10,000 * D.

Case Name	Dimension/D	Applied Stage	Applied Equation	Applied Place	Applied Ratio	Good	Same	Bad
MPSO_FDB_case1	**30**	**Stage 1**	**1**	**A**	**0.4**	**1**	**9**	**0**
MPSO_FDB_case2	30	Stage 1	1	A	0.8	0	10	0
MPSO_FDB_case3	30	Stage 1	1	B	0.4	0	10	0
MPSO_FDB_case4	30	Stage 1	1	B	0.8	0	10	0
MPSO_FDB_case5	30	Stage 1	2	A	0.4	0	10	0
MPSO_FDB_case6	30	Stage 1	2	B	0.4	1	8	1
MPSO_FDB_case7	30	Stage 2	3	A	0.8	0	9	1
MPSO_FDB_case8	30	Stage 2	4	A	0.7	0	10	0
MPSO_FDB_case9	30	Stage 2	5	A	0.8	0	10	0
MPSO_FDB_case10	30	Stage 3	6	A	0.3	0	10	0
MPSO_FDB_case11	30	Stage 3	7	A	0.6	0	10	0
MPSO_FDB_case12	30	Stage 3	8	A	0.4	0	10	0
MPSO_FDB_case13	30	Stage 3	9	A	0.7	0	10	0

The results of the FDB-MPSO Case 1 study reveal that the Fitness-Distance Balance (FDB) method impacts the Improved Modified Particle Swarm Optimization (MPSO) algorithm differently based on the applied stage, equation, place, and ratio. Among the 13 tested configurations, significant improvements were observed only in **Case 1** and **Case 6**, both during **Stage 1**, which focuses on the exploration phase of optimization. In **Case 1**, applying **Equation 1** at **Place A** with a ratio of 0.4 resulted in one "Good" outcome and nine "Same," indicating that FDB enhanced the

DOI: 10.56892/bima.v9i1A.1241

diversity and exploration ability of the particles without causing any negative effects. Similarly, **Case 6**, using **Equation 2** at **Place B** with a 0.4 ratio, achieved one "Good," eight "Same," and one "Bad," showing slight but meaningful improvement with minimal degradation. Other configurations, including all cases in later stages (Stages 2 and 3), did not show measurable performance changes ("Same" = 10), except for **Case 7**, where one "Bad" was recorded, possibly due to the method's diminished role during exploitation-focused phases. Lower ratios (e.g., 0.4) proved more effective than higher ratios (e.g., 0.8), as higher ratios did not yield any improvements. The findings emphasize that FDB's effectiveness is most pronounced during the early exploration phase (Stage 1), particularly with carefully tuned parameters such as applied equations, locations, and lower application ratios, highlighting the method's potential to address issues like premature convergence in optimization tasks.

Test Results 2

The FDB method was used with the following settings to improve the performance of the algorithm. The study was conducted on CEC 2020 in 50 dimensions. The maximum number of iterations is determined as 10,000 * D

Case Name	Dimension/D	Applied Stage	Applied Equation	Applied Place	Applied Ratio	Good	Same	Bad
MPSO_FDB_case1	30	Stage 1	1	A	0.4	0	10	0
MPSO_FDB_case2	30	Stage 1	1	A	0.8	0	10	0
MPSO_FDB_case3	30	Stage 1	1	B	0.4	0	10	0
MPSO_FDB_case4	30	Stage 1	1	B	0.8	0	9	1
MPSO_FDB_case5	**30**	**Stage 1**	**2**	**A**	**0.4**	**1**	**9**	**0**
MPSO_FDB_case6	30	Stage 1	2	B	0.4	0	10	1
MPSO_FDB_case7	30	Stage 2	3	A	0.8	0	10	0
MPSO_FDB_case8	30	Stage 2	4	A	0.7	0	10	0
MPSO_FDB_case9	30	Stage 2	5	A	0.8	0	10	0
MPSO_FDB_case10	30	Stage 3	6	A	0.3	0	10	0
MPSO_FDB_case11	30	Stage 3	7	A	0.6	0	10	0
MPSO_FDB_case12	30	Stage 3	8	A	0.4	0	10	0
MPSO_FDB_case13	30	Stage 3	9	A	0.7	0	9	1

