iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** V **Month of publication:** May 2024

DOI: https://doi.org/10.22214/ijraset.2024.62340

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

Optimal Deployment of Wireless Sensor Nodes using Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO)

Kiruthikeswaran Rajaram
Department of Aerospace Engineering, College of Aeronautics, Kent State University, Kent, OH

Abstract: ***Wireless Sensor Networks (WSNs) are emerging as a significant area of research due to their potential to autonomously monitor physical and environmental conditions. These networks comprise spatially distributed, low-cost sensor nodes with limited transmission range, processing capabilities, storage, and energy resources. The primary function of these networks is to collect data from various nodes and transmit it to a base station for subsequent processing. WSNs present several challenges, including optimal sensor deployment, node localization, base station placement, target node location, energy-aware clustering, and data aggregation. Recently, global researchers have been employing a bio-inspired optimization algorithm, Particle Swarm Optimization (PSO), to enhance the efficiency of WSNs. This report explores the application of the PSO algorithm for optimal sensor deployment in WSNs, contributing to the ongoing efforts to maximize the potential of these networks.***

Index Terms: ***Wireless Sensor Networks (WSNs), Sensor Nodes, Node Localization, Optimal Sensor Deployment, Particle Swarm Optimization (PSO), Ant Colony Optimization, Network Efficiency***

I. INTRODUCTION

In the rapidly evolving world of wireless sensor networks (WSNs), the strategic placement of sensor nodes is paramount. It plays a pivotal role in enhancing network performance and optimizing resource utilization. This report aims to delve into the complexities of optimizing the deployment of wireless sensor nodes, with a particular focus on two advanced algorithms - Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO).

WSNs are characterized by their distributed nature and inherent resource constraints. These characteristics necessitate the development of efficient strategies for node placement. The goal is to ensure comprehensive coverage, maximize energy efficiency, and facilitate robust data collection. In this context, PSO and ACO emerge as promising solutions.

PSO, an algorithm inspired by social behavior and natural phenomena, and ACO, which models the foraging behavior of ants, have been recognized for their effectiveness in addressing complex optimization challenges. These challenges are inherent in the deployment of WSNs. These algorithms iteratively refine node positions by leveraging the collective intelligence of nodes or agents. The objective is to strive toward an optimal solution that balances coverage, connectivity, and energy consumption.

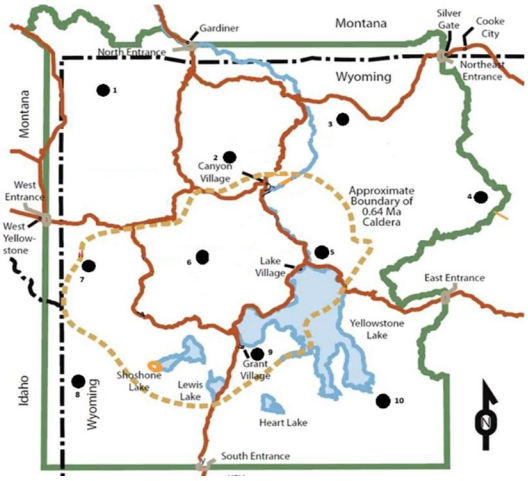


Fig 1: Landscape of Yellowstone National Park

This report provides a comprehensive exploration of PSO and ACO methodologies. It aims to elucidate their theoretical underpinnings, delve into the intricacies of their algorithms, and highlight their practical implications in the context of WSN deployment optimization. The report will also present empirical evaluations and comparative analyses. These will assess the efficacy and suitability of PSO and ACO in various deployment scenarios, shedding light on their strengths, limitations, and potential avenues for further research.

II. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO), conceptualized by Eberhart and Kennedy in 1995, stands as a population-based optimization technique. In this method, particles, representing potential solutions, traverse the problem space by mimicking the movement of the current optimum particles. Each particle maintains its coordinates in the problem space along with the best solution (fitness) it has attained thus far, termed as pbest. Additionally, PSO tracks another crucial metric: the best value achieved by any particle in the swarm, denoted as gbest. Each particle dynamically adjusts its velocity based on its individual flying experience (pbest) and the collective experience of the swarm (gbest), aiming to guide the population towards more favorable solution regions. Operating within a D-dimensional search space, each particle is akin to a volume-less entity. The manipulation of particles follows a set of equations, orchestrating their movement towards optimal solutions.

$$V_i^{q+1} = \omega.V_i^q + C_1 r [X^q{}_{P.Best} - X_i^q] + C_2 r [X^q{}_{G.Best} - X_i^q]$$

$$X_i^{q+1} = X_i^q + V_i^{q+1}$$

The first part of the equation is the previous velocity of the particle. The second is the “cognition” part, representing the exploiting of its own experience, where c_1 is an individual factor. And the third is the “social” pan. representing the shared information and cooperation among the particles, where c_2 is the societal factor.

