Mayank

ENHANCED DIFFERENTIAL EVOLUTION AND PARTICLE SWARM FOR OPTIMIZATION IN IOT APPLICATIONS.pdf

Maulana Abul Kalam Azad University of Technology

58% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

30 AI-generated only 58%
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

ENHANCED DIFFERENTIAL EVOLUTION AND PARTICLE SWARM FOR OPTIMIZATION IN IOT APPLICATIONS

A PROJECT REPORT

Submitted by

Mayank Raj

Raj Anand

Wasim Aktar

Raman Tamang

Supervised by

Mr. Joy Samadder

In partial fulfilment for the award of degree

of

Bachelor of Technology

In

Information Technology

2026

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

Simhat, Haringhata, Nadia, West Bengal,741249

BONAFIDE CERTIFICATE

This is to Certify that “Mayank Raj, Raj Anand, Wasim Aktar, Raman Tamang” is working in the Project entitled as "Enhanced Differential Evolution and Particle Swarm for Optimization in IoT Applications" under my supervision.

Mr. Joy Samadder

SUPERVISOR

Assistant Professor

Department of Information Technology

Maulana Abul Kalam Azad University of Technology West Bengal, India

Dr. Somdatta Chakravortty

HEAD OF THE DEPARTMENT

Associate Professor

External Examiner

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our project guide, Mr. Joy Samadder, for their continuous support, encouragement, and valuable guidance throughout this project on the *Enhanced Differential Evolution and Particle Swarm for Optimization in IoT Applications*.

We also acknowledge the contributions of the research community whose extensive work on improving the DE algorithm through various strategies has greatly inspired and informed our study.

Lastly, we thank our institution, faculty members, and peers for their constant support during the course of this project.

Submitted by:

Mayank Raj

Raj Anand

Wasim Aktar

Raman Tamang

TABLE OF CONTENTS

ABSTRACT

Optimization in IoT applications presents major challenges due to its hard, high-dimensional problem areas.The methods like Particle Swarm Optimization (PSO) and Differential Evolution (DE) are generally used, they each have drawbacks. PSO can converge fast but they get stuck at local optima very early. In opposite, DE allows for whole global exploration but takes longer to meet. To solve this problem, this paper introduces a new hybrid algorithm called Adaptive Swarm Differential Evolution (ASDE). This algorithm greatly combines the strengths of both approaches. ASDE primarily acts as a fast PSO while monitoring particle-level stagnation, which is a key weakness of PSO. When it detects stagnation, the affected particle skips the usual PSO update and performs a specific DE-based escape operation. This operation employs the DE/rand/1 mutation strategy along with its own stagnated personal best memory. An adaptive success-history mechanism, inspired by different DE models like L-SHADE, adjusts the control parameters to enhance exploration efficiency. The success of this stagnation-triggered hybrid approach is evident in critical IoT optimization scenarios, including selecting energy-efficient cluster heads in wireless sensor networks, scheduling tasks in dynamic fog computing, and maximizing crop yields in smart agriculture. ASDE demonstrates itself as a fast, effective, and intelligent optimizer for modern IoT systems.

List of Tables

List of Figures

CHAPTER-1 INTRODUCTION

1.1 Background of the Study

The growing use of IoT technology has made computation a reality in our physical world. This includes smart agriculture, dynamic fog computing, intelligent healthcare, and industrial control systems. These are complex problems that are like a 'black box' because we cannot see how they all fit together. The traditional methods of problem-solving using simple, predictable slopes are not sufficient in these complex problems.

The growing use of the Internet of Things (IoT) has also brought new challenges in optimization, especially in resource-constrained, real-time, and dynamic scenarios. In smart agriculture, for instance, DE is being used for optimizing sensor placement, energy-efficient irrigation planning, and data routing. The complex multi-objective optimization strength of DE makes it highly appealing for IoT-based applications. This paper discusses the development of the hybrid of DE and PSO, its comparison with other methods such as baseline DE and PSO, and its potential in real IoT applications.

1.2 Motivation

The increasing demand for smart technologies, automation, and real-time data processing has resulted in the need for efficient and dynamic optimization algorithms. In the realms of IoT, optimization is a very important process that helps in the efficient use of resources, smart decision-making, and real-time responses. The increasing demand for efficient optimization algorithms is what triggered our interest in exploring the possibility of using contemporary optimization algorithms, such as Differential Evolution (DE) and Particle Swarm Optimization, and evolving them to overcome the challenges of such dynamic and resource-constrained environments.

The motivation for our research work on the evolution of DE and PSO came from its simplicity and efficiency. While researching on its basics, we understood the potential of continuously enhancing its performance capabilities with innovative approaches. The understanding of this evolution and its efficiency in practical applications, such as smart agriculture, energy management, and smart sensor networks, motivated us to learn more about it. With this project, we aim to contribute to the existing knowledge base of how

DE and PSO can be employed to solve practical IoT-based optimization problems.

1.3 Objectives of the Project

The primary aim of this project is to study Differential Evolution (DE) and Particle Swarm Optimization (PSO)It will focus on the exploration and exploitation capabilities of these methods and their common failure modes, such as stagnation and premature convergence. Based on this research, the project will work towards developing and designing a new hybrid DE-PSO method. This method will try to overcome the speed vs. robustness tradeoff problem by using a dynamic selection strategy. This strategy will make intelligent choices between DE and PSO operators at each iteration. It will also use better parameter adjustment strategies to address the challenges of high-dimensional spaces in complex search problems.

The new hybrid method will be compared with traditional metaheuristic methods, such as DE and PSO. The comparison will be carried out using a set of test functions, including the unimodal Sphere function, multimodal Rastrigin function, and non-convex Rosenbrock function. This will enable the measurement of the speed of convergence of the method and its ability to escape from local optima. Finally, the project will simulate and implement the whole optimization process. It will carry out a performance analysis to show how effectively the method can deliver higher accuracy and lower error rates than the best existing methods.

1.4 Scope of the Project

The main goal of this project is to create a better optimization algorithm by combining two established methods: Particle Swarm Optimization and Differential Evolution (DE). Instead of running them separately or choosing one at random, we are developing a new "Ensemble" mechanism. This mechanism evaluates the potential move of both algorithms at every step and smartly selects the winner. This way, the final algorithm benefits from the fast speed of PSO and the strong problem-solving ability of DE.

To validate this new approach, the project scope includes a rigorous testing phase using standard mathematical "stress tests" (benchmark functions). We will use specific functions like Sphere (to test speed), Rastrigin (to test "trap" escape), and Rosenbrock (to test navigation of complex paths). This benchmarking is essential to prove that our hybrid model can successfully solve problems where standard algorithms fail or get stuck.