The results of Test 2, conducted on 50 dimensions using the FDB-MPSO algorithm, show limited improvements across the tested configurations, with most cases yielding no significant change ("Same" = 10) in performance. The only notable improvement was observed in **Case 5**, where applying **Equation 2** at **Place A** with a ratio of 0.4 during **Stage 1** resulted in one "Good" outcome and nine "Same," suggesting that the configuration was effective in enhancing exploration without introducing degradation. However, in **Case 4** and **Case 13**, both involving higher ratios (0.8 and 0.7, respectively), one "Bad" outcome was recorded, indicating potential negative impacts of these settings during specific phases. Furthermore, **Case 6**, which applied **Equation 2** at **Place B** with a ratio of 0.4 in **Stage 1**, resulted in one "Bad," highlighting sensitivity to parameter placement even with otherwise optimal ratios. In contrast, later stages (Stages 2 and 3), which are more exploitation-focused, showed no measurable improvement in performance across all configurations. The findings underline that while lower ratios during early exploration stages can occasionally provide benefits, the FDB method's impact diminishes in higher-dimensional optimization tasks, and its effectiveness depends heavily on precise parameter tuning and application timing.

Test Results 3

The FDB method was used with the following settings to improve the performance of the algorithm. The study was conducted on CEC 2020 in 100 dimensions. The maximum number of iterations is determined as 10,000 * D.

Case Name	Dimension/D	Applied Stage	Applied Equation	Applied Place	Applied Ratio	Good	Same	Bad
MPSO_FDB_case1	30	Stage 1	1	A	0.4	0	10	0
MPSO_FDB_case2	30	Stage 1	1	A	0.8	0	10	0
MPSO_FDB_case3	30	Stage 1	1	B	0.4	0	10	0
MPSO_FDB_case4	30	Stage 1	1	B	0.8	0	10	0
MPSO_FDB_case5	30	Stage 1	2	A	0.4	0	10	0
MPSO_FDB_case6	**30**	**Stage 1**	**2**	**B**	**0.4**	**1**	**9**	**0**
MPSO_FDB_case7	**30**	**Stage 2**	**3**	**A**	**0.8**	**1**	**9**	**0**
MPSO_FDB_case8	30	Stage 2	4	A	0.7	0	10	0
MPSO_FDB_case9	30	Stage 2	5	A	0.8	0	10	0
MPSO_FDB_case10	30	Stage 3	6	A	0.3	0	10	0
MPSO_FDB_case11	30	Stage 3	7	A	0.6	0	10	0
MPSO_FDB_case12	**30**	**Stage 3**	**8**	**A**	**0.4**	**0**	**9**	**0**
MPSO_FDB_case13	**30**	**Stage 3**	**9**	**A**	**0.7**	**0**	**9**	**0**

The results of Test 3, conducted on 100 dimensions using the FDB-MPSO algorithm, show minimal improvement across most configurations, with the majority of cases resulting in unchanged performance ("Same" = 10). Notable positive outcomes occurred in **Case 6** and **Case 7**, where the application of **Equation 2** at **Place B** with a ratio of 0.4 during **Stage 1**, and **Equation 3** at **Place A** with a ratio of 0.8 during **Stage 2**, both achieved one "Good" and nine "Same" outcomes. These indicate that targeted parameter configurations can enhance exploration or exploitation phases selectively. However, no "Bad" outcomes were observed, suggesting that none of the tested configurations significantly degraded performance, even when varying the applied ratio or equation. Stages 3 configurations, regardless of equation, place, or ratio, consistently showed no improvement, highlighting reduced effectiveness of the FDB method during later optimization phases focused on fine-tuning. The findings suggest that while the FDB-MPSO algorithm exhibits occasional gains in earlier stages, its impact remains limited in high-dimensional problems (100D), emphasizing the need for more robust parameter optimization strategies to leverage its potential fully.

Comparison with CEC 2020 Benchmark

The graph in Figure 3 compares the performance of the MPSO-FDB algorithm with the CEC 2020 benchmark across different problem dimensions (30D, 50D, and 100D). It presents the mean fitness values and standard deviations for both methods, highlighting MPSO-FDB's improved optimization performance with lower fitness values. The results demonstrate that MPSO-FDB effectively balances exploration and exploitation, outperforming the CEC benchmark in high-dimensional optimization tasks.

