Comparison between Meta-Heuristic Algorithms for Path Planning

Lidia Rocha$^{1[0000-0002-1018-9033]}$ and Kelen Vivaldini$^{1[0000-0001-5985-9635]}$

Department of Computer Science - Federal University of São Carlos, São Carlos/SP, Brazil (lidia@estudante.ufscar.br,vivaldinin@ufscar.br)

Abstract. Unmanned Aerial Vehicle (UAV) has been increasingly employed in several missions with a pre-defined path. Over the years, UAV has become necessary in complex environments, where it demands high computational cost and execution time for traditional algorithms. To solve this problem meta-heuristic algorithms are used. Meta-heuristics are generic algorithms to solve problems without having to describe each step until the result and search for the best possible answer in an acceptable computational time. The simulations are made in Python, with it, a statistical analyses was realized based on execution time and path length between algorithms Particle Swarm Optimization (PSO), Grey Wolf Optimization (GWO) and Glowworm Swarm Optimization (GSO). Despite the GWO returns the paths in a shorter time, the PSO showed better performance with similar execution time and shorter path length. However, the reliability of the algorithms will depend on the size of the environment. PSO is less reliable in large environments, while the GWO maintains the same reliability.

Keywords: Meta-Heuristic · Path Planning · Unmanned aerial vehicle.

Lidia Rocha is a MSc student with expected date of conclusion in July 2021, and Kelen Vivaldini is your advisor. This paper should be considered in WTDR.

1 INTRODUCTION

UAVs are being adopted as an important choice for application in several areas, surveillance [1], construction [2], environment monitoring [3], and others. Initially, the paths were defined by the pilots and the UAVs followed the points [4] . Over the years, technology has evolved and UAVs have become more autonomous, being able to determine their own path [5], and thus used to carry out missions that require high precision [6] or demand life risk [7]

To complete these missions, the UAV needs to move between two points without colliding with obstacles. In other words, path planning is necessary for locomotion of UAVs. Path planning is an essential area of robotics that aims to find a path for the robot to move from the starting node to the goal. UAVs have dynamic restrictions, so generated paths should respect them. To minimize UAV movement problems, paths can be formed from smooth curves [8].

Nowadays, the complexity of missions and environments has grown exponentially, being necessary to move around in various environments, such as jungles/mountains [9], labyrinths [10], and others. Some of these problems even have complexity non-deterministic polynomial-time (NP), that is, they are decision problems soluble in polynomial time by a Turing machine [11].

The meta-heuristic algorithms are general algorithmic adaptable to different optimization problems. Because they do not have complete mathematical knowledge about their behavior, these algorithms can be used to solve general problems. They were made to solve problems with little information. The importance of meta-heuristic methods comes from an optimal result, not necessarily the best, in a viable time, so it is widely used for optimization [12]. Meta-heuristic algorithms are one of the best techniques to optimize problems with NP-hardness [13].

An example of a meta-heuristic technique is Particle Swarm Optimization (PSO), which is based on the behavior of birds. [14] presents path planning for three-dimensional environments based on PSO. A chaos based logistic map is used to improve the particle initial distribution. Constant acceleration and maximum velocity are designed to adaptive linear-varying ones and a mutation strategy is implemented that undesired particles are replaced by those desired.

Another technique used is Grey Wolf Optimization (GWO), which is based on the hunting process and social hierarchy of grey wolves. [15] presents an overview of GWO with its mathematical modeling, mainly contributing to the area of path planning for vehicles for Autonomous Underwater Vehicle (AUV). The optimization performed by the GWO proved to be better both in execution time and in path length compared to Ant Colony Optimization (ACO).

The Glowworm Swarm Optimization (GSO) can also be used for path planning, which is based on the movement of glowworms, which always following other worms of the same species that have more luciferin. [8] presents an approach to path planning in three-dimensional environments using GSO and maintaining a safe distance from obstacles on the way. The algorithm was compared with various meta-heuristic techniques and with Dijkstra. Simulations obtained satisfactory results considering execution time and path length.

In general, these algorithms have shown good performance in unstructured real-world problems. In literature, PSO is showed as the faster algorithm but when is made the comparison is showed that GWO has a similar execution time. In some environments, GWO is better than PSO, as will be shown in this paper. In 3D GSO tends to return the better trajectory but the environments tested are structured, in this paper are tested unstructured environment in 2D. The comparison made in this paper studies the complexity of each algorithm, making it possible to predict what the behavior will be in different environments when analyzing the complexity of the algorithm, specifying the advantages and disadvantages of them.