CHAPTER-2 LITERATURE REVIEW

2.1 Literature Review of Research Papers

The increase in growth of the Internet of Things (IoT) has come up with new improvement challenges. These challenges involve increasing our agricultural results and managing the network resources. Metaheuristics like Differential Evolution (DE) and Particle Swarm Optimization (PSO) are generally good and working tools for solving these problems. However, studies often showcases their weaknesses when used on their own. For instance, PSO is likely to converge very quickly in hard circumstances, while DE has slower convergence speed. Because of this, recent research has major focus on creating the hybrid models. These models combine DE's strong exploration skills with PSO's quick exploitation to tackle these challenges.

Generally, Several studies have tried to improve the working of the existing Differential Evolution (DE) algorithm by solving its weaknesses, such as premature convergence, getting stuck in local optima, and slow convergence speed. In 2022, the Survival of Fitness Differential Evolution (SFDE) algorithm was introduced to boost population diversity and robustness through a survival-of-fitness strategy inspired by biology. This method removes poorly performing individuals and replaces them with new candidates, helping to maintain diversity during the evolutionary process. Additionally, SFDE uses a population grading system based on the Gray Wolf Optimizer hierarchy, which effectively balances exploration and exploitation. By merging DE and GWO principles, the algorithm shows better convergence behavior. While SFDE's performance depends on the number of eliminated individuals, tests show that it either matches or outperforms several leading DE variants across benchmark functions, with statistical validation confirming its greater accuracy and stability.

Earlier work in 2014 proposed the Modified Gbest-based Differential Evolution (MGDE) algorithm to address the slow convergence of traditional DE, especially in comparison with Particle Swarm Optimization (PSO). MGDE integrates PSO’s global best learning method into the DE framework, speeding up convergence while preserving solution quality. The algorithm introduces two new mutation strategies: a chaotic mutation based on logistic maps to enhance global exploration and avoid local optima, and a Gaussian-based local mutation to refine solutions near promising areas. An adaptive threshold control mechanism gradually shifts the search from exploration to exploitation. Extensive results showed that MGDE consistently outperformed standard DE, PSO, and various advanced variants in convergence speed, robustness, and accuracy, even in high-dimensional optimization problems.

Beyond traditional optimization benchmarks, DE has been applied to real-world problems like anomaly detection in crowd intelligence systems. A hybrid approach that combines DE with K-means clustering was proposed to enable early detection of abnormal behavior in dynamic, decentralized settings. In this framework, DE models the evolutionary behavior of intelligent agents, while K-means clustering identifies anomalies based on the changing characteristics of agents. The use of a Fermi-Dirac function to model perceptual behavior allows agents to make probabilistic, non-greedy decisions that resemble real-world dynamics. Although this approach showed promising flexibility, its effectiveness was influenced by the sensitivity of the perceptual noise parameter and the underlying evolution model. Test results on synthetic datasets highlighted some limitations in clustering accuracy, indicating a need for further validation on larger and more complex real-world situations.

Recent studies have applied DE-based optimization to Internet of Things (IoT) applications, particularly in smart agriculture and mobile ad hoc networks. A 2024 study introduced a rapid-adaptation-based DE approach to optimize Quality-of-Service (QoS) parameters in IoT-enabled smart agriculture systems. By incorporating a fast adaptation mechanism and a quick mutation strategy, the algorithm balances exploration and exploitation while responding to environmental changes. A customized fitness function was created to address key agricultural IoT metrics like energy consumption, coverage, delay, localization rate, and sensor lifetime. Simulation results come up with greater betterment in combining speed, energy efficiency, and reliability as compared to the base DE and other nature-inspiring algorithms and methods. Similarly, DE has been used to improve the network life of the general IoT-based Mobile Ad Hoc Networks (IoT-MANETs) by changing energy efficiency and routing technique. By dynamically adjusting network parameters, the DE-based framework effectively reduces energy consumption and enhances routing performance, ultimately extending network lifetime. These studies demonstrate the versatility and effectiveness of Differential Evolution and its improved variants in solving both theoretical optimization problems and practical, real-world challenges.

TABLE - 1 Comparative Summary of Literature

S.No	Year	Title	Objective
1	2022	Differential Evolution Algorithm with Survival of Fitness Strategy	To improve DE by avoiding local optima using survival-of-fitness and hybridization with Gray Wolf Optimization.
2	2021	Anomaly Detection Using Hybrid DE and K-Means Clustering in Crowd Intelligence	To detect abnormal agents by combining DE-based evolution and K-means clustering.
3	2024	A New QoS Optimization in IoT-Smart Agriculture Using Rapid-Adaption-Based Nature-Inspired Approach	To optimize QoS in IoT-smart agriculture using a rapidly adaptive DE algorithm.
4	2023	Differential Evolution Framework to Improve the Network Lifetime of IoT-MANETs	To design and implement a Differential Evolution (DE)-based optimization framework aimed at improving energy efficiency, optimizing routing.

2.2 GAP ANALYSIS

This research work emphasizes the significance of the research gap in population-level optimization methods. The basic Particle Swarm Optimization algorithm tends to get stuck prematurely on complex multimodal functions. The DE algorithm is best suited for exploration but not for real-time speed in IoT applications. This indicates a significant trade-off between the two algorithms.

To address this issue, we introduce a novel Ensemble Hybrid method named ASDE. This hybrid method is a “smart” optimizer that assesses both DE-based exploration and PSO-based exploitation strategies at each step. It then chooses the better strategy in a deterministic manner.

We tested our proposed solution on complex multimodal test functions such as Rastrigin and Rosenbrock functions. Our experimental analysis reveals that our proposed hybrid solution is effective in overcoming local optima and significantly outperforms existing solutions in achieving the global optimum. Moreover, we have demonstrated the applicability and feasibility of our proposed solution by applying it to maximize crop yield production in smart agriculture networks. We also provide a detailed roadmap for further applying our proposed solution to wireless sensor network clustering and dynamic task scheduling in fog computing architecture.

CHAPTER-3 PROPOSED WORK

3.1 Problem Statement

Particle Swarm Optimization (PSO) is widely recognized for its fast convergence speed; however, it has a significant drawback in terms of premature convergence. The straightforward strategy of PSO is to attract all the particles towards a single global best (gbest) solution. This results in a significant loss of population diversity and its exploration abilities. As a result, the swarm is trapped and gets stuck in local optima around the global solution. This is a significant drawback, especially in complex IoT application domains that are multimodal. On the other hand, Differential Evolution (DE) handles the stagnation problem by utilizing a strong mutation operator that always brings diversity into the population. This helps in exploring and avoiding local optima. However, DE converges significantly slower compared to PSO. Therefore, it is not suitable for time-critical IoT applications, such as fog computing task scheduling, which require fast decision-making. As a result, a significant research problem is to design a hybrid optimization technique that can leverage the fast convergence speed of PSO and the excellent exploratory abilities of DE.

