Swarm vs Evolutionary Algorithms for Path Optimization on Lunar Surfaces: A Metaheuristic Approach

*

Ritul Mamdapur*, Sukanya Dharwad†, Srushti Sandigawad†, Sneha Kayakad†, and Rashmi Benni†
Department of MCA, K L E Technological University, Hubballi, India
ritulmamdapur17@gmail.com, sukanyadharwad2004@gmail.com, srushtisandigawad@gmail.com,
snehakayakad5@gmail.com, rashmi.benni16@gmail.com

***Abstract*—Path optimization is a critical challenge in autonomous lunar exploration, where efficient traversal of rugged terrain is vital for mission success. Compared to other celestial bodies, the lunar environment presents unique challenges such as extreme temperature variations, high radiation levels, and a lack of atmosphere, making it a critical focus for autonomous exploration. Additionally, unlike other distant planets, the Moon's proximity to Earth and its potential for future human missions makes it a priority for path optimization research. Traditional machine learning algorithms often fall short in this domain due to their dependency on large datasets and tendency to over-fit. They are incompetent when dealing with dynamically changing environments like lunar terrain. These algorithms also struggle with the unpredictability of rugged terrains, which demand real-time adaptability and robustness—qualities that machine learning techniques often lack. In this study, we address these limitations by employing meta-heuristic algorithms, specifically Differential Evolution (DE), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). Lunar terrains, characterized by craters, sharp slopes, and high-cost traversal areas, require algorithms like DE, which uses mutation and recombination for exploration, PSO for efficient search through particle velocity updates, and ACO for intelligent path selection based on pheromone trails. Applying these methods to a simulated lunar dataset with varying terrain costs, we compare their effectiveness in path-finding, emphasizing their ability to minimize traversal costs while navigating complex, uncertain environments more reliably than traditional approaches.**

***Index Terms*—space rovers, search space optimization, meta-heuristics, particle swarm optimization(PSO), Differential Evolution (DE), Ant Colony Optimization(ACO), optimal solutions, exploration, heuristic approach, path optimization, and nature-inspired.**

I. Introduction

Rover path optimization is a crucial aspect of autonomous lunar exploration, where efficient navigation through unpredictable and challenging terrain is key to mission success. On the lunar surface, rovers must avoid obstacles, adjust to varying terrain conditions, and find the shortest and safest routes to their destination, ensuring minimal energy consumption and reduced risk of damage. The rocky, uneven, and cratered lunar landscape presents complex pathfinding challenges, requiring real-time decision-making capabilities to achieve mission objectives efficiently. Effective path planning on such terrain is essential to maximizing mission time, conserving resources, and preventing rover malfunction or failure.

Although powerful in many domains, machine learning algorithms face significant limitations when applied to this problem. They often rely on extensive datasets, struggle with generalizing to unseen environments, and tend to overfit to known conditions, making them less adaptable to the highly dynamic and uncertain nature of lunar landscapes [13]. Their tendency to get trapped in local optima, coupled with the computational cost of real-time processing in resource-constrained environments, makes them less suitable for autonomous lunar navigation tasks. As a result, meta-heuristic algorithms, which are designed to tackle complex optimization problems in uncertain and highly variable environments, provide a more robust and flexible alternative [6].

In this study, we focus on evolutionary and swarm-based meta-heuristic algorithms, specifically Differential Evolution (DE), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) [15], [16]. DE, an evolutionary algorithm, mimics natural selection processes like mutation and recombination to explore solutions, offering adaptability and diversity in search spaces [5], [7]. In contrast, ACO simulates the pheromone-based pathfinding behavior of ants, and PSO models the social dynamics of flocking birds to optimize search processes. These algorithms offer a balance between exploration and exploitation, crucial for navigating the rugged, high-cost traversal areas of lunar terrain. By comparing their performance in a simulated lunar environment, we aim to determine which approach is most effective in minimizing traversal costs while optimizing rover pathfinding under real-world constraints. Understanding the nuances of these methods will provide valuable insights into improving the autonomy and efficiency of future lunar exploration missions. Along with the difficulties posed by lunar terrain, rover path optimization also needs to take into consideration energy constraints, communication delays, and limited onboard processing power, all of which can have a big impact on mission success [8]. Because of the significant time difference between Earth

and the Moon, direct human intervention is limited, so the rover's capacity to make decisions on its own in real time is essential. By integrating effective meta-heuristic algorithms into its control system, the rover can balance risk factors, make energy-efficient navigation decisions, and dynamically adjust to the changing terrain without constant supervision [1]. Furthermore, algorithms must be able to quickly re-calibrate in the event of unforeseen hazards due to the lunar environment's unpredictable nature, which includes obstacles like craters, boulders, and steep slopes.

The article is well-structured. Section II discusses the initial investigation. Section III discusses the suggested work. Part IV presents outcomes and remarks Section V concludes the investigation, and Section VI discusses the research's future scope.

II. RELATED WORK

Path optimization, particularly in lunar environments is still an important area of research. Rovers can respond to these challenges using metaheuristic approaches, which provide strong solutions to the dynamic and unpredictable character of the processes under consideration. For these reasons, swarm intelligence algorithms such as PSO and ACO are particularly appropriate for lunar surfaces since they easily model terrain-based adaptation through their cooperative nature. To name a few, Yu Wang et al. used a PSO co-evolution algorithm with simulated annealing for UAV path planning and their results provide faster convergence as well as the ability to escape local optima in 3D spaces [18]. Changsheng Huang further improved PSO by a hybrid algorithm combining A* and PSO which also proved to be a more efficient version of A*, called APSO* [9] for mobile robots. Dongming Zhao first proposed the Multi-object Coordination with a Multicolony Symbiotic Optimization Algorithm (MOCMCSO) to balance path length and smoothness in intelligent patrol systems [19].

