Implementation of Swarm Intelligence Algorithms for Path Planning

Alex T Mathew, Ajay Paul, Akshay Rojan and Amith Thomas
Department of Electronics and Communication, Rajagiri School of Engineering and Technology, Kakkanad, Ernakulam, 682039, Kerala, India
E-mail: alexmathevi3@gmail.com

Abstract. The ability to navigate through an unknown terrain is an essential metric in evaluating the performance of multi-robot systems. With the use of swarm intelligence algorithms like Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), even robots with limited capability can perform this task. In this paper, we implement ACO and PSO to solve the path planning problem, and we also compare their efficiencies in different environments. This paper also implements a new hybrid algorithm that utilizes both ACO and PSO for more efficient path planning.

1. Introduction
Path planning is an important metric in evaluating the performance of multi-robot systems. Existing methods for path planning include A* algorithm, D* algorithm, Artificial Potential Field, Rapidly exploring random tree, etc.. By employing swarm intelligence algorithms like Particle Swarm Optimization[1] and Ant Colony Optimization[2], the pathfinding problem can be decentralized to multiple robots easily. These algorithms exploit the social behavior found in swarms like birds and ants. Intelligent behavior can be obtained from individuals with limited capability. In this study, we compare the efficiencies of ACO and PSO in multiple settings of a sample environment. A new methodology for solving the pathfinding problem using a hybrid of ACO and PSO, which drastically improves their performance, is implemented.

Earlier attempts to implement swarm intelligence algorithms for pathfinding have utilized either ACO or PSO as its backbone algorithm. Sahib Thabit and Ali Mohades[3], in their paper, proposed a new method for path planning of multi-robot systems. It implements multi-objective PSO and takes shortness, safety, and smoothness as the metrics. They have also introduced a concept of a probabilistic window that combines information obtained from the sensors and previous experiences of the robots. Jian-Hua Zhang et al.[4] proposed path planning using multi-objective bare bones PSO with differential evolution. This algorithm considers three parameters: shortness, safety, and smoothness of the path. Razif Rashid et al.[5] proposed using ACO to solve mobile robot path planning. They used different maps of varying complexities with static obstacles to test the effectiveness of the algorithm. Michael Brand et al.[6] in their paper, proposed a method to use ACO to solve path planning in a dynamic environment. They have also done a comparison of two pheromone initialization schemes.
2. Methodology
This study implements two algorithms ACO and PSO, both inspired by the behavior of natural swarms. In both these algorithms, individuals, either artificial ants or particles, provide the solution to the problem. Both PSO and ACO belong to the class of metaheuristic algorithms and are adjusted to suit the pathfinding problem. This study takes a single parameter, the shortness of the path, as the quantity to be optimized.

2.1. Particle Swarm Optimization
PSO is inspired by the behavior of swarms like a school of fish or a bird flock. For solving a problem, it creates several solutions called particles. These particles move over the search space based on their current position and velocity. The movement of these particles is influenced by two factors: the local best-known solution and the globally best-known solution. It forms a vector which decides where the particle would move from its current position. The equations which calculate the velocity and position of the particle are:

\[
v_i[n + 1] = v_i[n] + w_1 r_1 (P_{best} - p_i[n]) + w_2 r_2 (G_{best} - p_i[n]) \tag{1}
\]

\[
p_i[n + 1] = p_i[n] + v_i[n + 1] \tag{2}
\]

where \(w_1\) and \(w_2\) are the learning factors, \(r_1\) and \(r_2\) are two random numbers in the range \([0,1]\) and, \(P_{best}\) and \(G_{best}\) are the \(i^{th}\) particle’s best known solution and the globally best known solution respectively. \(p_i[n]\) is the current position of the particle and \(p_i[n + 1]\) is the next position of the particle. \(v_i[n]\) is the current velocity of the particle and \(v_i[n + 1]\) is the next velocity of the particle. Equation 1 calculates the velocity of the particle while (2) calculates the next iteration position.

For implementing PSO, we first define what a particle is. A particle is a solution to the problem at hand. In the path planning problem, this is the path taken by the robot. In this implementation, we select a few handle points between the start and destination. The handle points are then splined to form a path. The cubic spline function is used to create the path. The handle points which are solutions to the problem are represented as an array \([(x_1,y_1),(x_2,y_2),...,(x_n,y_n)]\). The algorithm works as follows:

- Step 1: Initialize the population size, number of iterations and learning factors.
- Step 2: Randomize the personal best solutions.
- Step 3: For each iteration, calculate the cost corresponding to each solution and update personal best. At the end of an iteration, find the global best solution and update the value.
- Step 4: The velocity vector and position vector for the next iteration is calculated.

2.2. Ant Colony Optimization
The Ant Colony Optimization algorithm utilizes agents who construct solutions by adding components to a state. Memory is kept of the transition from one solution component to the next. This simulates the pheromone information, which influences the actual ants. This study restricts the movement of the agent to the four cardinal directions. A pheromone value is associated with each component. Thus the more favorable paths have a higher pheromone value associated with it and are chosen more often. The probability that an ant \(k\) would move from node \(i\) to node \(j\), where \(j\) belongs to the points it can move to, is given by:

\[
p^k_{ij} = \begin{cases} 
(\phi_{ij})^\alpha (\omega_{ij})^\beta & \text{if } j \in N^k_i \\
0 & \text{if } j \notin N^k_i
\end{cases}
\]
The pheromone value is indicated by $\phi_{ij}$. $\omega_{ij}$ denotes the weight associated with the movement from node $i$ to node $j$. Here $N^k_i$ gives all the possible movements. $\alpha$ and $\beta$ are constants based on which more weight is given to pheromone value or heuristic information. The pheromone deposit also evaporates with time and this rate is given by:

$$\phi_{ij}[n + 1] \leftarrow (1 - \mu)\phi_{ij}[n]$$

where $\mu$ is the evaporation rate. The pheromone value is calculated for a path $k$ as:

$$\phi^k = \text{Minimum length} / \text{Cost}^k$$

where minimum length is the displacement between the two points. Cost is taken as path length of the $k^{th}$ path. The algorithm works as follows:

- Step 1: Set values for population size, number of iteration, evaporation rate.
- Step 2: In each iteration, the agents move along the map, constructing solutions. The costs are evaluated based on the path lengths. The pheromone values are updated.
- Step 3: At the end of each iteration, pheromone values evaporate to give more diverse solutions.