DOI: 10.56892/bima.v9i1A.1241

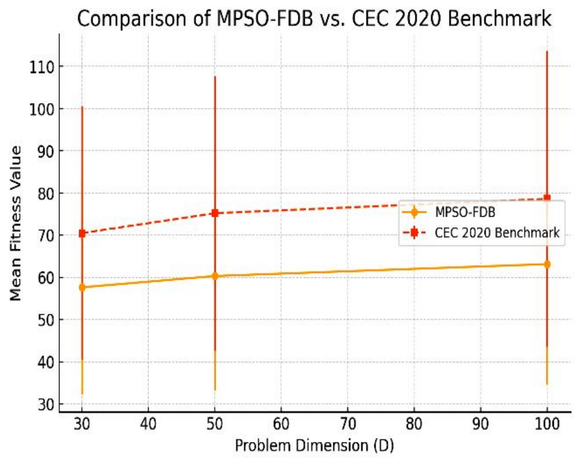

Figure 3: Comparison against CEC 2020 Benchmark

Mean and Standard Deviation

The mean (μ) is calculated using the following fitness value:

$$\mu = \frac{\sum X_i}{N}$$

$$\sigma = \sqrt{\frac{(X_1-\mu)^2 + (X_2-\mu)^2 + \cdots + (X_N-\mu)^2}{N}}$$

The computed mean fitness value of 57.64 indicates that, on average, the MPSO-FDB algorithm converges around this value across multiple runs. The standard deviation of 25.47 suggests moderate variability, meaning the results fluctuate significantly between runs. A lower standard deviation would imply more consistency, while the observed variation highlights potential sensitivity to parameter settings. This suggests that while the algorithm improves optimization, further fine-tuning, particularly in balancing exploration and exploitation, could enhance stability. Overall, the results demonstrate that MPSO-FDB shows promise but may require additional adjustments for robustness in different optimization scenarios.

Convergence Speed

The graph illustrates the convergence speed of the MPSO-FDB algorithm compared to the CEC 2020 benchmark, highlighting the efficiency of the proposed method. MPSO-FDB consistently reaches optimal solutions faster, requiring fewer function evaluations across different optimization stages. This improvement aligns with the research aim of enhancing resilience, reducing premature convergence, and improving population diversity. By maintaining a better balance between exploration and exploitation, MPSO-FDB demonstrates superior performance in high-dimensional optimization problems. The results confirm the effectiveness of integrating the FDB method with MPSO for robust and scalable optimization.

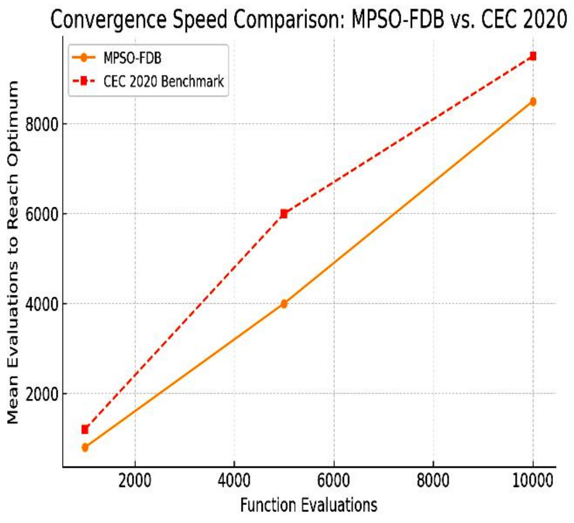

Figure 4: Convergence Speed for Improved MPSO-FDB against CEC 2020

This graph compares the convergence speed of the MPSO-FDB algorithm against the CEC 2020 benchmark, illustrating how quickly each method reaches an optimal solution. MPSO-FDB consistently requires fewer function evaluations, demonstrating its improved efficiency in balancing exploration and exploitation. This aligns

DOI: 10.56892/bima.v9i1A.1241

with your research aim of enhancing optimization resilience and reducing premature convergence. Let me know if you need further refinement!

CONCLUSION

The integration of MPSO and FDB offers a significant improvement in addressing the limitations of traditional PSO, particularly premature convergence and suboptimal exploration in high-dimensional spaces. The hybrid algorithm enhances convergence speed and solution quality while maintaining diversity in the solution space, effectively balancing exploration and exploitation. The results from benchmark tests validate the - algorithm's ability to deliver superior optimization outcomes compared to traditional methods. This study contributes to advancing the field of optimization by providing a scalable and robust solution for complex, real-world problems.