A. PSO Parameters

For the proposed method the number of particles is taken as 10 and the learning factor $C_1 = C_2 = 2$. An inertia weight factor is linearly reduced as the search proceeds from 0.9 to 0.4. The maximum velocity and maximum iterations are taken as 50 and 300 respectively.

B. Performance Improvement

Particle Swarm Optimization (PSO) is known for its quick convergence and consistent efficiency, regardless of the complexity of the problem space, such as the number of peaks and dimensions. However, it does face certain challenges, including premature convergence and sensitivity to parameter settings, which can lead to local rather than global optimization. As a result, a significant amount of research has been dedicated to overcoming these obstacles. Strategies include adapting parameters, enhancing diversity, and modifying the algorithm to strengthen PSO’s global optimization abilities and facilitate dynamic adaptation.

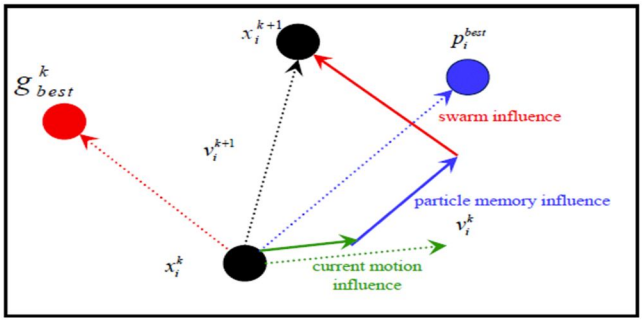


Fig 2: Geometric illustration of particle movement in PSO

C. Optimal WSN Deployment

The Wireless Sensor Network (WSN) conundrum revolves around determining the strategic placement of sensor nodes to achieve desired coverage, connectivity, and energy efficiency while minimizing node count. Inadequate sensor coverage results in unnoticed events, while dense sensor populations lead to congestion and delays. Optimal WSN deployment ensures quality of service, prolonged network lifespan, and cost-effectiveness. Existing PSO solutions for deployment are typically computed centrally, often on a base station, to ascertain sensor positions.

D. Sensor Coverage

A sensor positioned at coordinates (X_1, Y_1) effectively covers another point (X_2, Y_2) if the Euclidean distance between them satisfies the condition: $(X_1-X_2)^2 + (Y_1-Y_2)^2 \leq r^2$, where r represents the sensor's sensing range. The mean value of location points (X_i, Y_i) for i=1, 2, ..., M is denoted as (mx, my). The sensor node serves as the centroid of the location points it covers, with the sensing range r determined by the distance between the sensor node and the farthest location point. Area A is partitioned into R regions, each allocated sensor node by minimizing the Euclidean distance between location points and their respective centroids. Consequently, Area A is covered by R sensor nodes. Formulating the coverage problem as an optimization task involves determining the optimal deployment of R sensors across the set of points P, ensuring comprehensive coverage of every location point.

E. Problem Formulation

The primary aim of this study is to optimize the deployment of sensor nodes in a network. The goal is to minimize the distance between adjacent nodes, thereby maximizing network coverage, while concurrently adhering to all constraints.

The following assumptions underpin this study:

1) All sensor nodes are identical and possess mobility.
2) It is assumed that the deployed sensor nodes can comprehensively cover the sensing fields. Both the sensing coverage and communication coverage of each node are presumed to be circular, devoid of any irregularities.
3) The design variables in this study are the two-dimensional coordinates of the sensor nodes.
4) Each sensor node is assumed to cover an equal area of the sensing field.

These assumptions are commonly made in numerous sensor network applications and form the basis of our analysis in this report.

F. Flow Chart

In the context of wireless sensor networks, the concept of fitness, denoted as (F), is determined by the Euclidean distance between a sensor node and its nearest centroid. The calculation of fitness for each particle involves evaluating its proximity to the optimal solution. The particle within the swarm that exhibits the lowest fitness is identified as the global best particle, indicating its closeness to the optimal solution. The achievement of the swarm is recognized when all particles attain fitness values that are less than or equal to the range of the sensor network.