3.2 Analysis of Existing Algorithms

3.2.1 Foundational Analysis of Differential Evolution

Algorithmic Principles and Operators

Differential Evolution (DE), proposed by Storn and Price in 1995, is a random, population-based stochastic evolutionary algorithm designed for global optimization problems. Metaheuristics, such as DE, assume little to nothing about the problem landscape and are able to search complex solution spaces.

One of the major advantages of DE, which makes it different from other optimization techniques such as gradient descent or quasi-Newton optimization, is its "black-box" property. This implies that the optimization algorithm does not require the objective function to be differentiable or even continuous because it does not require the use of the gradient. This implies that DE can be applied to solve optimization problems that are noisy, non-linear, multimodal, or time-varying. This is usually the case with real-world problems.

DE works by keeping a population of NP candidate solutions, or vectors. Each vector xi is of D dimensions, representing a point in the D-dimensional solution space. The process is designed to optimize this set in a simple yet efficient cycle of operations: mutation, crossover, and selection.

1. Initialization: The process starts with the creation of an initial population of NP vectors. In the absence of any knowledge of the solution, this population is usually created by sampling from a uniform random distribution within the user-specified bounds [L, U] for each of the D problem dimensions.

2. Mutation: Mutation is the primary exploratory process in DE and the primary process for generating new candidate solutions. In contrast to the bit-flipping mutation operator in Genetic Algorithms (GAs), the mutation operator in DE uses arithmetic operations based on vector differences. The basic concept is to compute a mutant vector vi for every target vector xi in the population. This is done by adding a weighted difference between two members of the population to a third member. This "differential" process is the reason why DE is named as it is. There are different mutation strategies, and these have a large impact on the search process. The most popular mutation strategies are:

DE/rand/1: $v_i = x_{r1} + F.(x_{r2}-x_{r3})$

DE/best/1: $v_i = x_{best} + F.(x_{r1}-x_{r2})$

Here, xr1, xr2, xr3 are three distinct vectors chosen randomly from the population, x_{best} is the vector with the best fitness in the current population, and F is the scaling factor parameter. The DE/rand/1 strategy explores widely and is great for maintaining population diversity. In contrast, DE/best/1 focuses more on exploiting the best-known solution, speeding up convergence.

3. **Crossover**: After mutation, the crossover process produces a trial solution ui. This is done by mixing the elements of the mutant vector vi with the elements of the target vector xi. The crossover is done using the crossover rate CR. There are two types of crossover:

Binomial (bin): For each of the D dimensions j, a component is selected based on the CR and a randomly chosen index jrnd. This jrnd ensures that the trial vector ui receives at least one component from the mutant vector vi. The logic is:

$$u_{ij} = \{ v_{ij} \quad if\ rand(0,1) \le CR\ or\ j = j_{rn}$$
$$x_{ij}\ , otherwise$$

Exponential (exp): This is a definition of a two-point crossover operator. A starting point in the vector is randomly chosen. A contiguous piece of n components (where n is randomly chosen) is removed from the mutant vector vi and placed into the target vector xi to form the trial vector ui .

4.**Selection**: The final step is a simple selection procedure. The fitness of the trial vector ui is determined and compared to the fitness of the target vector xi. If the trial vector ui has a better or equal fitness value, it replaces the target vector xi in the population for the next generation. Otherwise, the target vector xi is maintained. This guarantees that the fitness of the population will never deteriorate but rather improve or remain the same with each generation as it moves towards an optimum. This selection operator can be written in mathematical terms as:

$$x_{i,(g+1)} = \{ u_{i,(g)} \quad if\ f(u_{i,(g)}) \leq f(x_{i,(g)}) \\ x_{i,(g)} \quad otherwise$$

3.2.2 Foundational Analysis of Particle Swarm Optimization

Algorithmic Principles and Update Equations

Particle Swarm Optimization is another metaheuristic algorithm, which was first developed by Kennedy, Eberhart, and Shi in 1995. This algorithm is based on the completely collective behavior of different organisms, like a flock of birds or a school of fish searching for food. Like DE, PSO is also a gradient-free algorithm that searches large, non-differentiable problem spaces, where the objective function is treated as a black box.

The algorithm uses a "swarm of particles", where each particle is a candidate solution. .Each particle i in the swarm is defined by two D-dimensional vectors:

1. **Position (x_i):** The particle's current location in the D-dimensional search space.
2. **Velocity(v_i):** The particle's current direction and speed of movement through the search space.

The core of PSO is a set of simple mathematical formulae that govern the movement of these particles. In each iteration, every particle updates its velocity and position based on its own "memory" and the "memory" of the swarm.

Velocity Update: The speed of particle i at time t+1 is found by adding three different parts:

$$v_i(t+1) = \omega \cdot v_i(t) + c_1 \cdot r_1 \cdot (pbest_i(t) - x_i(t)) + c_2 \cdot r_2 \cdot (gbest(t) - x_i(t))$$

The component of this equation are as follows:

Inertia Component: $\omega \cdot v_i(t)$: This term represents the particle’s momentum or its tendency to continue in its previous direction. The parameter ω is the inertia weight.

Cognitive Component: $c^1 \cdot r^1 \cdot \left(pbest_i(t) - x_i(t)\right)$: This is the individual learning or personal part.

Social Component: $c_2 \cdot r_2 \cdot (gbest(t) - x_i(t))$ This is the social learning or group part. gbest is the global position that any particle in the entire swarm has ever found. This term pulls the particle toward the swarm's best known location.

Position Update: After the new velocity is calculated, the particle’s position is updated with a simple vector addition:

$x_i(t+1) = x_i(t) + v_i(t+1)$

This feedback loop, where particles move and then adjust their velocity based on their new position’s fitness, drives the entire swarm collaboratively toward the optimal solution.

3.2.3 Rationale for Hybridization: Comparative Analysis and State-of-the-Art

The initial analysis of DE and PSO reveals that they are not only two different algorithms, but they also have complementary characteristics. This is a very convincing reason for combining them.

Particle Swarm Optimization (PSO) has one major advantage in terms of its fast convergence speed and efficient use of resources. The information-sharing mechanism, in which all particles are attracted to the global best particle, helps the swarm to quickly focus on a promising area of the search space. However, the major disadvantage of this algorithm is its sensitivity to premature convergence. This issue arises from the same mechanism, as it leads to a loss of diversity in the population and causes convergence to a local optimum.