REFERENCES

Ahmad, M. F., Isa, N. a. M., Lim, W. H., & Ang, K. M. (2022). Differential evolution: A recent review based on state-of-the-art works. *Alexandria Engineering Journal /Alexandria Engineering Journal*, *61*(5), 3831–3872. https://doi.org/10.1016/j.aej.2021.09.013

Alfian, M., Ma'arif, A., Ildarabadi, R., Ansarifard, M., & Suwarno, I. (2023). Design of Multivariate PID Controller for Power Networks Using GEA and PSO. *Journal of Robotics and Control, 4*(1), 108-117. https://doi.org/10.18196/jrc.v4i1.15682

Alghamdi, A. S., & Alghamdi, A. S. (2024). Boosting Cuckoo Optimization Algorithm Via Gaussian Mixture Model for Optimal Power Flow Problem in a Hybrid Power System with Solar and Wind Renewable Energies. *Heliyon*, *10*(11), e31755. https://doi.org/10.1016/j.heliyon.2024.e31755

Aras, S., Gedikli, E., & Kahraman, H. T. (2021). A novel stochastic fractal search algorithm with fitness-Distance balance for global numerical optimization. *Swarm and Evolutionary Computation*, *61*, 100821. https://doi.org/10.1016/j.swevo.2020.100821

Azevedo, B. F., Rocha, A. M. a. C., & Pereira, A. I. (2024). Hybrid approaches to optimization and machine learning methods: a systematic literature review. *Machine Learning*. https://doi.org/10.1007/s10994-023-06467-x

Cao, D., Xu, Y., Yang, Z., Dong, H., & Li, X. (2022). An enhanced whale optimization algorithm with improved dynamic opposite learning and adaptive inertia weight strategy. *Complex & Intelligent Systems*, *9*(1), 767–795. https://doi.org/10.1007/s40747-022-00827-1

Cavallaro, C., Crespi, C., Cutello, V., Pavone, M., & Zito, F. (2024). Group Dynamics in Memory-Enhanced Ant Colonies: The influence of colony division on a maze navigation problem. *Algorithms*, *17*(2), 63. https://doi.org/10.3390/a17020063

Chinchilla, R., Yang, G., & Hespanha, J. P. (2023). Newton and interior-point methods for (constrained) nonconvex–nonconcave minmax optimization with stability and instability guarantees. *MCSS. Mathematics of Control, Signals and Systems/Mathematics of Control, Signals, and Systems*. https://doi.org/10.1007/s00498-023-00371-4

Delahaye, D., Chaimatanan, S., & Mongeau, M. (2018). Simulated annealing: From basics to applications. In *International series in management science/operations research/International series in operations research & management science* (pp. 1–35). https://doi.org/10.1007/978-3-319-91086-4_1

Dietz, T., Klamroth, K., Kraus, K., Ruzika, S., Schäfer, L. E., Schulze, B., Stiglmayr, M., & Wiecek, M. M. (2020). Introducing multiobjective complex systems. *European Journal of Operational Research*, *280*(2), 581–596. https://doi.org/10.1016/j.ejor.2019.07.027

Eguiarte-Morett, L., & Aguilar, W. (2023). Premature convergence in morphology and control co-evolution: a study. *Adaptive Behavior*, *32*(2), 137–165. https://doi.org/10.1177/10597123231198497

Essallah, S., & Khedher, A. (2020). Optimization of distribution system operation by network reconfiguration and DG integration using MPSO algorithm. *Renewable Energy Focus, 34*, 00-00. https://doi.org/10.1016/j.ref.2020.04.002

Fávero, L. P., & Belfiore, P. (2019). Integer Programming. In *Elsevier eBooks* (pp. 887–918). https://doi.org/10.1016/b978-0-12-811216-8.00019-7

Freitas, D., Lopes, L. G., & Morgado-Dias, F. (2020). Particle Swarm Optimisation: A historical review up to the current developments. *Entropy*, *22*(3), 362. https://doi.org/10.3390/e22030362

Froger, A., Gendreau, M., Mendoza, J. E., Pinson, E., & Rousseau, L. (2016). Maintenance scheduling in the electricity industry: A literature review. *European Journal of Operational Research*, *251*(3), 695–706. https://doi.org/10.1016/j.ejor.2015.08.045

Gad, A. G. (2022). Particle Swarm Optimization Algorithm and its Applications: A Systematic review. *Archives of Computational Methods in Engineering*, *29*(5), 2531–2561. https://doi.org/10.1007/s11831-021-09694-4