The objective of this article is to present a comparative analysis of meta-heuristic algorithms for path planning in a 2D environment according to the metrics as average, variance, and standard deviation, applied to the execution time and path length. Besides, explaining which situation each algorithm would be best applied. Section 2 presents the methodology used, explaining the operation of each algorithm and its complexity. Section 3 describes the results of the simulation and makes a statistical analysis. Finally, the conclusion is presented in Section 4.

2 Methodology

First, we present how the path is formed from the cubic interpolation of splines with interpolation nodes, which are generated randomly. Then, we explain how the cost function of the meta-heuristics was defined. Lastly, the PSO, GWO, and GSO algorithms are explained how are used to optimize the generated path.

2.1 Smoothed Path

Cubic spline interpolation is an approximation technique that consists of dividing the range of interest into subintervals and interpolating based on cubic polynomials [16]. Thus, the path generated will be smoother, making it easier for the robot to move and make curves.

With $n+1$ points, each point is defined by x_i, y_i, where i is the order of the coordinates, without two equal points and $a = x_0 < x_1 < ... < x_n = b$. The $S(x)$ spline satisfies the following properties [17]:

1. On each subinterval $[x_{i-1}, x_i], S(x)$ is a polynominal of degree 3, where $i = 1, ..., n$
2. $S(x)$ is continuous in $[a, b]$ and has continues derivative in $[a, b]$ until degree 3

This method starts by selecting N interpolation nodes $(x'_1, y'_1), (x'_n, y'_n), ...,$ (x'_nN, y'_nN) between the start node and the objective, to determine the intervals. The spline nodes are used to obtain the m interpolation nodes $x_1, ..., x_m$ and $y_1, ..., y_m$ which when connected form a continuous path.

2.2 Cost Function

The cost function is based on the number of obstacles present in the environment and their distance from the generated path and the path length. The path length is defined from the euclidean distance, shown in Eq. 1.

$$L = \sum \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{1}$$

Then the euclidean distance between each obstacle and the path is calculated, defined with len in Eq. 2. This equation shows the overview of the cost function. Where r_o is the size of each obstacle and κ is the coefficient to avoid obstacles. If $1 - len/r_o$ is less than 0, it should be considered 0, as it means that the obstacles are far from the path, then there is no penalty.

$$cost = L * (1 + \kappa * (\sum mean(\frac{1 - len}{r_o}))) \tag{2}$$

2.3 Particle Swarm Optimization

PSO is based on the movement of a bird flock. The algorithm was created due to the interest in discovering the underlying rules that allowed a large number of birds to gather synchronously, suddenly changing direction, dispersing and regrouping, several times. The best simulation demonstrated that the movement is based on the speed of the nearest neighbor and randomness [18]. This algorithm can be see in Algorithm 3.

Algorithm 1 Particle Swarm Optimization

```
1: Initialize particle, generate random path and define as best path
2: for i in iterations do
3:     Define particle velocity as Eq. 3
4:     Limit particle velocity according maximum and minimum allowed
5:     Update path with the velocity
6:     Limit path with the environment size
7:     Define best path according lowest cost
8: end for
```

$$pV = w * pV + c1 * r1 * (bP - p) + c2 * r2 * (bP - p) \tag{3}$$

In Eq. 3 pV is the velocity of the previous particle, w is a value that decreases linearly from 2 to 0. This value represents the number of changes that can have each generation. That is, in the first generations the algorithm tends to make more sudden optimizations and across iterations become smoother. The value of $c1$ and $c2$ are acceleration constants and the variables $r1$ and $r2$ are two different random values, between 0 and 1. The value of bP is the coordinates of the best path and p are the coordinates of the last path.

2.4 Grey Wolf Optimization

This algorithm is inspired by the hunting process found in grey wolves. They prefer to live in small groups, with an average of 14 members, who follow a social hierarchy. The leaders are the alpha wolves, having the authority to carry out the group's decisions. Beta wolves are next, which helps the leader to make a decision. Next comes is the delta wolves, which dominate the omega wolves, which are the lowest in the ranking. In the GWO algorithm, alpha wolves represent the best solution, followed by beta and delta [15]. The algorithm is shown in Algorithm 2.

$$rV = (c1 * w * r1) * r2 \tag{4}$$

$$wP = |rV * bP(typeWolf) - p| \tag{5}$$

Algorithm 2 Grey Wolf Optimization

```
1: Initialize wolfs, generate random path and define as best path for each wolf
2: for i in iterations do
3:     Define alpha according Eq. 5 with best position of alpha
4:     Define beta according Eq. 5 with best position of beta
5:     Define delta according Eq. 5 with best position of delta
6:     Define as best path for each wolf
7:     Update path according average between wolfs
8: end for
```

In Eq. 4 $c1$ is a constant of the algorithm. The variables $r1$ and $r2$ are random values from 0 to 1. The value of p is the current path and bP is the best path for the wolf category being calculated.