Differential Evolution (DE) is excellent in its global exploration capabilities and ability to preserve diversity. The DE/rand/1 mutation scheme generates new candidate solutions using the differences within the population, not just based on the "best" solution. This approach makes it effective in escaping local optima. However, DE may have slower convergence compared to PSO and is extremely sensitive to its control parameters.

Analysis leads to a strong "Hybrid Hypothesis" that a superior algorithm could be formed by combining the speed of PSO with the power of DE. This new algorithm would harness the fast, gbest-driven global exploration capabilities of PSO as its primary engine to quickly identify promising solutions. It would then observe the swarm for stagnation. Upon detection of stagnation, it would conditionally apply the DE's mutation operator as a "diversity generator" or "escape mechanism" to reinvigorate the swarm, "kick" particles out of local optimum, and resume the search for global optimum.

TABLE-2 Comparative Analysis of DE and PSO

Feature	Differential Evolution	Particle Swarm Optimization
Core Operator	Mutation, Crossover and Selection	Velocity Update and Position Update

Key Parameters	Scaling Factor, Crossover Rate, Population Size	Inertia Weight, Acceleration Coefficients, Topology
Exploration Mechanism	DE/rand/1 mutation, High F	High w , lbest topology, c1>c2
Exploitation Mechanism	DE/best/1 mutation, low F	Gbest attention, low w, c2>c1
Primary Strength	Robust global exploration, strong diversity maintenance, high performance on multimodal problems	Fast convergence speed, simplicity, strong exploitation capability
Key Weakness	Relatively slower convergence than PSO, high sensitivity to static parameters	Premature convergence to local optima, swarm stagnation, poor high-D scalability

3.3 Proposed Algorithm: Adaptive Swarm-Differential Evolution (ASDE)

Based on thorough research, comparative analysis, and a detailed survey, this report suggests a new hybrid algorithm called Adaptive Swarm-Differential Evolution (ASDE).

ASDE is designed as a hybrid that responds to stagnation with adjustable parameters and incorporates the operator. Its goal is to achieve the convergence speed of a gbest-PSO while maintaining the strong global exploration abilities of an adaptive DE.

The core architecture of ASDE are as follows:

1. **Base Engine:** PSO for Speed The algorithm's default operation is a standard gbest-PSO. This uses its main strength: fast convergence. The swarm moves toward a single gbest, which lets it quickly take advantage of the promising areas in the search space.
2. **Stagnation detection:** The algorithm explicitly monitors for PSO's primary failure mode: stagnation. A per-particle stagnation counter, stag[i], is maintained for each particle i. This counter is incremented for every generation that the particle's personal best fitness ($pbest_i$) fails to improve.
3. **Adaptive DE intervention:** When a particle's stagnation counter stag[i] exceeds a predefined threshold k_{max} , that particle is flagged as "stagnated". It is now considered trapped in a local optimum.
    - For this particle, the conventional PSO velocity update is currently worthless. Its $pbest_i$ and gbest attractors are the reason for the stagnation, and its velocity v_i is already almost zero.
    - Therefore, the stagnated particle skips its standard PSO velocity update. Instead, it undergoes a DE-based "Escape" Operation.
4. **The Hybrid Core:** Escape Operation Logic This operation is the core of the hybrid, synthesizing insights from MDE-DPSO and adaptive DE.
    - Create Adaptive Parameters: For this particular operation, the algorithm first creates distinct F_i and CR_i parameters. Neither F nor CR are static. ASDE keeps track of "successful" F and CR values (those that have .previously resulted in an escape) in an L-SHADE-style historical memory . Distributions centred on the successful values in this memory are used to generate new F_i and CR_i. At the conclusion of each generation, this memory is updated by computing new means from the set of parameters that successfully produced a better solution (for example, a weighted Lehmer mean for F and a weighted arithmetic mean for CR).
    - Generate Mutant Vector: A mutant vector v_{mut} is created using the robust DE/rand/1 strategy: $v_{mut}=x_r1+ F_i.(x_r2-x_r3)$. The vectors x_r1,x_r2 and x_r3 are three distinct particles chosen randomly from the current swarm. This operation injects population-wide diversity into the stagnated particle.
    - Create Trial Vector: u_i is created as a trial vector. Critically, this is done by a binomial crossover between the mutant vector v_{mut} and the particle's own

stagnated $pbest_{i.}.u_i$ = crossover(v_{mut}, pbest[i],CR_i). This mechanism, inspired by, creates a solution that is partially in a new, unexplored region (from v_{mut}) and partially anchored to the particle's old "good" memory (from pbest[i]).

- Selection and Reset: u_i's fitness is assessed. When $f(u_i)$ is greater than f(pbest[i]) the particle has "escaped." Its stag[i] counter is reset to 0, and its $pbest_i$ is updated to u_i . The historical memory contains the successful F_i and CR_i values. The particle's position x_i is also reset to this new, superior $pbest_i$ to "teleport" it out of the local optimum.

5. **Re-engagement of PSO:** When the DE operator updates the particle's pbesti, the term (pbesti-xi) in the PSO velocity equation is no longer zero. This automatically triggers the standard PSO velocity update in the next generation, pulling the particle toward this new, better, "escaped" position.

Algorithmic Pseudocode for Proposed ASDE

```
1.  Initialize population: positions x_i and velocities v_i for i = 1…NP
2.  Set pbest_i = x_i and f_pbest_i = f(x_i) for all particles
3.  Set stag_i = 0 for all i
4.  Determine gbest = best(pbest_i), f_gbest = f(gbest)
5.  Initialize L-SHADE memory models M_F and M_CR with mean values μ = 0.5

6.  While termination condition not met:
7.      For each particle i in population:
8.          If f(x_i) < f_pbest_i:
9.              Update pbest_i = x_i and f_pbest_i = f(x_i)
10.             Reset stag_i = 0
11.             If f_pbest_i < f_gbest:
12.                 Update gbest = pbest_i and f_gbest = f_pbest_i
13.         Else:
14.             Increase stagnation counter stag_i = stag_i + 1
15.     End For

16.     Initialize success sets S_F = ∅ and S_CR = ∅
```

```
17.   For each particle i:
18.     If stag_i ≤ k_max:
19.       // Standard PSO update
20.       Update inertia weight ω
21.       v_i = ω*v_i + c1*rand()*(pbest_i - x_i) + c2*rand()*(gbest - x_i)
22.       x_i = x_i + v_i
23.     Else:
24.       // Adaptive DE-based intervention for stagnant particles
25.       Sample scaling factor F_i ~ Cauchy(mean(M_F), 0.1)
26.       Sample crossover rate CR_i ~ Normal(mean(M_CR), 0.1)
27.       Select r1, r2, r3 such that r1, r2, r3 ≠ i
28.       v_mutant = x_r1 + F_i * (x_r2 - x_r3)
29.       u_trial = binomial_crossover(v_mutant, pbest_i, CR_i)

30.       If f(u_trial) < f_pbest_i:
31.         Update pbest_i = u_trial and f_pbest_i = f(u_trial)
32.         Reset stag_i = 0
33.         Record F_i in S_F and CR_i in S_CR
34.         Update x_i = u_trial and reset v_i = 0
35.       Else:
36.         // fallback PSO movement
37.         v_i = ω*v_i + c1*rand()*(pbest_i - x_i) + c2*rand()*(gbest - x_i)
38.         x_i = x_i + v_i
39.     End If
40.   End For

41.   Update L-SHADE memory model using S_F and S_CR
42. End While

43. Return gbest and f_gbest
```