The Particle Swarm Optimization (PSO) process can be outlined as follows:

1) *Initialization:* The network information and algorithm parameters, including inertia, weight, learning factor, velocity boundary value, and maximum iteration count, are initialized. An array of particles is also initialized with random position and velocity vectors.
2) *Fitness Calculation:* The fitness for each particle at its current position is calculated by determining the distance to its nearest sensor.
3) *Fitness Minimization:* The fitness values are minimized with the ideal goal of reaching zero, indicating that the distance between points of interest and their nearest sensors falls within the sensor's sensing range. If a particle's fitness surpasses the current best, it is designated as the best particle for the subsequent move, and its fitness is updated accordingly.
4) *Position and Velocity Adjustment:* Each particle's position and velocity are adjusted based on the calculated fitness.
5) *Position Evaluation:* The algorithm determines whether the next position of the particle yields an improvement; if so, the particle adopts the new position, otherwise, the algorithm continues with the existing position.
6) *Iterative Process:* The process is repeated iteratively until all particles communicate with each other, collectively maximizing coverage.

This iterative process propels the optimization of sensor node deployment within the network, aiming towards comprehensive coverage and efficient resource utilization. This professional exploration of PSO provides a robust framework for optimizing wireless sensor networks, contributing to the advancement of this dynamic field.

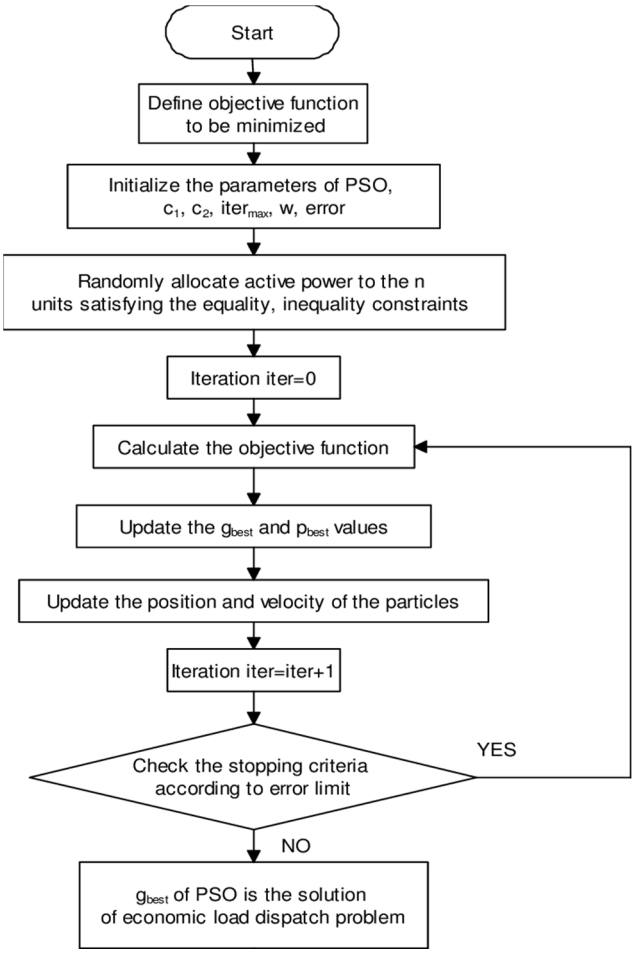


Fig 3: Flowchart of PSO Algorithm

III. IMPLEMENTING OF PSO TO PYTHON

The implementation of Particle Swarm Optimization (PSO) in Python involves coding the PSO algorithm using Python programming language. This entails defining classes or functions to represent particles, initializing their positions and velocities, updating them iteratively based on PSO equations, evaluating fitness functions, and iteratively optimizing the solution until convergence. Python offers various libraries and tools such as NumPy and SciPy that facilitate the implementation of PSO and enable efficient computation, making it a popular choice for implementing optimization algorithms like PSO.