Besides swarm intelligence, many ACO-based methods are also shown to be effective. In their work, Tianfeng Zhou and Wenhong Wei developed ACO for mobile robots to enhance the mobility of such agents by refining pheromone update mechanisms [20]. Christopher Carr and Peng Wang proposed the Fast-Spanning Ant Colony Optimization (FaSACO) that changed ant velocities to improve running times for various coverage path planning algorithms [4]. Dongdong Li et al used TDI-MSACO for flexibility and movement and also enhanced the convergence on rough lunar terrains and Muhammad Aria Rajasa Pohan et al. also used hybrid algorithms like RRT-PSO combining global and local search techniques for faster convergence in dynamic obstacle avoidance, which is widely applied in challenging environments [9], [12]. N. Volpato et al., Proposed a compatible sliding window approach in their work on tool-path optimization [14].

Differential Evolution (DE) is another effective way to optimize paths. Jun Chen et al. applied chaos initialization on DE for faster convergence and higher accuracy in the case of irregular lunar terrains [5]. Zheng Cai and coworkers incorporated DE with the Whale Optimization Algorithm (WOA), leading to enhanced efficiency on high-dimension challenges [3]. Ximing Liang has shown on a competitive swarm optimization with Levy flights [10], that bio-inspired algorithms such as these are promising too. The objective of Rashmi Benni was to search for an optimal path for autonomous mobile robots which was achieved by using FA, PSO, and DE [2].

Specific space-oriented solutions have also been pinpointed. Chen Yihan (2015) developed a PSO method that was suitable for the CubeSats, due to this it achieved and addressed specific geometry challenges in space crawling robots [17]. Sharan Nayak introduced a multi-rover motion planning method that was able to allocate and avoid the target in planetary terrains [11]. This highlights the crucial necessity of metaheuristic methods in achieving an adaptive and efficient path-planning methodology for space exploration as well as robotic applications. These approaches not only offer flexibility in handling the dynamic nature of extraterrestrial terrains but also provide robust solutions to time and resource constraints. As lunar exploration continues to evolve, the need for optimized algorithms tailored to these environments becomes increasingly critical.

III. PROPOSED WORK

The proposed study compares and evaluates various meta-heuristic strategies, including DE, ACO, and PSO, to optimize the lunar rover path planning. The goal is to improve the rover's decision-making and efficiency through an iterative process and meticulous hyperparameter optimization. Additionally, these techniques aim to explore the lunar terrain's search space to identify the most optimal path, reduce the terrain cost, and navigate through challenging environments while minimizing resource usage.

A. System Architecture

The rover path optimization system architecture on the lunar terrain utilizes metaheuristic algorithms designed to address the environment's peculiar difficulties, like craters and steep inclines. First, the input data with the terrain characteristics representing the cost of the terrain is prepared and cleaned, and the problem constraints are defined. This process involves filtering the data to remove noise or inconsistencies and ensuring that the terrain cost values are accurate. Next, through diligent hyper-parameter definition, algorithms such as Differential Evolution (DE), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) are selected and fine-tuned. Potential paths are assessed according to their cost concerning terrain obstacles using a fitness function in the optimization process. As new paths are generated, parameter tuning ensures continuous refinement of the solutions. This process continues until the most feasible path is found, at which point the newly generated path that outperforms the previously generated ones becomes the current optimal solution. The rover then follows the final best route to ensure safe and effective navigation across the lunar surface.

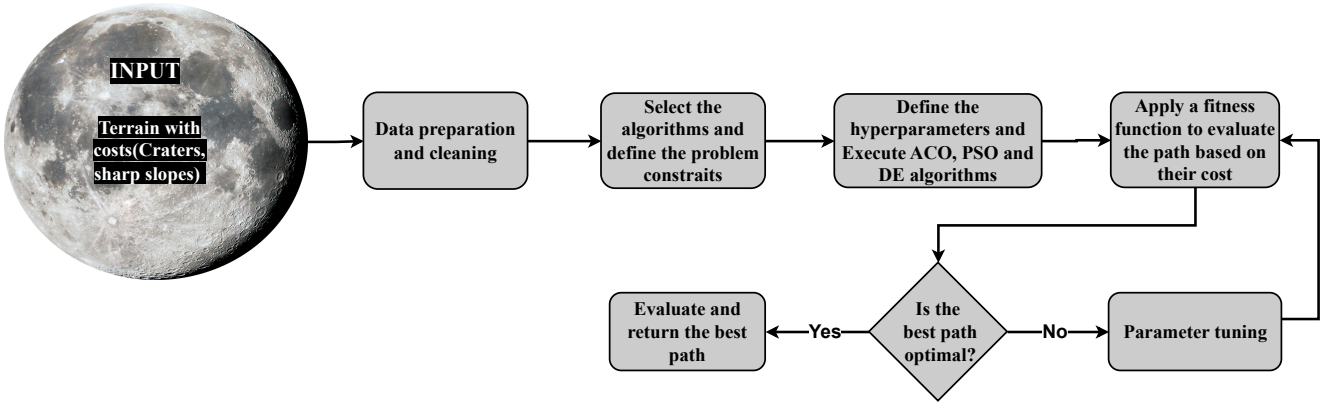

Fig. 1: System Architecture

B. Particle Swarm Optimization Algorithm

Particle Swarm Optimization is an optimization algorithm inspired by the social behavior of birds and fish schools. To evaluate the search space, it employs a population of potential solutions known as particles, which move around based on both the swarm's global best solutions and their personal best solutions. Each particle's individual experience and the collective knowledge of the swarm affect its mobility. In the context of space rover path planning, PSO helps identify the most efficient route by balancing factors like terrain difficulty, obstacles, and energy consumption. By iterating over possible paths, PSO ensures the rover navigates safely while optimizing resource usage, ultimately achieving the mission goals with minimal risk to the particles.