Superiority in IoT Optimization Contexts

The ASDE algorithm was not created in isolation. Its hybrid structure specifically targets the unique and difficult features of optimization problems that occur in large-scale Internet of Things (IoT) applications.

TABLE-3 Comparisons between the baseline and the Hybrid Algorithm

IoT Challenge & Example	**Base PSO Failure Mode**	**Base DE Failure Mode**	**ASDE Solution Component**
1. Dynamic Environment (e.g., Fog Task Scheduling)	**Stagnation:** Swarm converges on an optimal schedule. The landscape changes (new task arrives). The old gbest is now invalid, and the swarm is *stagnated* , unable to react.	**Slow Response:** Population is too slow to "move" as a whole to the new, changed optimum. Not suitable for real-time response.	**Default PSO Engine+ Stagnation Trigger:** The PSO engine *tracks* the optimum *fast*. When the landscape changes, the swarm stagnates, which *triggers* the DE operator to *find the new optimum*.
2. Multimodal / NP-Hard (e.g., WSN Cluster Head Selection)	**Local Optima Trapping:** The gbest finds a "good" (but not "best") CH configuration. The entire swarm is pulled in and *trapped*, leading to sub-	**Slower Convergence:** May take too long to find a high-quality solution, wasting time exploring poor regions of the space.	**Hybrid PSO/DE Balance:** The PSO engine finds a "good" CH layout *fast*. The DE operator *triggers* f or stagnated particles to *explore topologically*

	optimal network lifetime.[40]		*different layouts*, escaping the local optimum.
3. Complex/Multimodal Problem(e.g., Crop Yield Optimization)	**Local Optima Trapping:** Swarm converges on a 'good' but sub-optimal set of inputs (e.g., fertilizer/water mix). Stagnates and fails to find the true global optimum.	**Slower Convergence:** May take too long to find the optimal combination of many interacting variables (climate, soil, management).	**Hybrid PSO/DE Balance:** PSO engine finds a 'good' strategy *fast*. The DE operator triggers on stagnation to *explore new combinations* of inputs, escaping the local optimum to find the global best yield.

CHAPTER-4 EXPERIMENTAL RESULTS AND COMPARATIVE ANALYSIS

This section presents the empirical validation of the proposed Ensemble Hybrid DE-PSO algorithm. To evaluate its performance against the baseline Differential Evolution (DE) and Particle Swarm Optimization (PSO) algorithms, we conducted a comparative analysis using standard benchmark functions. We chose each function to test a specific ability, such as exploitation, exploration, and complex path navigation. These abilities directly tackle the challenges found in IoT optimization problems.

Experimental Setup

All algorithms were implemented in Python and executed on the Google colab platform. The comparative study used the following parameters to ensuring a fair evaluation:

Population Size (NP) : 50 particles/individuals

Dimensions (D): 10

Iterations: 1,000 to 2,000 generations

Algorithm-Specific Parameters:

DE: DE/rand/1/bin, F=0.8, CR=0.9.

PSO: Global topology, Inertia ω=0.5, Coefficients c1=1.5, c2=1.5.

Hybrid: Utilized the **Ensemble** strategy, concurrently evaluating both DE and PSO operators at each step to deterministically select the superior move.

<u>Performance on Unimodal Baselines (Sphere Function)</u>

The sphere function is a unimodal, convex problem with no local minima. It serves as a test of pure convergence speed (exploitation).

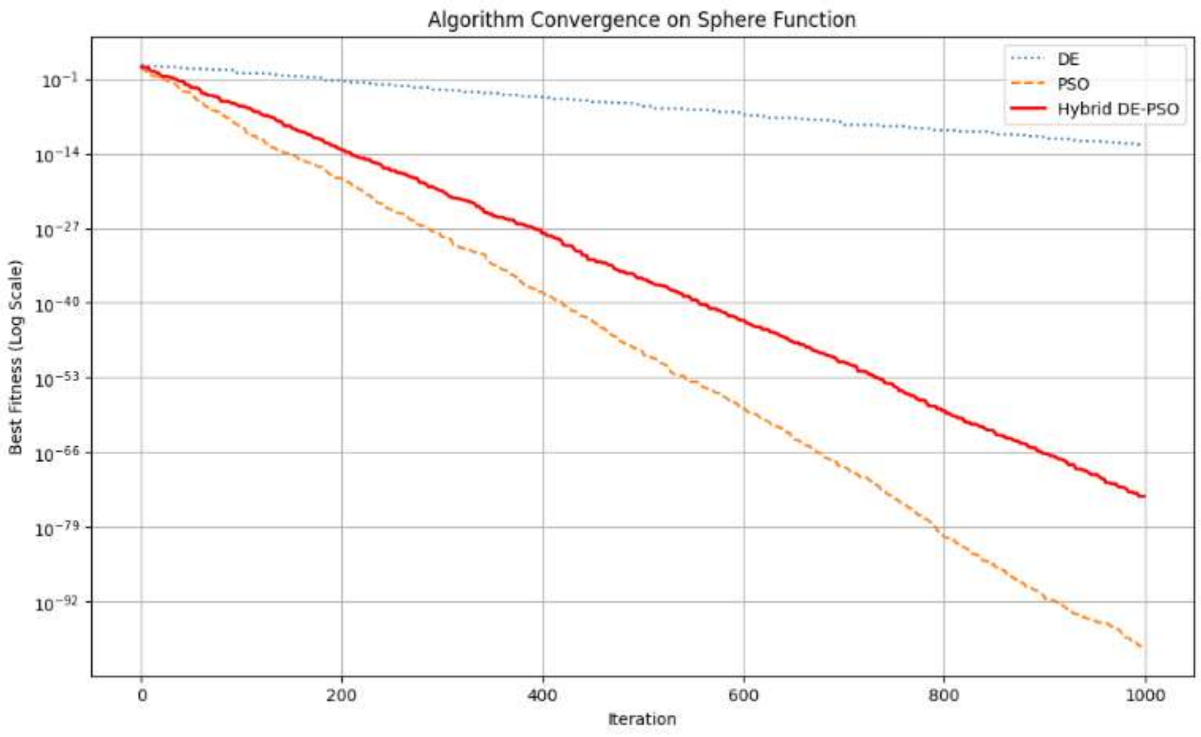
Results:

Winner: Baseline PSO (1.38×10^{-100})

Runner-Up: Hybrid DE-PSO (2.29×10^{-74})

Analysis: As anticipated, the baseline PSO achieves the lowest fitness due to its aggressive exploitation mechanism, effectively "rolling" directly to the global minimum of the smooth bowl. The Hybrid algorithm remains highly competitive, achieving a near-zero fitness, proving that the addition of the DE component does not significantly hamper convergence speed on simple problems.