A. Python Code

```
import random
import numpy as np
class Particle:
    def __init__(self, position):
        self.position = position
        self.velocity = np.zeros_like(position)
        self.best_position = position
        self.best_fitness = float('inf')

def PSO(ObjF, Pop_Size, D, MaxT):
    swarm_best_position = None
    swarm_best_fitness = float('inf')
    particles = [Particle(np.random.uniform(-0.5, 0.5, D)) for _ in range(Pop_Size)]
```

```
    for particle in particles:
        fitness = ObjF(particle.position)
        if fitness < swarm_best_fitness:
            swarm_best_fitness = fitness
            swarm_best_position = particle.position
        particle.best_fitness = fitness
        particle.best_position = particle.position

    for itr in range(MaxT):
        for particle in particles:
            w, c1, c2 = 0.8, 1.2, 1.2
            r1, r2 = random.random(), random.random()
            particle.velocity = (w * particle.velocity +
                                 c1 * r1 * (particle.best_position - particle.position) +
                                 c2 * r2 * (swarm_best_position - particle.position))
            particle.position += particle.velocity
            fitness = ObjF(particle.position)
            if fitness < particle.best_fitness:
                particle.best_fitness = fitness
                particle.best_position = particle.position
            if fitness < swarm_best_fitness:
                swarm_best_fitness = fitness
                swarm_best_position = particle.position

    return swarm_best_position, swarm_best_fitness

def F1(x):
    return np.sum(x**2)
def F2(x):
    return np.max(np.abs(x))

Objective_Functions = {'F1': F1, 'F2': F2}

Pop_Size = 100
MaxT = 100
D = 2

for funName, ObjF in Objective_Functions.items():
    best_position, best_fitness = PSO(ObjF, Pop_Size, D, MaxT)
    print(f"Running Function = {funName}")
    print(f"BEST POSITION : {best_position}")
    print(f"BEST COST : {best_fitness}")
    print()
```

IV. RESULT

The initial population is created randomly, and the objective function is calculated. The new sequence generation is based on the initial sequence illustrated in the following example. Consider the following initial sequence P_{ibest} and P_{gbest} as follows:

- Present: 2 6 3 5 4 1
- P_{ibest}: 6 1 2 5 3 4
- P_{gbest}: 5 3 6 4 2 1

Assume $C_1 = C_2 = 2$ and rand () =1. Then Pibest is generated by swapping the individuals of a present sequence.

Present: 2 6 3 5 4 1 Swap :(2, 6)
6 2 3 5 4 1 Swap :(2, 1)
6 1 3 5 4 2 Swap :(3, 2)
6 1 2 5 4 3 Swap :(4, 3)

Here (2,6) (2,1) (3,2) (4,3) are used for getting P_{ibest} from the present sequence. The P_{gbest} is generated by swapping the individual of a present sequence.

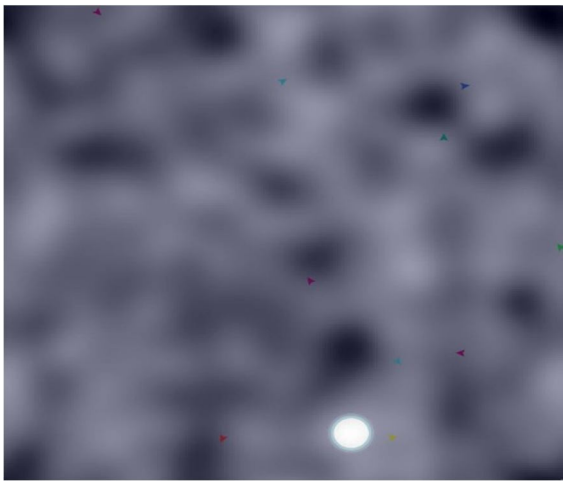
Present: 2 6 3 5 4 1 Swap :(2, 5)
5 6 3 2 4 1 Swap: (6, 3)
5 3 6 2 4 1 Swap: (2, 4)
5 3 6 2 4 1—$P_{gbest.}$

Hence (2, 5), (6, 3), and (2, 4) are used for getting P_{gbest} from the present sequence.

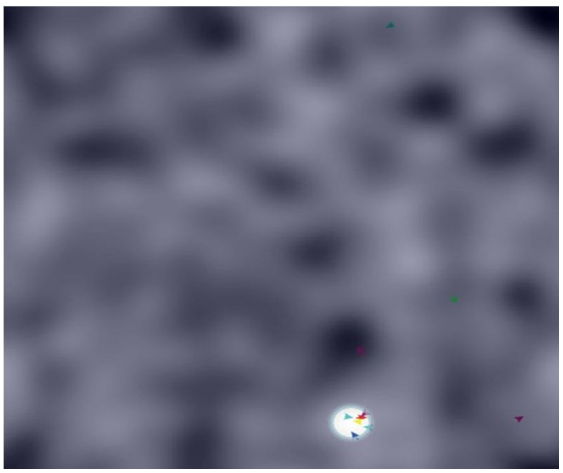
$V_{id} = \omega.V_{id} + C_1*rand()* [P_{id} – X_{id}] + C_2*rand() [P_{gd} – X_{id}]$