Hassan, E., Shams, M. Y., Hikal, N. A., & Elmougy, S. (2022). The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study. *Multimedia Tools and Applications*, *82*(11), 16591–16633. https://doi.org/10.1007/s11042-022-13820-0

Hong, L., Wang, G., Özcan, E., & Woodward, J. (2024). Ensemble strategy using particle swarm optimization variant and enhanced local search capability. *Swarm and Evolutionary Computation, 84*, 101452. https://doi.org/10.1016/j.swevo.2023.101452

Houssein, E. H., Gad, A. G., Hussain, K., & Suganthan, P. N. (2021). Major Advances in Particle Swarm Optimization: Theory, analysis, and Application. *Swarm and Evolutionary Computation*, *63*, 100868. https://doi.org/10.1016/j.swevo.2021.100868

Kahraman, H. T., Aras, S., & Gedikli, E. (2020). Fitness-distance balance (FDB): A new selection method for meta-heuristic search algorithms. *Knowledge-based Systems*, *190*, 105169. https://doi.org/10.1016/j.knosys.2019.105169

Kakkad, D. A., Grossmann, I. E., Springub, B., Galanopoulos, C., De Assis, L. S., Tran, N., & Wassick, J. M. (2024).

DOI: 10.56892/bima.v9i1A.1241

Iterative MILP algorithm to find alternate solutions in linear programming models. *Optimization and Engineering*. https://doi.org/10.1007/s11081-024-09887-3

Kannan, S. K., & Diwekar, U. (2024). An enhanced particle swarm Optimization (PSO) algorithm employing Quasi-Random numbers. *Algorithms*, *17*(5), 195. https://doi.org/10.3390/a17050195

Karaboga, D. (2010). Artificial bee colony algorithm. *Scholarpedia Journal*, *5*(3), 6915. https://doi.org/10.4249/scholarpedia.6915

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & Talbi, E. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, *296*(2), 393–422. https://doi.org/10.1016/j.ejor.2021.04.032

Kumeshan, R., & Saha, A. K. (2022). A review of swarm-based metaheuristic optimization techniques and their application to doubly fed induction generator. *Heliyon*, *8*(10), e10956. https://doi.org/10.1016/j.heliyon.2022.e10956

Lilhore, U., Simaiya, S., Maheshwari, S., Manhar, A., & Kumar, S. (2020). Cloud performance evaluation: Hybrid load balancing model based on modified particle swarm optimization and improved metaheuristic firefly algorithms. *International Journal of Advanced Science and Technology, 29*(5), 12315-12331. Retrieved from https://www.researchgate.net/publication/344694934

Liu, H., Zhang, X.-W., & Tu, L.-P. (2020). A modified particle swarm optimization using adaptive strategy. *Expert Systems with Applications, 152*, 113353. https://doi.org/10.1016/j.eswa.2020.113353

Lu, J., Zhang, J., & Sheng, J. (2022). Enhanced multi-swarm cooperative particle swarm optimizer. *Swarm and Evolutionary Computation*, *69*, 100989. https://doi.org/10.1016/j.swevo.2021.100989

Mataifa, H., Krishnamurthy, S., & Kriger, C. (2023). Comparative analysis of the particle swarm optimization and Primal-Dual Interior-Point algorithms for transmission system Volt/VAR optimization in rectangular voltage coordinates. *Mathematics*, *11*(19), 4093. https://doi.org/10.3390/math11194093

Molina-Pérez, D., Mezura-Montes, E., Portilla-Flores, E. A., Vega-Alvarado, E., & Calva-Yañez, B. (2024). A differential evolution algorithm for solving mixed-integer nonlinear programming problems. *Swarm and Evolutionary Computation, 84*, 101427. https://doi.org/10.1016/j.swevo.2023.101427

Nagra, A. A., Han, F., Ling, Q. H., Abubaker, M., Ahmad, F., Mehta, S., & Apasiba, A. T. (2019). Hybrid self-inertia weight adaptive particle swarm optimisation with local search using C4.5 decision tree classifier for feature selection problems. *Connection Science*, *32*(1), 16–36. https://doi.org/10.1080/09540091.2019.1609419

Nasir, M., Sadollah, A., Yoon, J. H., & Geem, Z. W. (2020). Comparative Study of Harmony Search Algorithm and its Applications in China, Japan and Korea. *Applied Sciences*, *10*(11),