2.3. Hybrid Algorithm

This algorithm utilizes both ACO and PSO to improve pathfinding capability. Both algorithms have advantages and drawbacks when used separately. ACO can find solutions even in complex maps, but the length of the path is longer compared to the ones found by PSO. PSO, on the other hand, gives better solutions as compared to ACO in less complex maps. We first implement ACO to find a preliminary solution. This solution is a collection of nodes that traverse the map from the start to destination, avoiding obstacles. We sample this path to reduce the number of points such that it holds the necessary information but has a lower memory requirement. The sampled path is taken as the global best solution for the PSO algorithm. We then use the PSO algorithm to optimize the preliminary solution.

3. Results

The following results were obtained using the Matlab Mobile Robotics Simulation Toolbox.
Figure 4. Using PSO: Start:[1,1], Destination:[8,4]
Figure 5. Using ACO: Start:[1,1], Destination:[8,4]
Figure 6. Using Hybrid Algorithm: Start:[1,1], Destination:[8,4]

Figure 7. Using ACO: Start:[1,1], Destination:[8,2]
Figure 8. Using Hybrid Algorithm: Start:[1,1], Destination:[8,2]

Figure 9. Using ACO: Start:[1,1], Destination:[10,4]
Figure 10. Using Hybrid Algorithm: Start:[1,1], Destination:[10,4]
Table 1. Comparison of the algorithms

| Algorithm           | Start | Destination | Best Path Length | Time for completion in seconds |
|---------------------|-------|-------------|------------------|-------------------------------|
| PSO                 | [1,1] | [10,10]     | 12.999           | 57.3275                       |
| ACO                 | [1,1] | [10,10]     | 21.5             | 17.7848                       |
| Hybrid Algorithm    | [1,1] | [10,10]     | 12.9380          | 44.2860                       |
| PSO                 | [1,1] | [8,4]       | 14.3678          | 34.6654                       |
| ACO                 | [1,1] | [8,4]       | 21.5             | 60.70                         |
| Hybrid Algorithm    | [1,1] | [8,4]       | 11.995           | 58.8226                       |
| PSO                 | [1,1] | [8,2]       | -                | -                             |
| ACO                 | [1,1] | [8,2]       | 44.5             | 144.9239                      |
| Hybrid Algorithm    | [1,1] | [8,2]       | 13.9768          | 230.5174                      |
| PSO                 | [1,1] | [10,4]      | -                | -                             |
| ACO                 | [1,1] | [10,4]      | 27.5             | 76.1992                       |
| Hybrid Algorithm    | [1,1] | [10,4]      | 12.9477          | 76.6766                       |

The hybrid algorithm, as shown by the comparison study, gave better solutions in every setting. The hybrid algorithm took lesser time for completion compared to ACO and PSO in some cases, as the other two algorithms took a longer time to achieve the same quality of solutions. For implementing PSO in more complex maps, the random solutions generated initially should have a larger deviation. In some cases, PSO failed to generate valid solutions.

4. Conclusion
This study gives an insight into the viability of using ACO and PSO for pathfinding. A new algorithm, a hybrid of ACO and PSO, is used for generating better solutions in the given map. A further improvement in the proposed algorithm would be to reduce the number of samples taken for secondary optimization. The sample points would provide information about the obstacles in the path. Reducing the number of sample points would improve the efficiency of the algorithm in more complex maps. Simulating the algorithm in diverse environments would help to decide on the different parameters (learning coefficients, population size, number of iterations, etc.) to be chosen. Further research needs to be conducted on the usage of swarm intelligence algorithms for pathfinding.

Acknowledgments
The authors would like to thank Ms. Tressa Michael and the Department of Electronics and Communication, Rajagiri School of Engineering and Technology, Kakkanad, for the help provided.

References
[1] J. Kennedy, R. Eberhart, Particle Swarm Optimization, Proceedings of ICNN’95 - International Conference on Neural Networks, 27 Nov.-1 Dec. 1995, Networks. doi:10.1109/icnn.1995.488968
[2] Dorigo, M., Birattari, M., Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28–39. doi:10.1109/mci.2006.329691
[3] Thabit, S., Mohades, A. (2018). Multi-robot Path Planning Based on Multi-Objective Particle Swarm Optimization. IEEE Access, 1–1. doi:10.1109/access.2018.2886245
[4] Zhang, J.-H., Zhang, Y., Zhou, Y. (2018). Path Planning of Mobile Robot Based on Hybrid Multi-Objective Bare Bones Particle Swarm Optimization With Differential Evolution. IEEE Access, 6, 44542–44555. doi:10.1109/access.2018.2864188
[5] Rashid, R., Perumal, N., Elamvazuthi, I., Tageldeen, M. K., Khan, M. K. A. A., Parasuraman, S. (2016). Mobile robot path planning using Ant Colony Optimization. 2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA). doi:10.1109/roma.2016.7847836

[6] Brand, M., Masuda, M., Wehner, N., Xiao-Hua Yu. (2010). Ant Colony Optimization algorithm for robot path planning. 2010 International Conference On Computer Design and Applications. doi:10.1109/iccda.2010.5541300