- Initialization
  Position $P_i = [p_{i1}, p_{i2}, ..., p_{in}]$
  Velocity $V_i = [v_{i1}, v_{i2}, ..., v_{in}]$
  where n is the lunar terrain dimension .

- Updating velocity

$$V_i(t{+}1) = w{\cdot}V_i(t){+}c1{\cdot}r1{\cdot}(P_i{-}X_i(t)){+}c2{\cdot}r2{\cdot}(G{-}X_i(t)) \tag{1}$$

  Tune the velocity of each particle based on its inertia, cognitive (personal best), and social (global best) components.
  w is the inertia weight, $c1$ and $c2$ are cognitive and social coefficients, $r1$ and $r2$ are random numbers, P_i is the personal best, G is the global best.
- Updating position

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{2}$$

  Updating the position of every particle based on its latest position and updated velocity.
  $X_i(t)$ is the latest position, $V_i(t+1)$ is the updated velocity.
- Updating personal best

$$P_i(t+1) = \begin{cases} X_i(t+1) & \text{if } f(X_i(t+1)) < f(P_i(t)) \\ P_i(t) & \text{otherwise} \end{cases} \tag{3}$$

  Updating the personal best position if the current position has a better fitness value.
  $P_i(t)$ is the personal best, $X_i(t{+}1)$ is the current position, and $fitness$ is the path evaluation function.
- Updating global best

$$G(t+1) = X_j(t+1) \tag{4}$$

  Changing the global best position to the best fitness value among all particles.
  $G(t{+}1)$ is the global best, $X_j(t{+}1)$ is the best position of particle j.
- Stopping criterion
  The algorithm terminates after a set number of iterations or when an optimal path on the lunar terrain is found.

C. Ant Colony Optimization

The Ant Colony Optimization (ACO) is a metaheuristic algorithm inspired by the cooperative behavior observed in ants. Specifically, it borrows from how ants locate the best routes between their colony and food sources by depositing pheromones along the paths they traverse; shorter and more effective routes build up greater concentrations of pheromones over time, which attracts additional ants to them. In the end, the majority of the colony is guided toward the best path by this positive feedback loop. The same behavior is adapted in this study to navigate complex terrain where different regions of the surface exhibit varying traversal costs due to craters, slopes, and other obstacles. Multiple ants are released to investigate routes, considering both the local terrain cost and the artificial pheromone trail. As ants identify more efficient paths, those paths accumulate higher pheromone levels, while less optimal routes experience pheromone decay. This approach enables the algorithm to dynamically adapt and converge on the lunar rover's safest and most cost-effective route while also

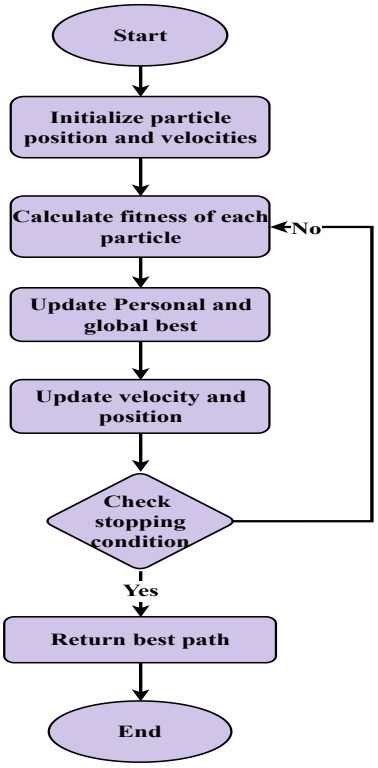

Fig. 2: The Particle Swarm Optimization Algorithm

minimizing energy consumption and time spent on hazardous terrain.

- Initialization
  Initialize the pheromone matrix and parameters for ants exploring the lunar terrain.

- Updating pheromone

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{5}$$

  Updating the pheromone intensity on the edges, considering evaporation rates and the new pheromones deposited by ants.
  $\tau_{ij}(t)$ is the pheromone on edge ij, ρ is pheromone evaporation rate, and $\Delta\tau_{ij}(t)$ is the deposited pheromone.
- Path selection probability

$$P_{ij}(t) = \frac{(\tau_{ij}(t))^{\alpha} \cdot (\eta_{ij})^{\beta}}{\sum_{k \in \text{allowed}} (\tau_{ik}(t))^{\alpha} \cdot (\eta_{ik})^{\beta}} \tag{6}$$

  Selecting a path based on pheromone values.
  α is the pheromone weight, β is the heuristic weight, η_{ij} is the heuristic information (inverse distance).