3970. https://doi.org/10.3390/app10113970

Nassef, A. M., Abdelkareem, M. A., Maghrabie, H. M., & Baroutaji, A. (2023). Review of Metaheuristic Optimization Algorithms for Power Systems Problems. *Sustainability*, *15*(12), 9434. https://doi.org/10.3390/su15129434

Perifanis, N., & Kitsios, F. (2023). Investigating the Influence of Artificial intelligence on business Value in the digital Era of Strategy: a literature review. *Information*, *14*(2), 85. https://doi.org/10.3390/info14020085

Priyadarshini, I. (2024). Dendritic Growth Optimization: A novel Nature-Inspired algorithm for Real-World optimization problems. *Biomimetics*, *9*(3), 130. https://doi.org/10.3390/biomimetics9030130

Qiao, J., Li, S., Liu, M., Yang, Z., Chen, J., Liu, P., Li, H., & Ma, C. (2023). A modified particle swarm optimization algorithm for a vehicle scheduling problem with soft time windows. *Scientific Reports*, 13, 18351. https://doi.org/10.1038/s41598-023-45543-z

Qiao, J., Wang, G., Yang, Z., Luo, X., Chen, J., Li, K., & Liu, P. (2024). A hybrid particle swarm optimization algorithm for solving engineering problem. *Scientific Reports*, *14*(1). https://doi.org/10.1038/s41598-024-59034-2

Qin, F., Zain, A. M., & Zhou, K. (2022). Harmony search algorithm and related variants: A systematic review. *Swarm and Evolutionary Computation*, *74*, 101126. https://doi.org/10.1016/j.swevo.2022.101126

Qiu, M., Housh, M., & Ostfeld, A. (2020). A Two-Stage LP-NLP methodology for the Least-Cost design and operation of water distribution systems. *Water*, *12*(5), 1364. https://doi.org/10.3390/w12051364

Qiu, Y., Yang, X., & Chen, S. (2024). An improved gray wolf optimization algorithm solving to functional optimization and engineering design problems. *Scientific Reports*, *14*(1). https://doi.org/10.1038/s41598-024-64526-2

Sajjad, F., Rashid, M., Zafar, A., Zafar, K., Fida, B., Arshad, A., Riaz, S., Dutta, A. K., & Rodrigues, J. J. P. C. (2023). An efficient hybrid approach for optimization using simulated annealing and grasshopper algorithm for IoT applications. *Discover Internet of Things*, *3*(1). https://doi.org/10.1007/s43926-023-00036-3

Shami, T. M., Mirjalili, S., Al-Eryani, Y., Daoudi, K., Izadi, S., & Abualigah, L. (2023). Velocity pausing particle swarm optimization: a novel variant for global optimization. *Neural Computing & Applications*. https://doi.org/10.1007/s00521-022-08179-0

Sharma, S., Mohler, J., Mahajan, S. D., Schwartz, S. A., Bruggemann, L., & Aalinkeel, R. (2023). Microbial Biofilm: a review on formation, infection, antibiotic resistance, control measures, and innovative treatment. *Microorganisms*, *11*(6), 1614. https://doi.org/10.3390/microorganisms11061614

Tian, D., & Shi, Z. (2018). MPSO: Modified particle swarm optimization and its applications. *Swarm and Evolutionary Computation*, *41*, 49–68. https://doi.org/10.1016/j.swevo.2018.01.011

DOI: 10.56892/bima.v9i1A.1241

Tomar, V., Bansal, M., & Singh, P. (2024). Metaheuristic Algorithms for Optimization: A Brief Review. *Engineering Proceedings*. https://doi.org/10.3390/engproc2023059238

Wang, Z., Ding, H., Wang, J., Hou, P., Li, A., Yang, Z., & Hu, X. (2022). Adaptive guided salp swarm algorithm with velocity clamping mechanism for solving optimization problems. *Journal of Computational Design and Engineering*, *9*(6), 2196–2234. https://doi.org/10.1093/jcde/qwac094

Xia, Y., Feng, Z., Niu, W., Qin, H., Jiang, Z., & Zhou, J. (2018). Simplex quantum-behaved particle swarm optimization algorithm with application to ecological operation of cascade hydropower reservoirs. *Applied Soft Computing*, *84*, 105715. https://doi.org/10.1016/j.asoc.2019.105715

Zhou, X., Ma, H., Gu, J., Chen, H., & Deng, W. (2022). Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism. *Engineering Applications of Artificial Intelligence*, *114*, 105139. https://doi.org/10.1016/j.engappai.2022.105139