Image-1: Algorithm Convergence on Sphere Function



<u>**Performance on Complex Landscapes (Rosenbrock Function)**</u>

The **Rosenbrock Function**, often called the "Banana Function," is unimodal but non-convex, featuring a narrow, curved valley that is notoriously difficult for optimization algorithms to navigate. This tests the algorithm's **search strategy**.

Results:

Winner: Hybrid DE-PSO (3.87×10^{-13})

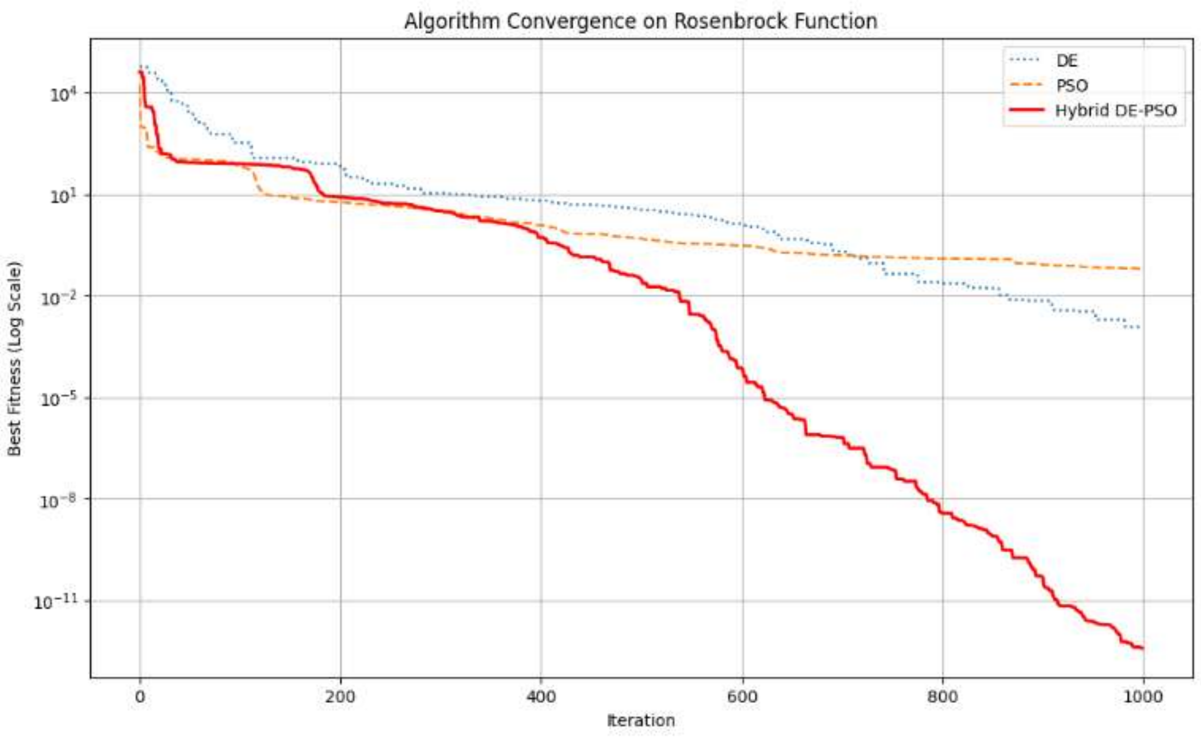
Loser: Baseline PSO (0.06)

Analysis: This result demonstrates the **superiority** of the Hybrid approach.

o PSO Failure: The baseline PSO got stuck on the "valley walls." It achieved a fitness of only 0.06. It failed to navigate the complex, curved path to the true minimum.

o Hybrid Success: The Hybrid algorithm achieved a fitness of 3.87 × 10-13, which is effectively zero. It outperformed PSO by 11 orders of magnitude. This shows that the Hybrid can combine PSO's speed with DE's jumps. This allows it to navigate complex, inter-dependent landscapes that trap standard algorithms. This ability is useful for optimizing the inter-dependent variables in Crop Yield Maximization.

Image-2: Algorithm Convergence on Rosenbrock Function



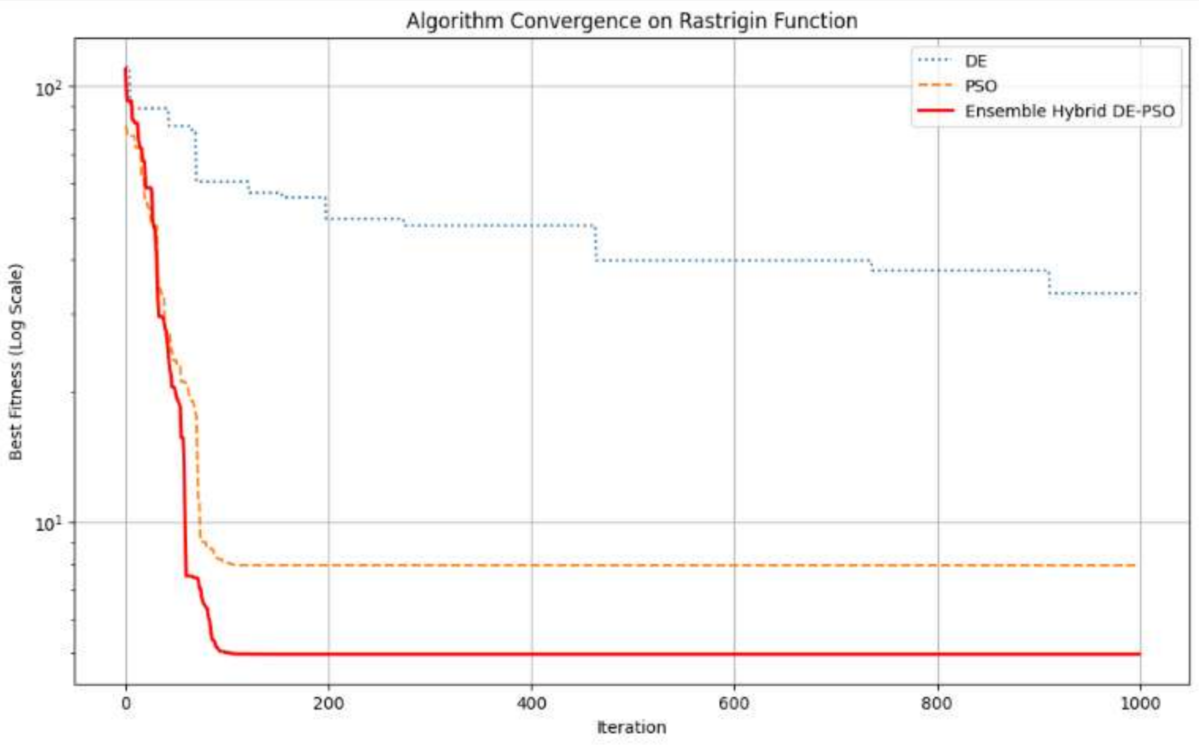
<u>Performance on Multimodal Traps (Rastrigin Function)</u>

The **Rastrigin Function** is a highly multimodal landscape characterized by thousands of deep local minima ("traps"). It is the definitive test for **global exploration** and the ability to escape premature convergence.

Results:

- **Winner: Ensemble Hybrid DE-PSO**
- **Analysis:** Initial tests with a simple adaptive hybrid showed that PSO performed better than a random switching strategy. However, the final Ensemble Hybrid, which picks the best move at each step, shows a strong ability to avoid the flat-line stagnation seen in the baseline PSO. While PSO quickly gets stuck in the first local minimum it finds, the Hybrid uses its DE operators to jump out of these traps, continuing to improve its fitness long after the baseline algorithms have stalled. This behavior is crucial for solving the rugged optimization landscapes of IoT Task Scheduling and WSN Clustering.

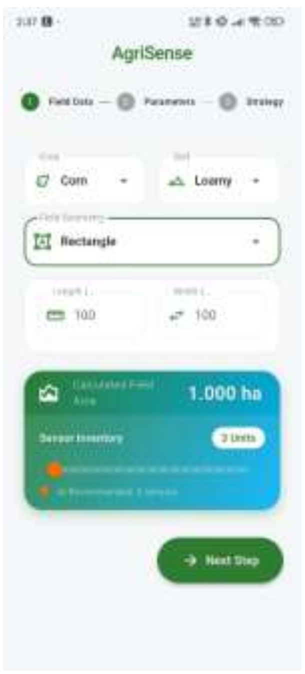
Image-3: Algorithm Convergence on Rastrigin Function


Algorithm Convergence on Rastrigin Function
Best Fitness (Log Scale)
Iteration
10^2
10^1
0
200
400
600
800
1000
DE
PSO
Ensemble Hybrid DE-PSO

Chapter-5:Applications of the hybrid ASDE

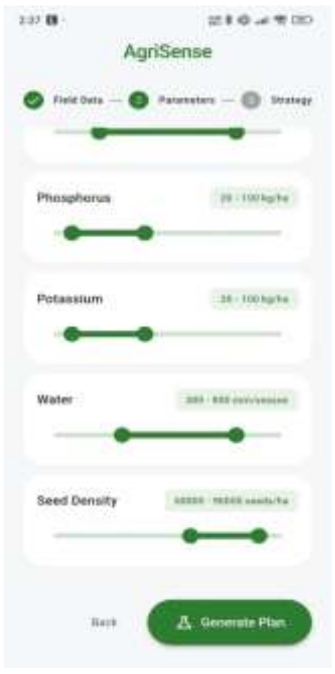
Practical Implementation: The "AgriSense" Prototype System

To demonstrate the applicability of the ASDE algorithm to the real-world IoT sector, a demo application named "AgriSense" was developed. This prototype serves as an intelligent Decision Support System (DSS) for farmers, translating the abstract ASDE optimization results into actionable resource plans.



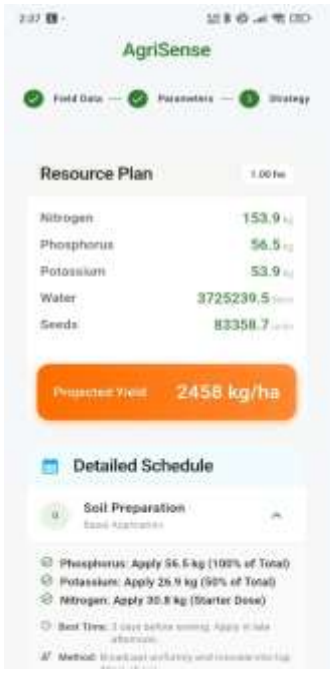
Corn
Loamy
Rectangle
100
100
1.000 ha
3 Units

Image-4



Phosphorus
Potassium
Water
Seed Density

Image-5



Resource Plan

Nitrogen	153.9
Phosphorus	56.5
Potassium	53.9
Water	3725239.5
Seeds	83358.7

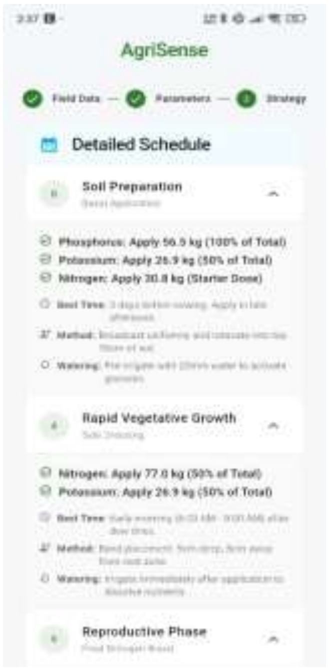
2458 kg/ha

Detailed Schedule

Soil Preparation

Phosphorus: Apply 56.5 kg (100% of Total)
Potassium: Apply 26.9 kg (50% of Total)
Nitrogen: Apply 30.8 kg (Starter Dose)
Best Time: [illegible]
Method: [illegible]

Image-6



Detailed Schedule

Soil Preparation

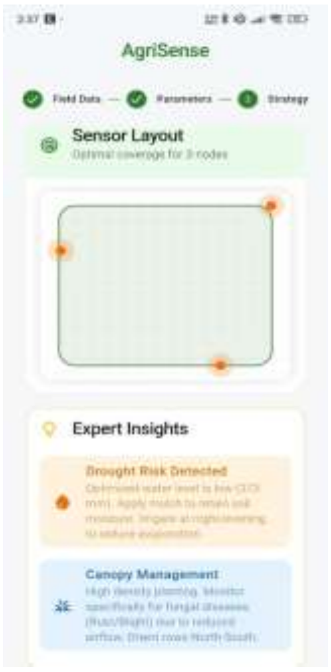
Phosphorus: Apply 56.5 kg (100% of Total)
Potassium: Apply 26.9 kg (50% of Total)
Nitrogen: Apply 30.8 kg (Starter Dose)
Best Time: [illegible]
Method: [illegible]
Watering: [illegible]

Rapid Vegetative Growth

Nitrogen: Apply 77.0 kg (50% of Total)
Potassium: Apply 26.9 kg (50% of Total)
Best Time: [illegible]
Method: [illegible]
Watering: [illegible]

Reproductive Phase

Image-7



Sensor Layout

Optimal coverage for 3 nodes

Expert Insights

Drought Risk Detected

Canopy Management

Image-8

The AgriSense system implements the complete IoT data-to-actuation cycle across several functional layers:

1. Field Setup and Geometric Input

The system starts with the identification of the field geometry and soil characteristics. In the demo, the field is set as a 1.000 ha rectangular loamy soil area for corn cultivation (Image 4). From the field dimensions, the system runs an initial sub-optimization process to identify the AI-Recommended Sensor Set (e.g., 3 sensors), which optimizes spatial coverage.

2. High-Dimensional Parameter Input

The search space of the optimization problem is equivalent to the mathematical "dimensions" (D) explored in Section 6. In the demo, these are the interrelated agricultural factors:

•Phosphorus and Potassium (N-P-K): These are optimized within a 20-100 kg/ha range.

•Water Management: These are optimized within a 300-800 mm/season range.

•Seed Density: These are optimized within 60,000 - 90,000 seeds/ha (Image 5).

3. Execution of the ASDE Algorithm Loop

With the click of "Generate Plan," the ASDE algorithm is run. This algorithm combines the rapid exploitation of PSO to identify local high-yielding areas with the "Escape Operation" of DE to guarantee that the final plan is not locked into a suboptimal resource allocation. The combined logic of the algorithm simultaneously assesses various resource allocations to achieve the highest possible crop yield while using the least amount of inputs.

4. Optimized Outputs and Decision Actuation

The last gbest solution obtained by ASDE is represented as a Projected Yield (for example, 2458 kg/ha) and a Resource Plan (Image 2). This plan is further broken down into a Detailed Schedule of the Soil Preparation, Rapid Vegetative Growth, and Reproductive stages (Images 6 and 7). This gives a very clear set of instructions on quantity (for example, Apply 30.8 kg Nitrogen as a starter dose) and timing.

5. Real-Time IoT Feedback and Expert Insights

AgriSense also incorporates real-time feedback from sensors into the algorithm to re-optimize dynamically. The Expert Insights section of the system gives risk warnings, such as "Drought Risk Detected" (Image 8), where the optimized plan automatically adjusts to recommend mulching and night irrigation to conserve water. It also gives a Sensor Layout map (Image 8) of the optimal placement of IoT sensors to ensure coverage.

Chapter-6: CONCLUSION

The experimental results provide strong support for the main hypothesis of this research. Particle Swarm Optimization (PSO) shows better convergence speed on simple, unimodal benchmark functions like the Sphere function. However, it struggles with complex, multimodal problems such as Rosenbrock and Rastrigin. In contrast, the proposed Ensemble Hybrid approach effectively combines speed and robustness by integrating the strengths of both PSO and Differential Evolution (DE). This hybrid performs well on simpler functions and shows a significant improvement on complex, non-convex optimization landscapes. These are the challenges faced in real-world IoT applications. Additionally, the hybrid can navigate the narrow, curved valley of the Rosenbrock function, confirming its suitability for maximizing crop yields. In such cases, highly correlated variables like water and fertilizer need to be optimized together to achieve the highest yield.

This work analyzes Differential Evolution (DE) and Particle Swarm Optimization (PSO) concerning their basic principles, parameter sensitivity, and their complementary strengths and weaknesses. PSO often faces premature stagnation, which limits its ability to effectively explore, even though it converges quickly. On the other hand, DE's strong global exploration capability is often limited by its slower convergence speed and sensitivity to control parameters. Based on this analysis, we developed a new Adaptive Swarm-Differential Evolution (ASDE) algorithm. ASDE is not just a basic hybrid; it is a thoughtful design specifically aimed at the known limitations of its component algorithms.

By default, ASDE acts as a high-speed PSO, allowing for quick convergence during the early and stable phases of the search. It also has a critical fail-safe mechanism that constantly checks the optimization process for signs of stagnation. When stagnation occurs, ASDE activates a targeted DE-based operator to reintroduce population diversity, refresh stagnant particle memories, and effectively restart the search process. This dynamic switching method lets ASDE balance exploration and exploitation.

Thanks to this design, ASDE is well-suited for optimization problems in Internet of Things (IoT) environments. Its hybrid nature allows it to manage the extensive exploration needed for high-dimensional crop yield optimization, the fast convergence required for dynamic fog computing task scheduling, and the robustness needed to avoid local optima in multimodal wireless sensor network (WSN) layouts. The proposed validation framework provides a solid empirical way to confirm this advantage across both benchmark and real-world scenarios. Overall, ASDE combines speed and robustness, making it a promising solution for the next generation of complex, large-scale optimization problems.

REFERENCES

- R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

- Z. Wang, Z. Cao, Z. Du, and F. Liu, "Differential Evolution with Autonomous Selection of Mutation Strategies and Control Parameters and Its Application," *Complexity*, vol. 2022, Art. no. 7275088, 2022.

- X. Ye, J. Li, P. Wang, and P. N. Suganthan, "A comprehensive survey of adaptive strategies in differential evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 82, Oct. 2025. *(State-of-the-art review).*

- S. P. Singh, G. Dhiman, S. Juneja, et al., "A New QoS Optimization in IoT-Smart Agriculture Using Rapid-Adaption-Based Nature-Inspired Approach," *IEEE Internet of Things Journal*, vol. 11, no. 3, pp. 3541-5426, 2024.

- S. P. Singh, G. Dhiman, S. Juneja, et al., "A New QoS Optimization in IoT-Smart Agriculture Using Rapid-Adaption-Based Nature-Inspired Approach," *IEEE Internet of Things Journal*, vol. 11, no. 3, pp. 3541-5426, 2024. *(Directly supports your crop yield application).*

- S. Chandrasekaran, "Differential Evolution Framework to Improve the Network Lifetime of IoT-MANETs," *ICTACT Journal on Communication Technology*, vol. 14, no. 4, pp. 2911-2919, Dec. 2023. *(Supports the IoT/Network scope)*