- Pheromone deposit

$$\Delta\tau_{ij}(t) = \sum_{\text{ant } k} \frac{Q}{L^k} \tag{7}$$

  Path distance is inversely proportional to the pheromone contribution of each ant .
  Q is the total pheromone deposit, L^k is the length of the path traveled by ant k.
- Stopping criterion
  The algorithm terminates when it reaches the maximum number of iterations or the most optimal path on the lunar terrain is found.

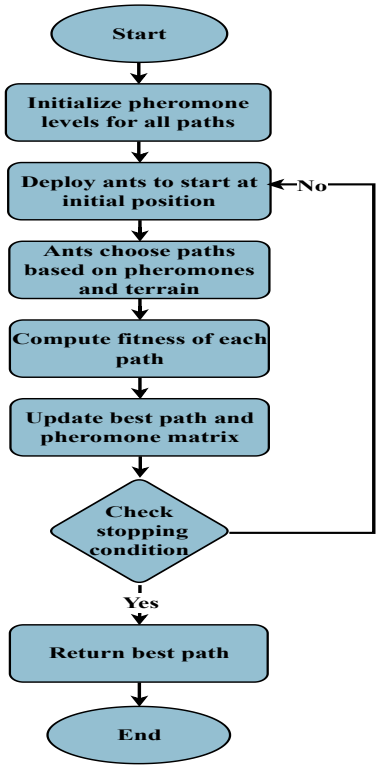

Fig. 3: The Ant Colony Optimization Algorithm

D. Differential Evolution Algorithm

DE algorithm is an optimization method that falls under the category of genetic algorithms for solving multi-dimensional problems. It begins by generating a population of potential solutions, then iteratively refines them through three key operations: mutation is another variation method that deals with evolving solutions, crossover is the method of recombination of solutions, and lastly, selection is used to select optimality through a proper fitness function. When it comes to planning for space rovers, DE helps identify the optimal path, considering factors like surface conditions and barriers.

As such, this exploration examines a huge number of links and therefore identifies the most optimal or subservices to the best solution. This ensures that the rover moves on drives without compromising on unnecessary risks or expending resources in the process of efficiently accomplishing the laid down mission.

- Initialization
  Initialize a population of N candidate paths across the lunar terrain:
  $$X = \{x_1^{(0)}, x_2^{(0)}, \ldots, x_N^{(0)}\} \tag{8}$$
  $x_i^{(0)}$ represents the initial paths, and N is the size of the population.
- Mutation
  For each target path x_i, generate a mutant path v_i:
  $$v_i = x_a + F \cdot (x_b - x_c) \tag{9}$$
  Creates a mutant path by combining the scaled difference of two other paths with the new offspring path.
  x_a, x_b, x_c are distinct candidate paths, F is the mutation factor.
- Crossover
  Create a trial path u_i by mixing components of the mutant path v_i and target path x_i:
  $$u_{ij} = \begin{cases} v_{ij} & \text{if rand}_j \leq \text{CR or } j = j_{\text{rand}} \\ x_{ij} & \text{otherwise} \end{cases} \tag{10}$$
  Generates the trial path by combining the mutant and target paths based on crossover probability.
  CR is the crossover probability, rand_j is a random number, and u_{ij}, v_{ij}, x_{ij} are components of the trial, mutant, and target paths.
- Selection
  Evaluate and select between the trial path u_i and target path x_i:
  $$x_i^{(t+1)} = \begin{cases} u_i & \text{if } f(u_i) \leq f(x_i^{(t)}) \\ x_i^{(t)} & \text{otherwise} \end{cases} \tag{11}$$
  Selects the path with the most optimal fitness between the trial and target paths.
  $f(u_i)$ and $f(x_i^{(t)})$ represent the fitness values of the trial and target paths, and t is the iteration.
- Stopping criterion
  The algorithm stops after a fixed number of iterations or when the most optimal fitness value is reached, representing the shortest lunar path.

IV. RESULTS AND DISCUSSION

For the comparative analysis, we ran several tests to assess how well three meta-heuristic algorithms—Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Differential Evolution (DE)—performed in resolving lunar path planning issues. To imitate increasingly challenging lunar conditions, the trials were conducted on three distinct test cases: 10x10, 20x20, and 30x30 grids. Each test case represented a different level of complexity. The trials were designed

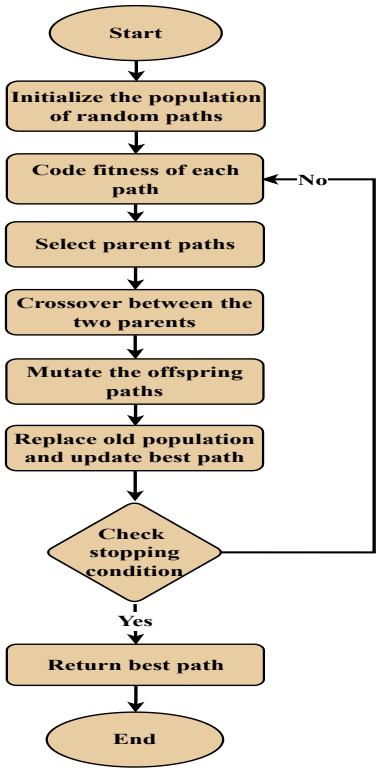

Fig. 4: The Differential Evolution Algorithm

in three phases to allow for a more in-depth study. To see how these parameters affect the algorithms' performance and to evaluate algorithm performance under more difficult settings on different grid sizes, we varied the population and iterations across all grid sizes. Key factors such as CPU time, fitness values, and convergence speed were analyzed throughout each phase to compare the performance of DE, PSO, and ACO in finding optimal paths across different grid sizes.