Velocity=1*1{(2,6),(2,1),(3,2),(4,3)}+1*0.57{(2,5),(6,3),(2,4)}

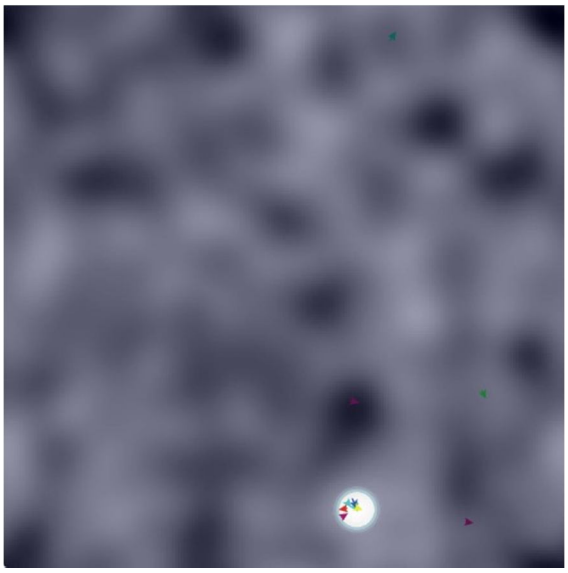
The 57% of the change in both parts is considered. Hence the first two changes in both the parts (2,6), (2,1) and (2,5),(6,3) are considered. New sequence=present+ velocity=2 6 3 5 4 1+ (2, 6), (2, 1), (2, 5), (6, 3) Hence the sequence generated for the next generation is 3 1 6 2 4 5. Similarly for all other particles the new sequences are generated, and the objective function is evaluated and is shown in Fig.4



a. Randomly distributed particles



b. Particles position after 50 interactions



c. Particles position after 90 interactions

Figs 4: Flowchart of PSO Algorithm

Table and Graph

Iterations	Optimization results (gbest)	Fitness (gbest)
1	gbest = (0.97, 0.77, 7.98, 2.56, 12.03)	19265
10	gbest = (0.65, 0.58, 4.82, 3.69, 10.22)	25103
20	gbest = (0.69, 0.60, 1.67, 8.82, 10.83)	26156
30	gbest = (0.70, 0.60, 1.67, 8.84, 12.28)	30156
40	gbest = (0.89, 0.40, 1.67, 8.82, 8.10)	41269
50	gbest = (0.85, 0.49, 0.95, 1.55, 8.80)	44312
60	gbest = (0.30, 0.59, 5.41, 5.89, 11.47)	44438
70	gbest = (0.71, 0.64, 1.37, 4.67, 11.06)	44569
80	gbest = (0.55, 0.61, 1.16, 2.77, 9.14)	45625
90	gbest = (0.55, 0.61, 1.16, 2.77, 9.14)	45625

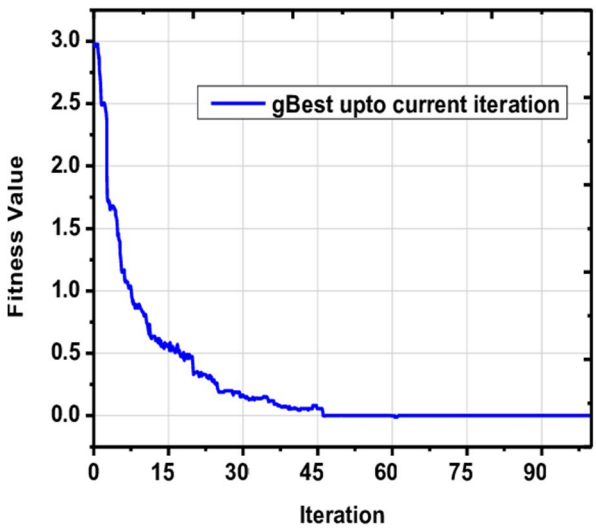


Fig 5: Graph

V. ANT COLONY OPTIMIZATION

A. Introduction

Ant Colony Optimization (ACO) emerged in the 1990s as a pioneering metaheuristic algorithm inspired by the foraging behavior of ants. It was conceptualized by Marco Dorigo and his team, drawing insights from observations of ant colonies' efficient food-gathering processes. ACO falls within the domain of swarm intelligence, leveraging the collective actions of simple agents to address complex optimization problems. Historically, Marco Dorigo, during his tenure at the Free University of Brussels, developed the foundational Ant System, an early instance of ACO tailored for solving the Traveling Salesman Problem (TSP). Since its inception, ACO has undergone substantial refinement, leading to its widespread adoption across diverse optimization domains.