2.5 Glowworm Swarm Optimization

GSO is developed based on the behavior of glowworms. The glowworms can change the intensity of bioluminescence. Each glowworm has a luminous pigment, called luciferin, which is modified according to its position, allowing the glowworm to shine at an intensity similar to that of the function being optimized. Each glowworm chooses the neighbor with the highest luciferin, using a probabilistic mechanism, and moves towards it. However, the glowworm can only follow a certain number of worms nearby and even a limited distance. Naturally, glowworms use your bioluminescence light to signal, and taxis toward, in direction for other worms [19]. This algorithm can be see in Algorithm 3.

Algorithm 3 Glowworm Optimization

```
 1: Initialize glowworm, generate random path and define as best path
 2: for i in iterations do
 3:     Update luciferin according Eq. 6
 4:     Define neighbors according Eq. 7
 5:     Define probability movement until each neighbor according Eq. 8
 6:     Choose the neighbor with higher probability to be followed
 7:     Update path according Eq. 9
 8:     Update neighbor range according Eq. 10
 9:     Define best path according lowest cost
10: end for
```

$$l = (1 - ld) * l + le * gC \tag{6}$$

$$N = all((dist < range)\&(l < l_v)) \tag{7}$$

$$pG = \frac{l_v - l}{|\sum l - (nV * l)|} \tag{8}$$

In Eq. 6 ld is the decay of brightness glowworm, l is the brightness of the previous glowworm, le is the enhancement of the brightness of the glowworm and gc is the path cost of the previous glowworm. In Eq. 7 $dist$ is the distance between the nodes of the path generated by the current glowworm and the neighbor being checked, $range$ is the range to see the neighbors, and l_v is the luciferin neighbor glowworm. In Eq. 8 nV is the number of neighbors used in the current iteration.

$$nP = p + s * (p_t - p)/norm(p_t - p) \tag{9}$$

$$range = min(rB, max(0.1, range + (\lambda * (kN - nV)))) \tag{10}$$

In Eq. 9 p is the current path, s is the maximum value that the new position may vary from the current and p_t is the path that the best neighbor is doing. In Eq. 10 rB is the range boundary of glowworms; λ is the value to go decreasing gradually the number of viable neighbors for each iteration, to decrease variability across generations; kN is the maximum amount of neighbors; and nV is the number of neighbors used in the current iteration.

2.6 Complexity Analysis

The cost function of the algorithms has time complexity $O(obs)$, obs being the obstacles present in the environment. The cost function is based only on the path length and the collision with obstacles.

PSO and GWO have time complexity $O(it * pop * O(obs) + pop)$, with it as the number of iterations and pop the population size. The part of $+pop$ is due to the need to initialize the population at the beginning of the algorithm. And $it * pop * O(obs)$ is because all individuals in the population need to be updated, according to the cost function, during all iterations. The main difference between both algorithms is that in GWO there are more instructions as it is necessary to optimize 3 types of wolves per iteration. The GSO has $O(it * pop * (pop + nvar + O(obs)) + pop)$, with $nvar$ being the number of interpolation nodes of spline. The complexity differs in $pop + nvar$ because during the iterations, in addition to calculating the cost function, it is necessary to update the range extension between the glowworms and check with all neighbors which will be the best to follow.

3 Simulations and analysis

This section was divided in simulations and a statistical analysis, to analyze the performance of meta-heuristics algorithms, the PSO, GWO, and GSO were implemented in two scenarios. Some parameters of them are shown in Table 1, according [20]. Each algorithm had a population of 15 elements, with 50 interactions, and was performed 100 times to determine the variance and standard deviation of algorithms for execution time and path length.

Table 1. Parameters Adopted in the Algorithms

Algorithms	Parameters
PSO	personal learning coefficient (c1) = 1.5
	global learning coefficient (c2) = 1.5
	inertia weight = 1.0
	inertia weight damping ratio = 0.98
GWO	c1 = 2
	w is a value that decreases linearly from 2 to 0
GSO	luciferin init = 25
	luciferin decay = 0.4
	luciferin enhancement = 0.6
	neighbor range = 20
	initial range = 5
	boundary range = 50.2
	$\beta = 0.5$

The map used in the first scenario has $10m$ x $10m$ and in the second scenario has $30m$ x $30m$, with 6 and 11 obstacles, respectively, of varying size. The nodes are represented by Cartesian coordinates (x, y), where x and y represent in the map. In the first scenario the trajectory the start and end node are (0, 0) and (10, 10), respectively. In the second scenario the trajectory the start and end node are (2, 2) and (27, 27), respectively. All of these scenarios are unstructured environments, ie, with randomly positioned obstacles that can break any large-scale formation, and thus lack this implicit guidance [21].