Across various iterations and population parameters, ACO consistently produced the most optimal pathways in the 10x10 grid. It took 26.9 seconds to reach a final fitness of 132 after 10 iterations and with a population size of 50. The path quality was marginally improved by increasing the population to 75 and the number of iterations to 20, which decreased the final fitness to 123 but raised the CPU time to 27.5 seconds. ACO determined the optimal path with a final fitness of 95 for the longest test (50 iterations and 100 population), although the computation time increased significantly to 81.3 seconds. DE, on the other hand, showed far quicker convergence, achieving a final fitness of 239 for 10 iterations and 50 populations in just 3.6 seconds. The fitness was 220 by doubling the iterations to 20. DE's strength in speed is demonstrated by the 50 iterations with 100 population test cases, which produced a fitness of 120

yet took 6.7 seconds. PSO trailed in terms of speed and path optimization. In 7.2 seconds, after 10 iterations and 50 populations, it discovered a path with a 344 fitness value. The fitness grew a little to 327 at 20 iterations and 75 population, while the computation time went up to 19 seconds. PSO reached a fitness of 220 after 56.5 seconds, showing modest improvement but at a much higher computational cost.

10x10 Grid Analysis				
Algorithm	Iterations	Population	Final Fitness	CPU Time (secs)
ACO	10	50	132	26.9
PSO	10	50	344	7.2
DE	10	50	239	3.6
ACO	20	75	123	27.5
PSO	20	75	327	19.0
DE	20	75	220	2.4
ACO	50	100	95	81.3
PSO	50	100	220	56.5
DE	50	100	120	6.7

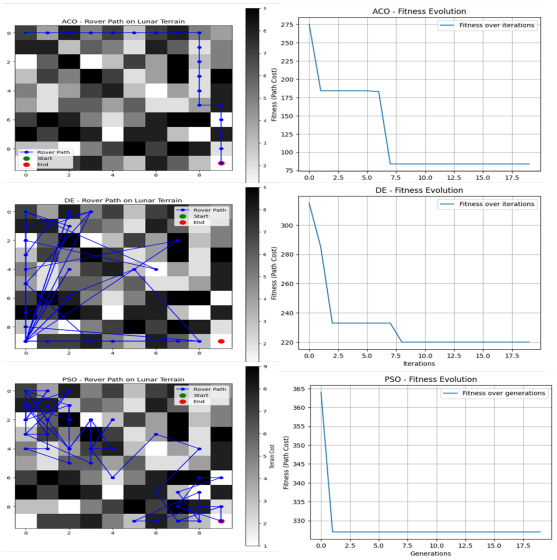

Fig. 6: Test Case 1: Rover Path Optimization on 10x10 Terrain

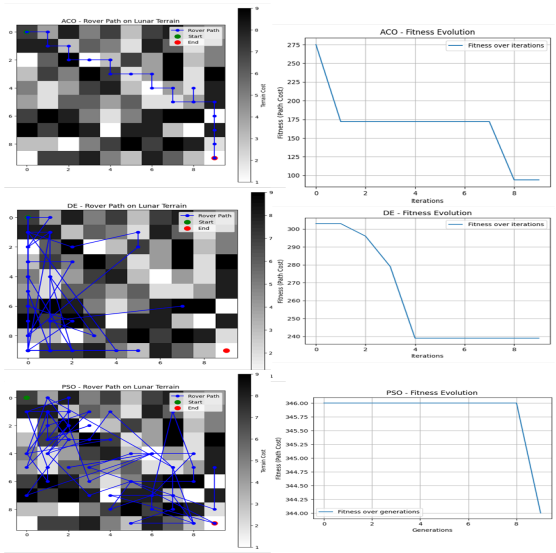

Fig. 5: Test Case 1: Rover Path Optimization on 10x10 Terrain

With 10 iterations and 50 populations in 52.2 seconds, ACO achieved a final fitness of 709 in the 20x20 grid, demonstrating its ability to produce the most efficient path. As iterations and population increased to 20 and 75 respectively, the fitness improved to 617. After 50 iterations and 100 populations, the optimal path was discovered and the final fitness was 439 and a computation time of 587.6 seconds. The ACO graphs in this test scenario exhibit early stabilization of fitness values in higher iteration runs, along with smooth convergence towards the optimal path.

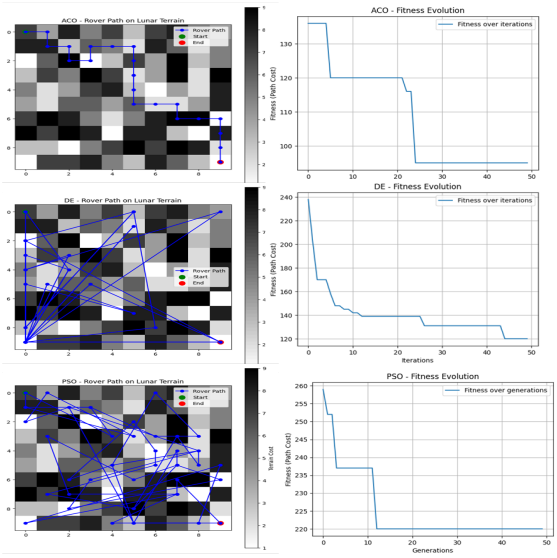

Fig. 7: Test Case 1: Rover Path Optimization on 10x10 Terrain

DE demonstrated the fastest performance, completing 10 iterations and 50 populations with a fitness of 781 in 3.4 seconds. Even then, DE occasionally fell short of completing the best courses, especially as the grid's size and complexity rose.DE demonstrated its efficiency in convergence speed by achieving a fitness value of 534 for the 20x20 grid in just 8.4 seconds with 20 iterations and a population size of 75.