At its essence, ACO mimics ants' pheromone trail-laying behavior. Ants deposit pheromones along paths they traverse, with the intensity of these trails reflecting path attractiveness. The algorithm iteratively constructs solutions by probabilistically selecting components based on both pheromone trails and heuristic information. As iterations progress, paths yielding superior solutions accumulate more pheromones, steering subsequent iterations towards increasingly optimal solutions.

B. Key Components of ACO

1) Pheromone Trails: Representing collective memory guiding exploration.

2) Heuristic Information: Directing the search towards promising regions of the solution space.

3) Pheromone Update Rule: Governing pheromone deposition and evaporation, balancing between exploration and exploitation.

ACO has found application in diverse optimization challenges such as routing, scheduling, and logistics. Its efficacy in handling complex, combinatorial problems with irregular structures has made it indispensable in both academic research and industrial applications. The success of ACO has spurred further innovation, leading to the development of hybrid and variant algorithms tailored to address specific optimization challenges.

Figure AA

In our proposed method, we employ ACO with three paths and twelve "ants" to simulate signal transmission between communication towers. The paths represent different signal routes, while the ants symbolize the transmissions sent and received within a cycle of time.

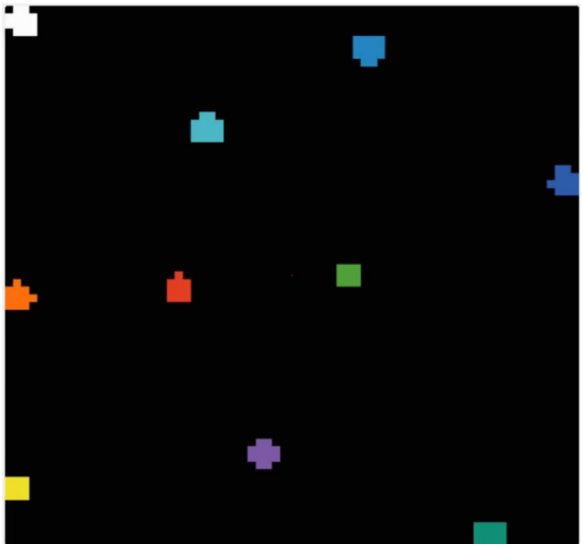


Figure BA

To simulate our model, we utilized NetLogo, a program capable of running simulations based on predefined code. Our model comprises three main components: the nest, food piles, and ants. In our representation, Information Tower 1 corresponds to the nest, while the remaining nodes are depicted as food piles with distinct colors. These nodes are positioned approximately in accordance with their real-world locations to provide a 1:1 representation of the distances between them. For visual clarity, we provide two figures: Figure AA depicting a map of Yellowstone National Park with labeled paths, and Figure BA illustrating the map representation within the NetLogo program. Additionally, Figure BB provides a key explaining the representation of each area in Figure BA about Figure AA.

Color	Node Type	Node
White	Nest	Information Tower 1, Point 1
Cyan	Food Pile	Point 2
Sky Blue	Food Pile	Point 3
Blue	Food Pile	Point 4
Green	Food Pile	Point 5
Red	Food Pile	Point 6
Orange	Food Pile	Point 7
Yellow	Food Pile	Point 8
Purple	Food Pile	Point 9
Turquoise	Food Pile	Information Tower 2, Point 10

Figure BB

The purpose of the simulation is to determine the shortest and most frequently used path among three options: Path L, Path M, and Path R, as depicted in Figure AA. The simulation, facilitated by the NetLogo program, employs the nest as a point of origin, with ants more inclined to head toward it as they get closer.

In operation, the simulation starts with the release of 200 ants randomly into the simulation area. When an ant reaches a food pile, it promptly returns to the nest to deposit the food, leaving a trail of pheromones. As ants move, the black background gradually turns white, indicating the accumulation of pheromones. The identification of the optimal path relies on two main factors: the time taken for pheromones to envelop a path and the direction ants take after reaching Information Tower 2 (Point 10). Due to program limitations preventing the existence of two nests, the direction of ants leaving Point 10 is particularly significant in determining the optimal path.