The simulations were implemented in Python language (version 3.6). The computer's CPU was a Intel® Core ^{TM} i7-7700M, 2.8GHz, memory is 8GB, and 4GB (NVIDIA GeForce GTX 1050) dedicated graphics memory.

3.1 Simulation

In Fig. 1 is showed the trajectories generated to unstructured environments with 10 and 30 meters, respectively. As we can observe the path generated by each algorithm have been smoothed with the splines to facilitate the mobility of UAVs, due to limited dynamics and aerodynamics [8].

With it, the UAVs can carry out the trajectory in less time without stop. The environments tested are unstructured, similar to forests [3] and hilly areas [6], where can be made missions of monitoring or rescue.

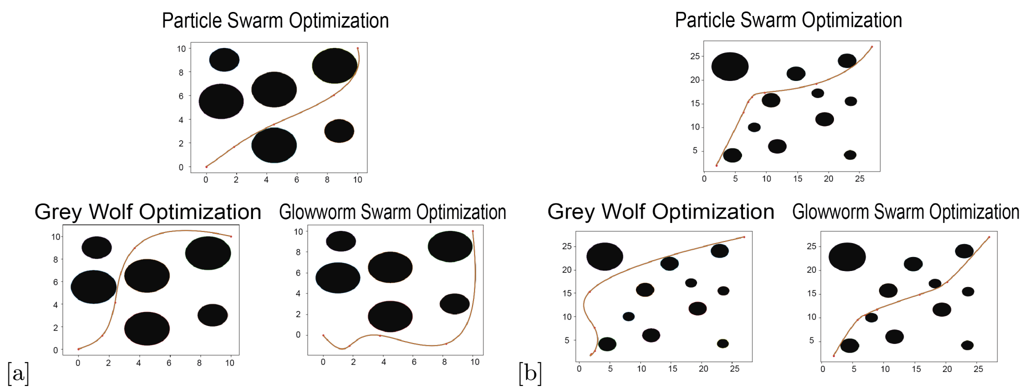

Fig. 1. Comparison between generated path - a) 10 meters b) 30 meters

3.2 Statistical Analysis

The metrics defined are average, variance, and standard deviation. Each algorithm executed 100 times. The metrics were applied to the execution time, that the algorithm took to calculate the best path, and to the path length. Tab. 2 shows the results of the algorithms, where the time is represented seconds and distance in meters.

Table 2. Statistical analysis of execution time and path length

	PSO 10m	GWO 10m	GSO 10m	PSO 30m	GWO 30m	GSO 30m
Average time	0.38	0.33	0.52	0.55	0.50	0.64
Average distance	18.01	24.47	28.36	43.01	47.76	56.99
Variance time	0.0009	0.001	0.005	0.0005	0.002	0.008
Variance distance	4.54	54.3	46.6	14.82	7.50	23.87
Standard deviation time	0.03	0.14	0.07	0.07	0.04	0.09
Standard deviation distance	2.13	7.36	15.7	12.17	8.66	15.45

Tab. 2 shows that GWO can find the path in less time, with low variance and standard deviation from time, which shows reliability in its execution time. However, PSO, on the other hand, managed to find the shortest path, also having low variance and with a small standard deviation, showing reliability in the results found. As seen in the Section 2.6, the GSO has a higher time complexity, which can be confirmed by analyzing the average time between the three algorithms.

Although the PSO did not obtain the shortest time, the algorithm can be considered as the one with the best performance because it obtained the shortest path length and similar time to GWO, which obtained the shortest time. Also, demonstrate the smallest variance and standard deviation concerning the path length and execution time.

These results are to environment with 10 meters. With 30 meters the results are similar, except the variance and standard deviation of distance that are very variable. In environment with 30 meters the minor variance and standard deviation were of GWO shortly thereafter of PSO. That is, when the environment increases, the PSO becomes less reliable and the GSO maintains the same reliability, concerning finding the best path.

According Tab. 2 the algorithms have small differences when the size of environment is changed, in relation to average path length and execution time. Thus, in small environments the PSO proved to be better for use in problems that demand reliability and need to operate in real-time. However, in large environments the GSO would be a better choose, which despite presenting a longer path, is still faster and more reliable.

4 Conclusion

Path planning is a complex problem with NP complexity. In unstructured environments, this complexity tends to increase even more and the algorithms can have an impracticable execution time with a high computational cost. Therefore, in this article, uses meta-heuristic techniques, which generalize the solution of the problem and tend to find an optimal result in an acceptable time.

Were made a comparison between PSO, GWO, and GSO according to metrics for execution time and path length for understand what is the better algorithm for each task. The GSO presented a longer execution time, which was already expected due to its high time complexity. GWO showed the best execution time among the algorithms, while PSO returned the shortest path between them. In small environments, PSO has a low standard deviation and variance in the execution time and path length being better to perform real-time tasks, but in large environments, the GWO is the best option, according to its reliability.

For future work, will be realized a study to prove how is the best size of population and number of interactions to result in the best results to these algorithms. Besides, to using three-dimensional, dynamic, and unknown scenarios. And will add a secure area to turn the trajectory more security and feasible to the UAV carry out.

Acknowledgment

The authors would like to acknowledge CAPES for financial support.

References

1. VandenBerg III, D.J., Swears, B.S.: Systems and methods for autonomous vehicle operator vigilance management (April 7 2020) US Patent 10,611,384.

2. Dutta, S., Cai, Y., Huang, L., Zheng, J.: Automatic re-planning of lifting paths for robotized tower cranes in dynamic bim environments. Automation in Construction **110** (2020) 102998
3. Vivaldini, K., Martinelli, T., Guizilini, V., Souza, J., Oliveira, M., Ramos, F., Wolf, D.: Uav route planning for active disease classification. Autonomous Robots **43** (07 2018)
4. Freimuth, H., König, M.: Planning and executing construction inspections with unmanned aerial vehicles. Automation in Construction **96** (2018) 540–553
5. Aggarwal, S., Kumar, N.: Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. Computer Communications **149** (2020) 270–299
6. Kulkarni, S., et. al: Uav aided search and rescue operation using reinforcement learning. Computer Science, Engineering (2020)
7. Puliti, S., Dash, J.P., Watt, M.S., Breidenbach, J., Pearse, G.D.: A comparison of uav laser scanning, photogrammetry and airborne laser scanning for precision inventory of small-forest properties. Forestry: An Int. Journal of Forest Research **93**(1) (2020) 150–162
8. Pandey, P., Shukla, A., Tiwari, R.: Three-dimensional path planning for unmanned aerial vehicles using glowworm swarm optimization algorithm. International Journal of System Assurance Engineering and Management **9**(4) (2018) 836–852
9. Zhao, H., Zhang, X., Yu, R., Li, R., Zhu, M.: Research on uav formation network technology in high dynamic environment. In: IEEE Int. Conf. on Virtual Reality and Intelligent Systems. (2019)
10. Mikhaylov, I., Kukhtiaeva, V.: Algorithm of autonomous uav orientation for applying in complex indoor environment. In: 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), IEEE (2017) 943–946
11. Goldreich, O.: P, NP, and NP-Completeness: The basics of computational complexity. Cambridge University Press (2010)
12. Osman, I.H., Laporte, G.: Metaheuristics: A bibliography (1996)
13. Rubio, F., Rodríguez, I.: Water-based metaheuristics: How water dynamics can help us to solve np-hard problems. Complexity **2019** (2019)
14. Shao, S., Peng, Y., He, C., Du, Y.: Efficient path planning for uav formation via comprehensively improved particle swarm optimization. ISA transactions **97** (2020) 415–430
15. Panda, M., Das, B.: Grey wolf optimizer and its applications: A survey. In: Proc. of Third Int. Conf. on Microelectronics, Computing and Communication Systems, Springer (2019) 179–194
16. McKinley, S., Levine, M.: Cubic spline interpolation. College of the Redwoods **45**(1) (1998) 1049–1060
17. Fritsch, F.N., Carlson, R.E.: Monotone piecewise cubic interpolation. SIAM Journal on Numerical Analysis **17**(2) (1980) 238–246
18. Eberhart, R., Kennedy, J.: Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks. Volume 4., Citeseer (1995) 1942–1948
19. Kaipa, K.N., Ghose, D.: Glowworm swarm optimization: theory, algorithms, and applications. Volume 698. Springer (2017)
20. Dewangan, R.K., Shukla, A., Godfrey, W.W.: Three dimensional path planning using grey wolf optimizer for uavs. Applied Intelligence **49**(6) (2019) 2201–2217
21. Francis, G., Ott, L., Marchant, R., Ramos, F.: Occupancy map building through bayesian exploration. The International Journal of Robotics Research **38**(7) (2019) 769–792