However it started to show early stagnation, falling short of the ideal fitness values that ACO was able to attain. As it can be observed in the graphs where fitness values plateau before reaching the optimal solution, DE's routes frequently converged prematurely in the bigger test cases. The fitness was 775 for 50 iterations and 100 population, but even with the algorithm's fast 24-second completion time, the best solution was not produced. PSO was unable to match the path quality of ACO, achieving a final fitness of 1066 in 22.7 seconds after 10 iterations and 50 population. Even though convergence was slower, the PSO graphs indicated a steady improvement as the number of iterations rose. The greatest test case, with 50 iterations and 100 population, obtained a fitness of 1260 in 237.1 seconds, exhibiting sluggish but continuous convergence tendencies. With 20 iterations and 75 population, the fitness increased to 602 in 65 seconds.

20x20 Grid Analysis				
Algorithm	Iterations	Population	Final Fitness	CPU Time (secs)
ACO	10	50	709	52.2
PSO	10	50	1066	22.7
DE	10	50	781	3.4
ACO	20	75	617	243.5
PSO	20	75	602	65.0
DE	20	75	534	8.4
ACO	50	100	439	587.6
PSO	50	100	1260	237.1
DE	50	100	775	24.0

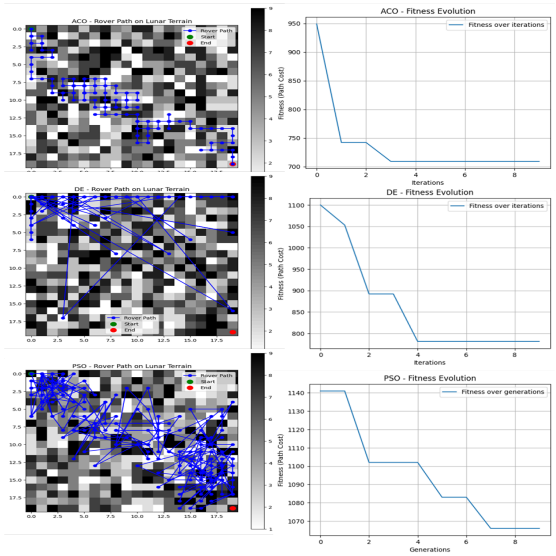

Fig. 8: Test Case 2: Rover Path Optimization on 20x20 Terrain

Even in the most intricate 30x30 grid, ACO's supremacy

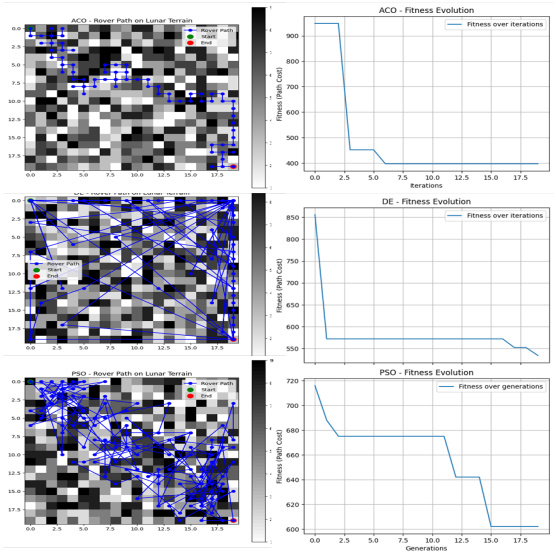

Fig. 9: Test Case 2: Rover Path Optimization on 20x20 Terrain

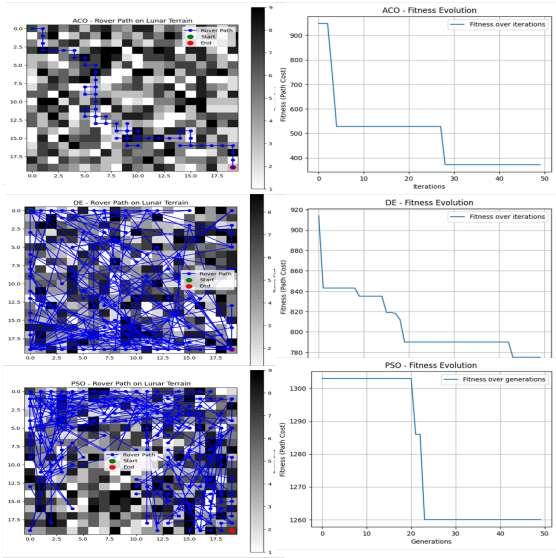

Fig. 10: Test Case 2: Rover Path Optimization on 20x20 Terrain

in path optimization was maintained. ACO reached a 1910 fitness after 10 iterations and 50 population in 156.3 seconds, and a 1433 fitness after 20 iterations and 75 population in 551.5 seconds.

30x30 Grid Analysis				
Algorithm	Iterations	Population	Final Fitness	CPU Time (secs)
ACO	10	50	1910	156.3
PSO	10	50	2003	78.9
DE	10	50	2019	8.5
ACO	20	75	1433	551.5
PSO	20	75	2058	227.3
DE	20	75	1982	18.6
ACO	50	100	1348	1864.0
PSO	50	100	1993	783.1
DE	50	100	1944	48.3

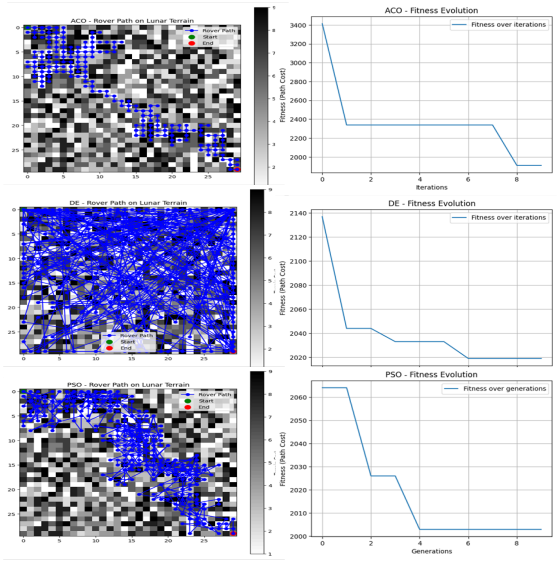

Fig. 11: Test Case 3: Rover Path Optimization on 30x30 Terrain

ACO achieved a final fitness of 1348 with 50 iterations and 100 population, which was the best result. While ACO requires more computing, its graphs of convergence show a slower but consistent improvement in overall test scenarios, suggesting that ACO eventually narrows down on the best course. Even with its increased speed, DE found it more difficult to maneuver in this larger grid. It generated a fitness of 2019 in 8.5 seconds for 10 iterations with 50 populations,and at a population size of 75 and 20 iterations, DE generated a fitness value of 1982 in 18.6 seconds. But once again, DE did not finish the pathways in more complicated cases, causing fitness values to prematurely stagnate in the graphs.This problem was more noticeable in the longer runs, where DE converged to a less-than-optimal fitness of 1944 after 48.3 seconds with 50 iterations and 100 population, even after increasing the number of iterations and population. As the number of iterations rose, PSO performed better, but it was

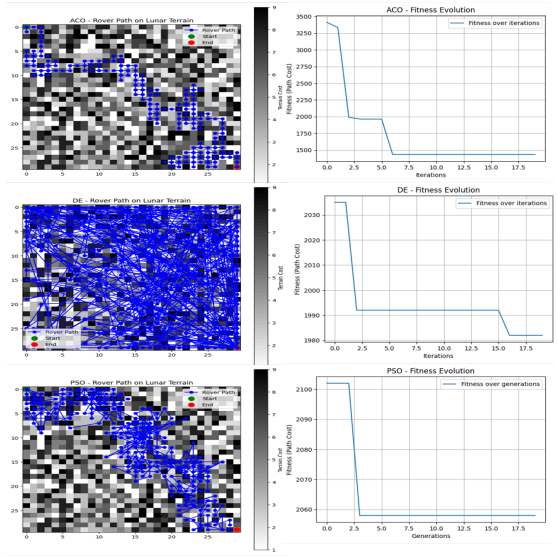

Fig. 12: Test Case 3: Rover Path Optimization on 30x30 Terrain

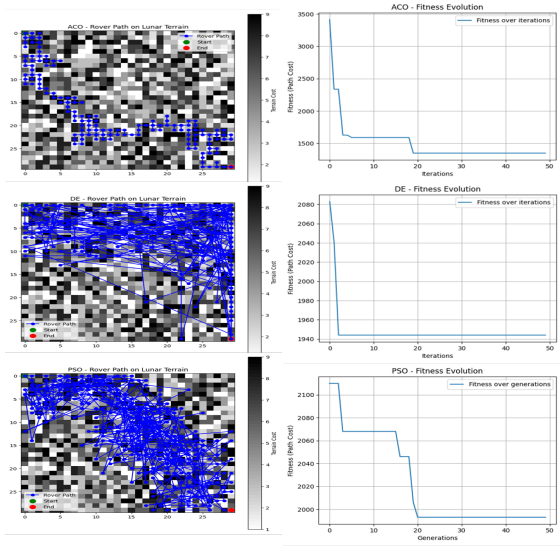

Fig. 13: Test Case 3: Rover Path Optimization on 30x30 Terrain

still slower than DE and less optimized than ACO. After 78.9 seconds and 10 iterations with 50 population, PSO generated a fitness of 2003. PSO completed the more difficult test scenario in 227.3 seconds with a fitness of 2058 after 20 iterations and 75 population. With 50 iterations and a population size of

100, PSO demonstrated continuous convergence with a fitness value of 1993 in 783.1 seconds. Even with longer iteration runs, the graphs show slow but consistent progress; however, convergence took much longer than for ACO and DE.

V. Conclusion

Based on the analysis and findings, on larger and denser grids, Ant Colony Optimization (ACO) showed the best performance for path optimization on lunar terrains, consistently achieving optimal fitness values on larger and denser grids at a significantly higher computational time. In every test case (10x10, 20x20, and 30x30 grids), ACO's iterative pheromone-guided mechanism successfully navigates difficult terrains with craters and steep slopes. On smaller grids (10x10), Differential Evolution (DE) demonstrated exceptional convergence speed, reaching optimal solutions quickly. However, in more complex scenarios (20x20 and 30x30 grids), DE struggled with premature stagnation and often settled on suboptimal solutions because of its limited adaptive exploration strategies. Although Particle Swarm Optimization (PSO) required more computational resources and showed slower improvements in path quality than ACO and DE, it maintained consistent performance across all grid sizes and showed gradual convergence without stagnation. These differences in performance can be explained by the algorithms' trade-offs between exploration and exploitation; ACO's collective search behavior makes it an excellent choice for difficult terrains, DE prioritizes speed over reduced optimality, and PSO strikes a balance between incremental gains and computational cost.

VI. Futurework

Future studies will explore how well other meta-heuristic algorithms perform, assess how reliable the proposed method is, and broaden the study to include more complicated situations like multi-robot systems, dynamic environments, and other terrain parameters. In difficult circumstances, the solution's flexibility and convergence speed might be further improved by using hybrid algorithms that mix swarm-based and evolutionary methodologies. The reliability, success, and sustainability of the proposed approach will be evaluated in practical applications such as transportation, surveillance, and warehouse management.

References

[1] Neil Abcouwer, Shreyansh Daftry, Tyler del Sesto, Olivier Toupet, Masahiro Ono, Siddarth Venkatraman, Ravi Lanka, Jialin Song, and Yisong Yue. Machine learning based path planning for improved rover navigation. In *2021 IEEE Aerospace Conference (50100)*, pages 1–9, 2021.

[2] Rashmi Benni, Shashikumar Totad, Karibasappa K G, and Sachin Karadgi. Search space optimization for autonomous mobile robots using meta-heuristics. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–8, 2023.

[3] Zheng Cai, Yit Hong Choo, Vu Le, Chee Lim, and Mingyu Liao. Enhancing the whale optimisation algorithm with sub-population and hybrid techniques for single- and multi-objective optimisation. *Soft Computing*, 28:1–31, 11 2023.

[4] Christopher Carr and Peng Wang. Fast-spanning ant colony optimisation (fasaco) for mobile robot coverage path planning. 05 2022.

[5] Jun Chen, Jing Liang, and Yan Tong. Path planning of mobile robot based on improved differential evolution algorithm. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 811–816, 2020.

[6] Xuefeng Dai and Yang Wei. Application of improved moth-flame optimization algorithm for robot path planning. *IEEE Access*, 9:105914–105925, 2021.

[7] David. Using particle swarm optimization as pathfinding strategy in a space with obstacles. 12 2021.

[8] Liang Hu, Wasif Naeem, Eshan Rajabally, Graham Watson, Terry Mills, Zakirul Bhuiyan, Craig Raeburn, Ivor Salter, and Claire Pekcan. A multiobjective optimization approach for colregs-compliant path planning of autonomous surface vehicles verified on networked bridge simulators. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1167–1179, 2020.

[9] Changsheng Huang, Yanpu Zhao, Mengjie Zhang, and Hongyan Yang. Apso: An a*-pso hybrid algorithm for mobile robot path planning. *IEEE Access*, 11:43238–43256, 2023.

[10] Ximing Liang, Dechang Kou, and Long Wen. An improved chicken swarm optimization algorithm and its application in robot path planning. *IEEE Access*, 8:49543–49550, 2020.

[11] Sharan Nayak, Michael Paton, and Michael W. Otte. A heuristic-guided dynamical multi-rover motion planning framework for planetary surface missions. *IEEE Robotics and Automation Letters*, 8(5):2542–2549, 2023.

[12] Muhammad Aria Rajasa Pohan, Bambang Riyanto Trilaksono, Sigit Puji Santosa, and Arief Syaichu Rohman. Path planning using combined informed rapidly- exploring random tree star and particle swarm optimization algorithms. *IEEE Access*, 12:56582–56608, 2024.

[13] Izzati Saleh, Nuradlin Borhan, Azan Yunus, and Wan Rahiman. Comprehensive technical review of recent bio-inspired population-based optimization (bpo) algorithms for mobile robot path planning. *IEEE Access*, 12:20942–20961, 2024.

[14] Neri Volpato, L.C. Galvao, L.F. Nunes, Rômulo Souza, and Karina Oguido. Combining heuristics for tool-path optimisation in material extrusion additive manufacturing. *Journal of the Operational Research Society*, 71:1–11, 05 2019.

[15] Yuting Wan, Yanfei Zhong, Ailong Ma, and Liangpei Zhang. An accurate uav 3-d path planning method for disaster emergency response based on an improved multiobjective swarm intelligence algorithm. *IEEE Transactions on Cybernetics*, 53(4):2658–2671, 2023.

[16] Lei Wang, Lili Liu, Junyan Qi, and Weiping Peng. Improved quantum particle swarm optimization algorithm for offline path planning in auvs. *IEEE Access*, 8:143397–143411, 2020.

[17] Chen Yihan, Cao Mingtao, Liu Wei, Yu Yuxin, and Shi Pangbo. Crawling robot path planning on the surface of the cubesat. *Advances in Astronautics Science and Technology*, 7, 02 2024.

[18] Zhenhua Yu, Zhijie Si, Xiaobo Li, Dan Wang, and Houbing Song. A novel hybrid particle swarm optimization algorithm for path planning of uavs. *IEEE Internet of Things Journal*, 9(22):22547–22558, 2022.

[19] Dongming Zhao, Huimin Yu, Xiang Fang, Lei Tian, and Pengqian Han. A path planning method based on multi- objective cauchy mutation cat swarm optimization algorithm for navigation system of intelligent patrol car. *IEEE Access*, 8:151788–151803, 2020.

[20] Tianfeng Zhou and Wenhong Wei. Mobile robot path planning based on an improved aco algorithm and path optimization. *Multimedia Tools and Applications*, pages 1–24, 05 2024.