The results of the simulation indicate that as ants reached Point 10, they overwhelmingly headed directly toward Point 5 instead of Point 9 (refer to Figure CA). The simulation concluded upon the depletion of food at Point 5, marking a convergence criterion. Figure CB demonstrates that none of the ants leaving from Point 10 headed towards Point 4, suggesting that Path M is the most optimal route back to the nest.

In summary, the simulation effectively determines the most efficient path for ant movement, highlighting the practical application of ACO in solving optimization problems.

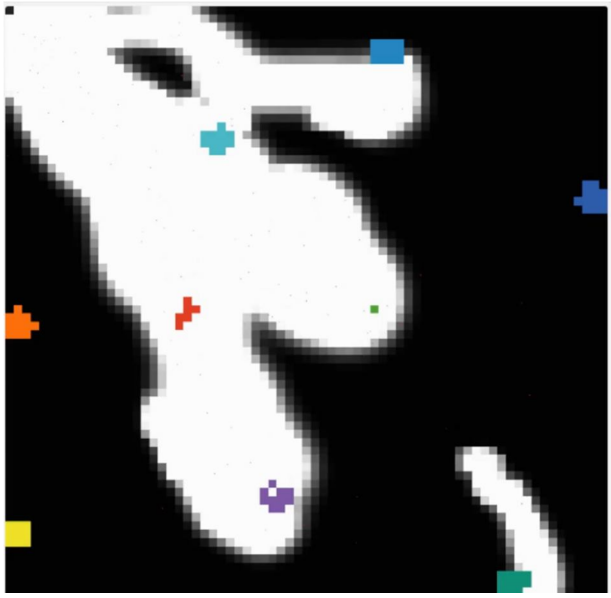


Figure CA

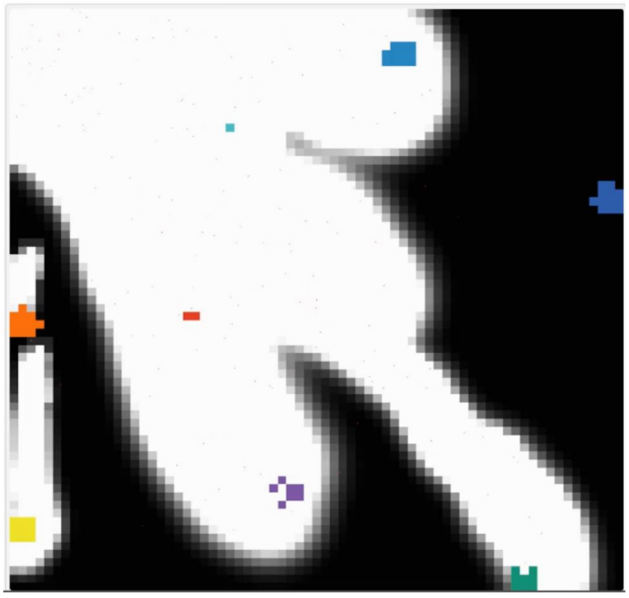


Figure CB

VI. CONCLUSION

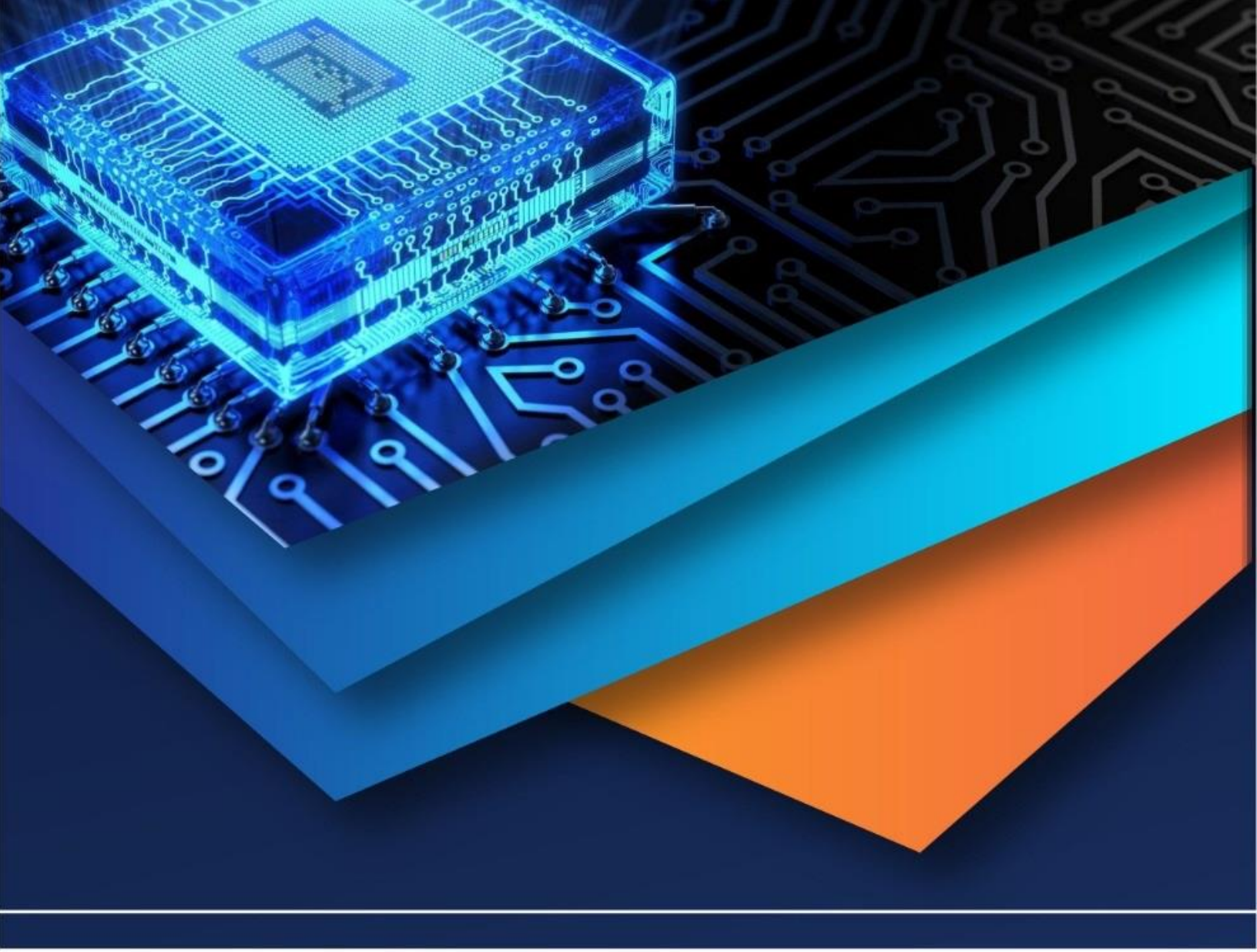
In conclusion, after a comprehensive analysis of Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) for the crucial task of detecting wildfires using wireless sensor nodes in Yellowstone National Park, it becomes evident that PSO emerges as the superior choice.

PSO showcases remarkable efficacy in optimizing the deployment of sensor nodes, efficiently maximizing the coverage area while minimizing energy consumption. Its ability to swiftly converge to optimal solutions, adapt to dynamic environmental changes, and mitigate the impact of local optima sets it apart in the context of wildfire detection.

While ACO demonstrates notable capabilities, particularly in complex routing scenarios, its performance in this specific application falls short compared to PSO. The inherent characteristics of PSO, such as its simplicity, scalability, and robustness, make it the preferred optimization algorithm for ensuring timely and accurate wildfire detection, thereby significantly enhancing the safety and conservation efforts within Yellowstone National Park.

REFERENCES

[1] J. Yick, B.Mukherjee, and D.Ghosal, “Wireless Sensor Network Survey”, computer networks, Vol 52,no-12,pp. 2292 2330,2008
[2] Eberhart R. and Kennedy J. (1995). A new optimizer using Particle Swarm Theory. Proc 6th Int. symposium on Micromachine and Human Science, pp.39-43
[3] Poli, R. 2007. Analysis of the publications on the Applications of Particle Swarm Optimisation, Journal of Artificial Evolution and Applications, 2008, 1-10.
[4] Lazinica, A. 2009. Particle Swarm Optimization. (Edited), Published by In-Tech, Croatia, January
[5] K.V. Price, R.M Storn, and J.A Lampinen, Differential evolution: A Practical Approach to Global Optimization, Ser. Natural Computing series; Berlin, Germany: Springer-Verlag, 2005



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY