Unmanned Combat Aerial Vehicle Path Planning by Brain Storm Optimization Algorithm

Edin DOLICANIN[1], Irfan FETAHOVIC[1], Eva TUBA[2], Romana CAPOR-HROSIK[3], Milan TUBA[4]*

[1] Dept. of Technical Sciences, State University of Novi Pazar, Vuka Karadzica bb, Novi Pazar, 36300, Serbia
edin@np.ac.rs; ifetahovic@np.ac.rs

[2] Faculty of Informatics and Computing, Singidunum University, Danijelova 32, Belgrade, 11000, Serbia
etuba@ieee.org

[3] Inst. for Marine and Coastal Res., Univ. of Dubrovnik, Kneza Damjana Jude 12, Dubrovnik, 20000, Croatia
rcapor@unidu.hr

[4] Dept. of Mathematical Sci., State University of Novi Pazar, Vuka Karadzica bb, Novi Pazar, 36300, Serbia
tuba@ieee.org (**Corresponding author*)

Abstract: The use of the unmanned aerial vehicles is rapidly growing in ever wider range of applications where military use is among the oldest ones. One of the fundamental problems in the unmanned combat aerial vehicles control is the path planning problem that refers to establish the optimal route from the start position to the target, where optimality can be defined in numerous ways. Path planning represents a multi-objective constrained hard optimization problem. In this paper, we adjusted a recent swarm intelligence brain storm optimization algorithm for finding the unmanned combat aerial vehicle optimal path considering fuel consumption and safety degree. The proposed method was tested and compared to eleven different methods from literature. Based on the simulation results, it can be concluded that our proposed approach is robust, exhibits better performance in almost all cases and has potential for further improvements.

Keywords: Unmanned combat aerial vehicle, Path planning, Swarm intelligence, Brain storm optimization.

1. Introduction

The unmanned aerial vehicles (UAV) represent remotely piloted or self-piloted aircrafts. They have numerous advantages such as low cost, good maneuvering ability, high survival rate. Along with these qualities, UAV have the ability to carry different equipment or packages, e.g. cameras, sensors, weapons, etc. Initially, they were used for military purposes, as unmanned combat aerial vehicles (UCAV), to carry missiles and for drone strikes, but due to the mentioned qualities, UAVs are nowadays widely used for various applications including scientific [14], commercial [26], surveillance [7] and industry [27] applications.

UCAV are usually used in hostile environments, hence control systems for them are very challenging. The UCAV path planning problem is one of the most important parts of the autonomous control model for the UCAV. It refers to the problem of providing the optimal path from the starting point to the final destination, considering several objectives and constraints. Numerous research papers propose different methods for solving this problem. Path planning can be understood as a multi-objective constrained optimization problem that can include objectives like minimization of the fuel consumption, path length, exposure to threat, average altitude, path smoothness, etc.

Being a hard optimization problem, there are no deterministic methods for solving path planning in a reasonable time. Various techniques and methods can be used to tackle such NP-hard problems. In the last two decades, nature inspired algorithms, especially the swarm intelligence ones, have been widely and successfully used for finding acceptable solutions very quickly. Swarm intelligence algorithms are relatively new but powerful metaheuristics, where solutions for various problems of this kind are found by using swarms of simple agents. Each of these agents follows rather simple rules and they exchange gathered information between themselves. That communication enables swarms to exhibit remarkable intelligence. Particle swarm optimization (PSO) and ant colony optimization (ACO) are the oldest representatives of the swarm intelligence algorithms. After these two, numerous other swarm intelligence algorithms like fireworks [12], [19] and firefly algorithms [29], [21] or elephant herding optimization [22], have been proposed and successfully applied in various applications [20], [18], [1].

Numerous swarm intelligence algorithms and their modification were proposed for solving the UAV and UCAV path planning problem. However, recent brain storm optimization algorithm was not one of them. In this paper, brain storm optimization (BSO) algorithm is adjusted for solving the UCAV path planning problem. Quality of the proposed method is proven by comparing

https://doi.org/10.24846/v27i1y201802

the obtained results with other state-of-the-art algorithms from literature.

The rest of the paper is structured as follows. A brief literature review is given in Section 2. A mathematical model for the path planning problem used in this paper, as well as performance criteria, i.e. considered objectives, are described in Section 3. The brain storm optimization algorithm and our proposed adjustment are presented in Section 4. The simulation results along with a comparison with state-of-the-art methods are described in Section 5. At the end, the conclusion and suggestions for a possible future work are given in Section 6.

2. Literature review

Path planning, especially for the UAV and UCAV, represents an active research topic of great practical importance and numerous methods were proposed for solving it. In the recent years, most of the proposed methods are based on the nature inspired algorithms.

A method based on unsupervised neural network for mobile robot path planning was proposed in [3]. Mobile robot was supposed to collect objects in the search area, thus the problem was considered as the travelling salesman problem. Simulation results have shown that the proposed method finds a path close to the optimal one.

Obstacles avoidance algorithm for mobile robot control that combines intelligent bug algorithm and follow the gap method, named intelligent follows the gap method (IFGM), was proposed in [33]. The IFGM has the ability to dynamically adjust path and avoid U and H shape obstacles. Based on the simulation results, the proposed algorithm was efficient considering distance-time performance.

An algorithm for finding a good path for mechanical arm was proposed in [16]. Bayesian particles filter was used for inverse kinematics problem, while computational geometry was proposed for collision detection problem. Using an appropriate set of parameters, the proposed method was able to find good path in complex two-dimensional space environments.

In [15], a modified genetic algorithm named multi-frequency vibrational genetic algorithm was used for solving the UAV path planning problem. The proposed algorithm included new mutation application strategy that improved global and local search. By introducing Voronoi diagram concepts to the initial population phase, the number of needed generations was reduced. The method was applied to two different environments and the results were compared to three different version of the genetic algorithm. It has been shown that the proposed procedure reduced computational time.

An improved constrained differential evolution algorithm was proposed for finding the optimal feasible path in three-dimensional space for the UAV [31]. Several objectives were considered, such as maximum turning angle, climbing slope, terrain, forbidden flying areas, map and threat areas. This algorithm was combined with the level comparison method. Based on the simulation results, where two scenarios were considered, the proposed method achieved good performance considering the solution quality, robustness and the constraint handling ability.

A hybrid algorithm that combines differential evolution (DE) and quantum particle swarm optimization algorithm was proposed in [9] for UAV route planning on the sea. The method was compared to the standard genetic algorithm, differential evolution, particle swarm optimization and quantum PSO. The comparison results proved that the proposed hybrid algorithm is better in quality of solution, robustness and convergence speed.

Artificial bee colony algorithm (ABC) improved by a balance evolution strategy was proposed for unmanned combat aerial vehicles path planning problem in [11]. A balance evolution strategy was introduced to improve convergence by balancing local and global search. This method had better performance compared to the standard ABC and to other two modified ABC algorithms proposed for path planning in literature.

In [30], predator-prey pigeon inspired optimization algorithm was proposed for solving three-dimensional unmanned combat aerial vehicle path planning problem in a dynamic environment. Predator-prey model was used to improve the convergence speed in pigeon inspired algorithm (PIO). The proposed method proved to be better for the path planning than the original PSO, PIO and DE.

Path planning for unmanned combat aerial vehicles was solved by a modified firefly

algorithm in [24]. The modification was applied to the exchange of information between fireflies with better objective function values so the convergence speed was increased. The proposed algorithm was more effective for the path planning problem compared to the original firefly algorithm and to several other swarm intelligence algorithms such as ACO, DE, GA, PSO and others.

In [23] the bat algorithm was modified and adjusted for the UCAV path planning. Mutation operator for generating a new position of the bats in new generation was replaced in some cases by the differential evolution operator. Two objectives were considered for finding the optimal path, fuel consumption and threat avoidance. The proposed method was compared with the original bat algorithm and with other population based algorithms (ACO, PSO, DE, GA, etc.). Simulation results have shown that this method had better performance compared to other mentioned methods.

3. UCAV path mathematical model

Unmanned combat aerial vehicles path planning represents an active research topic and various path modellings were proposed. UCAV path planning problem can be described as an optimization problem where the goal is to find the optimal route from the start point (S) to the target (T), according to some metrics. Optimal path can be defined in numerous ways since different desirable features can be considered, for example the shortest or the smoothest path, the minimal fuel consumption, the safest path, etc. In this paper two different criteria were used, fuel consumption and safety degree.

Unmanned combat aerial vehicles are moving in three dimensional space, but in this paper altitude was not considered, which means that two dimensional space was used for the path planning problem. This simplification was also considered in other papers from the literature [23], [24].

For solution modelling, we adopted a method which is commonly used not only for the UCAV path planning [11], [13], but also for the robot path planning [28], [32]. The used model is rather simple, but efficient. The main idea is to ensure UCAV's moving towards the target position.

The path model used in this paper transforms the path planning problem into a D-dimensional optimization problem in the following way. The first step is to transform the coordinate system so that the x-axis is the line that connects the start and the target positions. Additionally, start point is translated to (0,0) which is accomplished by the following transformation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_s \\ y_s \end{bmatrix}, \quad (1)$$

where (x,y) represent the original coordinates, (x',y') are the new coordinates in the transformed coordinate system, (x_s,y_s) are coordinates of the start position S and ϕ is anti-clockwise angle from x-axis to the vector $\overrightarrow{ST}$:

$$\varphi = \arcsin \frac{|y_T - y_S|}{||\overrightarrow{ST}||} \quad (2)$$

Next, the path is modelled by dividing the new x-axis, the line S-T, into $n+1$ equal segments by n points. The x-coordinate of each path point is determined as shown in Figure 1.

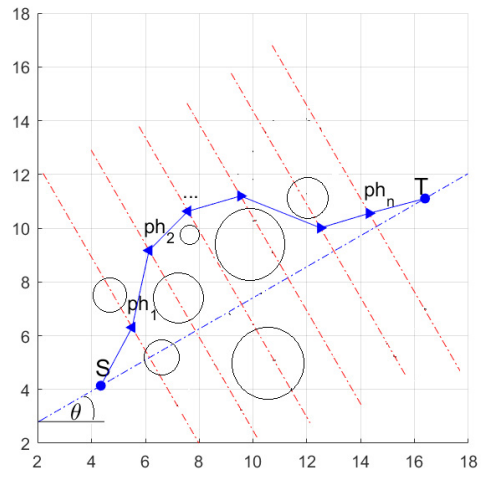

Figure 1. Path modelling

One path point is located on each red line presented in Figure 1. The y-coordinate for each point needs to be found for the optimal path. In Figure 1 path points are marked by ph_1, ph_2, …, ph_n and n determines the optimization problem dimensionality. The complete path of the UCAV is represented as:

$$PH = (S, ph_1, ph_2, \ldots, ph_n, T). \quad (3)$$

3.1 Performance criteria

The goal of the path planning methods is to find the optimal route for an UCAV where the criteria

for the optimality can be different. In this paper two criteria were considered: fuel consumption and safety degree.

The first criterion, fuel consumption is proportional to the path length so it can be represented as a function of the length. Total fuel consumption is defined as:

$$F_{fuel} = \int_0^L w_f \, dl, \tag{4}$$

where L is the path curve and w_f represents the fuel cost for each path point. In this paper we used w_f=1 so the fuel cost is equal to the path length which is the sum of distances between corresponding neighbor points in the path. If the start position S is denoted as ph_0 and the target position T as ph_{n+1}, the path length can be calculated as:

$$L(PH) = \sum_{i=0}^{n} d(ph_i, ph_{i+1}), \tag{5}$$

where $d(a,b)$ is the Euclidean distance between points a and b.

The second criterion considered in this paper is safety. UCAV during their flight can be faced with certain threats such as radars, radiation, missiles, antiaircraft artillery and similar. These threats can be represented as circle areas with different radii and threat degree or weight [24]. If the path goes through that danger areas, UCAV is exposed to the threat of a same degree. The longer the path passes through the danger area, the probability of UCAV being damaged or destroyed is larger. On the other hand, probability to be damaged if the path is outside of the threat circle is equal to 0.

Similarly to the fuel consumption, the threat cost can be calculated as:

$$F_{threat} = \int_0^L w_t \, dl, \tag{6}$$

where again L is the path curve and w_t represents the threat weight for each path point. In order to determine the total threat cost, it is necessary to include all the threats. Each threat has a circle range inside that circle where the threat degree is constant. The total threat cost produced by N_t threats while UCAV flies along the path segment L_{ij} is defined by the following equation [24]:

$$w_{t,L_{ij}} = \int_0^{L_{ij}} \sum_{k=1}^{N_t} \frac{t_k}{((x-x_k)^2+(y-y_k)^2)^2} dl, \tag{7}$$

where t_k represents the danger degree of the threat k with the center at (x_k,y_k). In order to simplify calculation of the threat or safety degree, Eq. 7 can be converted into discreet model. Each path segment can be divided into $m+1$ equal subsegments by m points and the threat degree of the path segment L_{ij} is represented by the average of the danger degrees in these m points. In this paper we used the same model presented in [11], [23] and [24] where each path segment was divided by 5 points thus the threat produced by N_t threats while UCAV flies along the path segment L_{ij} was calculated by the following equation:

$$w_{t,L_{ij}} = \frac{L_j^5}{5} \sum_{k=1}^{N_t} t_k \left(\frac{1}{d_{1,k}^4} + \frac{1}{d_{2,k}^4} + \frac{1}{d_{3,k}^4} + \frac{1}{d_{4,k}^4} + \frac{1}{d_{5,k}^4}\right), \tag{8}$$

where L_{ij} represents the length of the path segment between ph_i and ph_j, d_{mk} is the Euclidean distance between the m^{th} subsegment point of the segment L_{ij} and threat k, while t_k represents the threat level of the k^{th} threat.

4. BSO algorithm for the UCAV path planning

Brain storm optimization algorithm (BSO) is a swarm intelligence algorithm based on the human idea generation process which is known as the brainstorming process. BSO was proposed by Yuhui Shi in 2011 [17]. Since then the BSO was used in numerous applications like the support vector regression parameter selection [25], the energy optimization in grid systems [2], the wireless sensor deployment [6], and many others. Due to the extensive use of the BSO, various modifications [4], [5] and hybridizations [10] of the BSO were proposed.

The brainstorming process was simplified and generalized in order to implement the BSO algorithm. The simplification was made so that that brainstorming process still contains enough elements to implement optimization algorithm that has local and global search, i.e. exploitation and exploration, while the unimportant details were discarded.

In the brain storm optimization algorithm ideas represent the simple agents. Ideas are d-dimensional vectors. Brainstorming process starts by n initial random solutions and after this

initialization, ideas are clustered into *m* clusters. For clustering the solutions, k-means algorithm is usually used, but other algorithms can also be used. In each cluster the best idea, i.e. the solution with the best fitness function value is set to be the cluster's center.

The iteration process generates new solutions built by combining existing solutions. After that, new and old solutions are compared and the better ones are kept for the next iteration. Based on the algorithm's parameter p_{6b} one or two of the existing solutions can be used for generating a new one. With the probability p_{6b} one solution, $x_{selected}$, is selected and based on it the new solution x_{new} is calculated by the following equation:

$$x_{new} = x_{selected} + \varsigma \cdot n(\mu,\sigma), \quad (9)$$

where $n(\mu,\sigma)$ is a random number generated from Gaussian distribution with mean μ and variance σ, while ς is a coefficient that controls the influence of the Gaussian random value which is calculated in each generation by the following equation:

$$\varsigma = \log sig((0.5*maxIteration - currentIteration)/k)*rand() \quad (10)$$

where *maxIteration* and *currentIteration* are maximal number and the current number of iterations, respectively. Parameter *k* changes the logsig() function's slope, where logsig is a logarithmic sigmoid transfer function, while *rand()* represents random value from uniform distribution within (0,1).

On the other hand, with the probability *1-*p_{6b}, two solutions will be chosen and combined to generate a new one as the average of the selected two solutions.

Exploration is controlled by another algorithm's parameter, p_{5a} and is implemented by replacing the cluster center with a new random solution.

Besides the already mentioned parameters, the BSO procedure, described in the following Algorithm pseudo-code, has several more parameters that need to be determined. Parameter p_{6bi} refers to the cluster probability to be chosen for selecting the solution (idea) that will be used for generating a new solution and is proportional to the number of ideas in each cluster. Parameters p_{6bii} and p_{6c} are the probabilities of using cluster's center or random solution from the chosen clusters.

Algorithm Pseudo-code of the BSO algorithm

Initialization
Randomly generate *n* potential solutions.
repeat
  Cluster *n* solutions into *m* clusters.
  Rank solutions in each cluster and set the best one as cluster center.
  Randomly generate a value *r* between 0 and 1.
  if $r<p_{5a}$
    Randomly select a cluster center.
    Randomly generate an individual to replace the selected cluster.
  end if
  repeat
    Generate new solutions.
    Randomly generate a value r between 0 and 1.
    if $r<p_{6b}$
      Randomly select a cluster with probability p_{6bi}.
      Randomly generate a value r_1 between 0 and 1.
      if $r\_1<p_{6bii}$
        Select the cluster center and add random values to it in order to generate new individual.
      else
        Randomly select a solution from the chosen cluster and add random value to the solution to generate new one.
      end if
    else
      Randomly select two clusters.
      Generate random value r_2 between 0 and 1
      if $r_2<p_{6c}$
        Two cluster centers are combined to generate new solution.
      else
        Two solutions from each selected cluster are randomly chosen to be combined to generate new individual.
      end if
    end if
    The newly generated solution is compared with the same solution index and the better one is kept.
  until l n new solution is generated
until Maximal iteration number is reached.

In this paper we propose the BSO for unmanned combat aerial vehicle path planning problem. During the flight UCAV should avoid threat areas while minimizing the fuel consumption. By reducing it, the threat degree can be increased and vice versa: in order to avoid threat areas path length can be increased which results in larger fuel consumption. Since these two objectives cannot be minimized at the same time, we used parameter λ that determines the influence of each of them. The

final objective function that need to be minimized by brain storm optimization algorithm is:

$$F(PH) = \lambda F_{fuel} + (1-\lambda) F_{threat}. \quad (11)$$

5. Simulation results

Our proposed method for the UCAV path planning was implemented in *Matlab R2016a* and all the simulations were performed on the platform with Intel ® Core™ i7-3770K CPU at 4GHz, 8GB RAM, Windows 10 Professional OS.

In order to test the quality of our proposed method we compared it with the methods proposed in [24] and [23] and qualitatively with the one mentioned in [8]. In [24] a modified firefly algorithm (MFA) was used for solving unmanned combat aerial vehicle path planning problem. Modifications include adding Levy flight with dynamic parameter for generating new positions of the fireflies and adding information exchange between the fireflies with the best fitness function values. These modifications increased the local search and convergence speed. In [23] a bat algorithm with mutation (BAM) was proposed for the same problem as the one considered in this paper. A differential evolution mutation operator was added into the original bat algorithm with the aim to speed up the convergence of the algorithm. Besides the mutation operator, another change was to fix dynamic parameters of the bat algorithms.

In this paper we organized simulations in the same way as in [24] and [23] where one flight environment was considered. The parameter λ of the fitness function was set to 0.5, the start position was set to (10, 10) while the target point had the coordinates (55, 100). Five threat zones are known in the field. All information, i.e. coordinates, threat radii and threat grades are listed in Table 1.

In [24] and [23] the proposed methods were compared with numerous nature inspired algorithms such as the genetic algorithm (GA), the stud genetic algorithm (SGA), the differential evolution (DE), the evolutionary strategy (ES), the particle swarm optimization (PSO), the ant colony optimization (ACO), the original bat algorithm (BA) and the original firefly algorithm (FA). In this paper we included these simulation results which encompassed performance analysis in two different cases: when maximal number of generation was changed and when different dimensionality of the problem was considered. Then, the reported results were normalized in the same way as it was done in [24] and [23] and obtained fitness function values were reduced by 50.

Table 1. Information about threats in the field

No.	Location (km)	Threat radius (km)	Threat grade
1	(45,50)	10	2
2	(12,40)	10	10
3	(32,68)	8	1
4	(36,26)	12	2
5	(55,80)	9	3

Parameters for the brain storm optimization algorithm were set by conducting preliminary computational experiments. The probability for generating a new random solution p_{5a} was set to 0.2 and the parameter for selecting one cluster p_{6b} was 0.8. In the case that one cluster was chosen, probability p_{6bii} of using its center was 0.4, which was the same as the probability p_{6c} of combining cluster centers. Probability p_{6bi} referred to the probability of the cluster being chosen and it was proportional to the number of individuals in it.

The first set of simulations included the changing of maximal number of fitness function evaluations. For these experiments, the dimension was set to 20, the maximal number of fitness function evaluation were 1500, 3000, 4500, 6000 and 7500, the same as in [24] and [23] and the population size for the BSO was 20. The comparison of the results for the best, average and the worst fitness function values obtained in 100 independent runs by methods from [24], [23] and our proposed method is presented in Table 2.

Our proposed algorithm found the best solution in all cases, except for the 3000 objective function evaluations. The bat algorithm with mutation found the best path for the UCAV for that case. Good results were also achieved by MFA, FA and DE, but with a higher number of objective function evaluations. The other algorithms were inferior. Our proposed algorithm showed constant improvements with the increase in maximal number of generations.

It is very important that when other algorithms started to stagnate, BSO made significant improvement by increasing the number of objective function evaluations to 7500.

Table 2. The best, mean and the worst normalized fitness function values obtained in 100 runs for different maximal number of generations

Ev. No.		GA	SGA	DE	ES	PSO	ACO	FA	MFA	BA	BAM	BSO
1500	Best	1.2604	1.7370	2.4179	9.6276	2.7827	10.7202	1.4713	0.7030	4.0662	0.6208	**0.6064**
	Meant	4.2541	4.5491	12.3797	20.5653	10.076	16.3819	6.2034	1.9576	16.6782	**1.4842**	2.7838
	Worst	10.2501	13.0102	25.3999	4109676	28.6115	18.7099	28.0425	**4.6726**	39.0832	11.7494	8.7150
3000	Best	1.5073	1.3218	0.8503	10.6318	2.3469	10.8912	0.6577	0.5382	4.7582	**0.4900**	0.5314
	Meant	3.5523	3.4353	6.0887	20.6706	9.1725	16.2884	4.3526	1.3048	14.9048	**0.9337**	1.2222
	Worst	8.2047	11.0529	18.6288	38.5875	25.7065	18.4316	29.3022	4.5749	29.9962	9.7666	**3.9941**
4500	Best	1.0991	1.1559	0.5319	11.1469	2.3738	9.9096	0.5459	0.4857	4.1112	0.4724	**0.4112**
	Meant	3.4269	3.1636	3.7267	20.1996	9.5459	16.1408	4.1809	0.9933	14.4874	0.9123	**0.9062**
	Worst	10.5257	13.3517	13.8150	46.0828	29.6341	17.4223	27.8480	4.9631	31.1293	7.6952	**2.9817**
6000	Best	1.0792	0.7595	0.5047	11.2403	3.4276	12.3080	0.4931	0.4661	3.1463	0.4590	**0.4541**
	Meant	3.0080	2.6434	2.6358	20.8610	8.9917	16.3976	2.2791	0.8984	12.4323	0.8000	**0.7533**
	Worst	6.7466	7.5385	10.4226	31.3944	33.0709	17.214	26.5768	9.1502	24.9732	6.7334	**2.1708**
7500	Best	1.0640	1.0166	0.4792	12.3745	2.5221	7.1358	0.4753	0.4508	4.4072	0.4636	**0.3744**
	Meant	2.9160	3.1409	1.9715	20.7600	7.8005	16.1958	2.2064	0.7025	11.4213	0.7422	**0.7014**
	Worst	8.9162	13.5830	8.9560	34.8908	27.3858	16.9896	26.3005	3.6783	24.7175	**3.3564**	3.4103

Average of fitness function values obtained in 100 runs are also presented in Table 2. Our proposed BSO algorithm had the best mean values for 4500, 6000 and 7500 fitness function evaluations, while for the 1500 and 3000 BAM obtained better results. BSO needed more generation to find the optimal solution, i.e. it had slower convergence but excellent ability of finding better solution.

Based on the worst fitness function values obtained in 100 runs presented in Table 2 it can be concluded that our proposed method exhibited relatively good performance in all cases. The worst solution was found by 1500 fitness function evaluations and it was 8.7150 but when the number of fitness function evaluation had been increased to 3000, the worst solution was significantly improved and it was 3.9941. All other algorithms, except MFA for 1500 and BAM for 7500 evaluations, had larger difference between the best and the worst solution which proves the robustness of our proposed BSO method.

Standard deviation which would reveal the true robustness of the algorithms was not reported in [24] and [23]. Without that data we can just make a rough conclusion based on the best, the worst and mean results. Since our proposed algorithm had the smaller difference between the best and the worst solutions, it can be concluded that compared to the other mentioned nature inspired algorithms, our proposed BSO algorithm was the most stable one.

The second set of experiments included testing the algorithm with different problem dimensions. Maximal number of fitness function evaluations was set to 6000 in order to make it comparable with other mentioned algorithms. Algorithms were tested for the dimensions 5, 10, 15, 20, 25, 30, 35 and 40. The results obtained are presented in Table 3.

Our proposed method found best solutions for the dimensions smaller than 35, i.e. dimensions 5, 10, 15, 20, 25 and 30. When the problem dimensions was set to 35 and 40, the MFA achieved the best solutions. However, our proposed BSO algorithm outperformed even the MFS when the maximal number of fitness function evaluations was increased to 10,000. In that case, the proposed BSO algorithm found the best solutions 0.4418 and 0.4502, for dimensions 35 and 40, respectively (the mean solutions were 1.1495 for dimension 35 and 1.4210 for dimension 40). This is the consequence of the nature of the BSO algorithm which has the ability to work very well with multi-objective and large optimization problems, but it needs certain number of iterations which does not necessarily means to increase in computational time.

On average, in 100 runs our proposed method was better for all the problem dimensions except in the case of the problem dimension 40 where the bat algorithm with modification had better performance. When for that problem dimension the number of fitness function evaluations was increased to 7000, the proposed BSO algorithm found better solution of 1.6892 which again showed its ability to improve when other algorithms stagnate.

Table 3. The best, mean and the worst results obtained in 100 runs with different problem dimensions and 6000 fitness function evaluations

D		GA	SGA	DE	ES	PSO	ACO	FA	MFA	BA	BAM	BSO
5	Best	5.2471	9.9596	4.3568	12.3746	5.6082	10.1164	4.3585	4.3573	10.6909	4.3575	**0.3843**
	Mean	10.5709	10.8836	8.0557	31.8202	10.0765	11.4856	8.7499	9.1673	56.4830	9.0542	**0.3856**
	Worst	20.1888	22.6326	9.7959	62.1765	13.3267	12.6928	15.7395	12.4186	295.2557	10.2403	**0.3873**
10	Best	1.5716	1.5498	1.3952	8.0656	2.1101	7.4746	1.3990	1.3966	2.3600	1.3953	**0.3690**
	Mean	2.3722	2.2813	3.1206	27.2252	7.2212	12.5333	2.1801	1.5740	19.4251	2.7075	**0.4304**
	Worst	6.3799	5.7899	12.4821	74.6665	23.2604	18.2565	6.7095	3.7858	58.7386	10.7242	**0.4372**
15	Best	0.8299	0.9700	0.6204	7.7408	3.2257	9.8297	0.6172	0.6115	3.0757	0.6094	**0.3774**
	Mean	2.1136	1.8973	2.3737	22.0792	7.7362	10.2484	2.8217	0.8967	13.6018	1.2318	**0.4305**
	Worst	8.1499	9.9385	12.5250	50.3214	28.0228	10.9917	44.2763	3.8319	35.7454	10.1928	**0.6771**
20	Best	0.8600	0.8426	0.4913	9.6276	2.3738	10.0836	0.4626	0.4552	2.3950	0.4679	**0.4541**
	Mean	2.9612	2.8621	3.0044	20.4717	9.9091	16.3303	3.7327	0.7004	13.6305	0.7609	**0.7003**
	Worst	9.4820	11.6024	18.8897	38.7234	34.7133	17.0266	28.9142	2.0279	33.7068	3.7420	**2.0108**
25	Best	1.5243	1.3743	0.6265	12.3169	2.3740	11.5490	0.4908	0.4571	5.0173	0.4484	**0.3929**
	Mean	3.7244	3.7238	4.6029	22.7244	10.3315	11.5842	3.9039	0.9987	14.9017	0.7093	**0.6851**
	Worst	12.7971	16.0736	17.1415	33.4598	31.6741	12.2373	16.4518	3.7043	24.9265	**3.5192**	4.0914
30	Best	1.7026	1.5147	1.1301	18.0090	3.6751	13.8615	0.6828	0.5160	7.2470	0.4671	**0.4411**
	Mean	5.3097	4.3798	11.4103	25.4016	12.7964	13.9422	4.9621	1.3568	16.6162	1.1067	**1.0530**
	Worst	22.1291	14.0512	29.6529	37.4566	35.6656	14.4647	15.9757	8.3364	30.0844	10.2851	**8.0677**
35	Best	2.1602	1.5319	1.2849	16.8613	5.4765	16.9476	1.0829	**0.4709**	7.4484	0.4795	0.5979
	Mean	6.0765	5.4943	19.1074	27.2172	13.8799	18.3452	5.9955	1.6009	17.7033	1.4617	**1.4535**
	Worst	24.4790	15.6693	39.4435	46.6475	38.0578	18.7271	33.8871	**5.8830**	32.7374	8.8193	8.2009
40	Best	2.4178	1.9406	3.9617	19.8244	5.5384	17.6142	1.5225	**0.4506**	8.6500	0.6028	0.6003
	Mean	7.6989	7.4237	28.7062	30.0177	15.1555	24.7642	7.8558	2.1978	19.9737	**1.8769**	2.0039
	Worst	19.2098	22.5022	45.4130	44.3624	35.5090	27.0641	36.6626	**7.7236**	33.2634	8.4273	8.2159

That result proves the quality of the proposed brain storm optimization algorithm for the UCAV path planning problem even for the larger dimensions. In these experiments, we fixed the number of objective function evaluations in order to fairly compare the results but it is well known that for larger problem dimensions more iterations are needed.

In most cases, our proposed algorithm had the smallest worst solution in 100 runs (see Table 3). All algorithms except MFA, BAM and our BSO have very large worst solutions. For the dimensions 5, 10 and 15, the proposed BSO algorithm had significantly smaller worst solutions compared to BAM, while in other cases the difference was not so substantial. Compared to the MFA, the proposed BSO had significantly smaller worst solution just for the dimension 5, while for the dimension 35 and 40, the MFA obtained better worst solutions. In other cases, worst solutions were similar.

Additionally, we qualitatively compared our proposed BSO method with the method proposed in [8] where the artificial neural network (ANN) trained by imperialist competitive algorithm (ICA) was used for the UCAV path planning. That method was tested for two different environments and compared with the artificial bee colony algorithm (ABC). Simulation results showed that the proposed ICA-ANN algorithm was superior to the ABC algorithm. Optimal path was defined by the same fitness function as in this paper.

Two test environments were used in [8] with 7 and 8 danger areas, but without exact coordinates reported so we approximated them from the picture. In both cases our proposed BSO algorithm found path as straight as possible while the ICA-ANN algorithm had some unnecessary turns and arcs during the avoidance of the danger zones.

6. Conclusion

In this paper we adjusted the brain storm optimization algorithm for unmanned combat aerial vehicle path planning problem. The fuel consumption and safety of the UCAV were considered as criteria for the path optimality. The proposed method was experimented on test environments from literature with circular danger zones and different threat degrees and was compared to eleven other methods from literature. Based on the simulation results, it can be concluded that the proposed brain storm optimization algorithm exhibited very promising

features. It had better performance for smaller problem dimensions, while for larger problem dimensions more iterations were needed. However, the results were further improved compared to other algorithms. A future work can include hybridization or modification of the brain storm optimization algorithm in order to improve the convergence speed and to adjust it for larger dimensional problems. The third dimension, i.e. UCAV altitude, can also be included. The BSO exhibited robust and superior performance in the tested cases. Yet, its potential is even greater considering its unique features of clustering results that can be exploited for more complex multi-objective formulations of the UCAV path planning problem.

Acknowledgements

This research was supported by the Ministry of Education, Science and Technological Development of Republic of Serbia, Grant No. III-44006.

REFERENCES

1. Alihodzic, A., Tuba, E., Capor-Hrosik, R., Dolicanin, E. & Tuba, M. (2017). Unmanned aerial vehicle path planning problem by adjusted elephant herding optimization. In *IEEE 25th Telecommunications Forum* (*TELFOR*) (pp. 804-807).

2. Arsuaga-Rios, M. & Vega-Rodriguez, A. (2015). Multi-objective energy optimization in grid systems from a brain storming strategy, *Soft Computing*, *19*, 3159-3172.

3. Brassai, S. T., Iantovics, B. & Enachescu, C. (2012). Optimization of Robotic Mobile Agent Navigation, *Studies in Informatics and Control*, *21*(4), 403-412.

4. Cao, Z., Shi, Y., Rong, X., Liu, B., Du, Z. & Yang, B. (2015). Random grouping brain storm optimization algorithm with a new dynamically changing step size, *Advances in Swarm and Computational Intelligence*, *LNCS*, *9140*, 357-364.

5. Chen, J., Cheng, S., Chen, Y., Xie, Y. & Shi, Y. (2015). Enhanced brain storm optimization algorithm for wireless sensor networks deployment, *Advance in Swarm and Computational Intelligence*, *LNCS*, *9140*, 373-381.

6. Chen, J., Wang, J., Cheng, S. & Shi, Y. (2016). Brain storm optimization with agglomerative hierarchical clustering analysis, *Advances in Swarm Intelligence*, *LNCS*, *9713*, 115-122.

7. Darrah, M., Wilhelm, J., Munasinghe, T., Duling, K., Yokum, S., Sorton, E., Rojas, J. & Wathen, M. (2015). A flexible genetic algorithm system for multi-UAV surveillance: Algorithm and flight testing, *Unmanned Systems*, *03*(01), 49-62.

8. Duan, H. & Huang, L. (2014). Imperialist competitive algorithm optimized artificial neural networks for UCAV global path planning, *Neurocomputing*, *125*, 166-171.

9. Fu, Y., Ding, M., Zhou, C. & Hu, H. (2013). Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *43*(6), 1451-1465.

10. Jia, Z., Duan, H. & Shi, Y. (2016). Hybrid brain storm optimisation and simulated annealing algorithm for continuous optimisation problems, *International Journal of Bio-Inspired Computation*, *8*(3), 109-121.

11. Li, B., Gong, L.-G. & Yang, W.-L. (2014). An improved artificial bee colony algorithm based on balance-evolution strategy for unmanned combat aerial vehicle path planning, Article ID 232704, *The Scientific World Journal*, 1-10.

12. Li, J., Zheng, S. & Tan, Y. (2017). The effect of information utilization: Introducing a novel guiding spark in the fireworks algorithm, *IEEE Transactions on Evolutionary Computation*, *21*(1), 153-166.

13. Liu, Y., Zhang, X., Guan, X. & Delahaye, D. (2016). Adaptive sensitivity decision based path planning algorithm for unmanned aerial vehicle with improved particle swarm optimization, *Aerospace Science and Technology*, *58*, 92-102.

14. Nex, F. & Remondino, F. (2014). UAV for 3D mapping applications: A review, *Applied Geomatics*, *6*(1), 1-15.

15. Pehlivanoglu, Y. V. (2012). A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV, *Aerospace Science and Technology*, *16*(1), 47-55.

16. Reyes-Amaro, A., Mesejo-Chiong, A., Mas-Sanso, R. & Jaume-I-Capo, A. (2013). Using particle filters to findfree obstacle trajectories for a kinematic chain, *Studies in Informatics and Control*, *22*(2), 185-194.

17. Shi, Y. (2011). Brain storm optimization algorithm, *Advances in Swarm Intelligence, LNCS*, *6728*, 303-309. Springer.

18. Tuba, E., Alihodzic, A. & Tuba, M. (2017). Multilevel image thresholding using elephant herding optimization algorithm. In *14th IEEE International Conference on Engineering of Modern Electric Systems* (*EMES*) (pp. 240-243).

19. Tuba, E., Tuba, M. & Beko, M. (2016). Support vector machine parameters optimization by enhanced fireworks algorithm, *Advances in Swarm Intelligence*, *LNCS*, *9712*, 526-534.

20. Tuba, E., Tuba, M. & Dolicanin, E. (2017). Adjusted fireworks algorithm applied to retinal image registration, *Studies in Informatics and Control*, *26*(1), 33-42.

21. Tuba, M., & Bacanin, N. (2014). Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems, *Neurocomputing*, *143*, 197-207.

22. Wang, G., Deb, S., Gao, X. Z. & Coelho, L. D. S. (2017). A new metaheuristic optimisation algorithm motivated by elephant herding behaviour, *Intern. Journal of Bio-Inspired Computation*, *8*(6), 394-409.

23. Wang, G., Guo, L., Duan, H., Liu, L. & Wang, H. (2012). A bat algorithm with mutation for UCAV path planning, Article ID 418946, *The Scientific World Journal*, 1-15.

24. Wang, G., Guo, L., Duan, H., Liu, L. & Wang, H. (2012). A modified firefly algorithm for UCAV path planning, *International Journal of Hybrid Information Technology*, *5*(3), 123-144.

25. Wang, J., Hou, R., Wang, C. & Shen, L. (2016). Improved v-support vector regression model based on variable selection and brain storm optimization for stock price forecasting, *Applied Soft Computing*, *49*, 164-178.

26. Whitehead, K., Hugenholtz, C. H., Myshak, S., Brown, O., LeClair, A., Tamminga, A.Barchyn, T. E., Moorman, B. & Eaton, B. (2014). Remote sensing of the environment with small unmanned aircraft systems (UASs), part 2: Scientific and commercial applications, *Journal of unmanned vehicle systems*, *2*(3), 86-102.

27. Wu, K. J., Gregory, T. S., Moore, J., Hooper, B., Lewis, D. & Tse, Z. T. H. (2017). Development of an indoor guidance system for unmanned aerial vehicles with power industry applications, *IET Radar, Sonar & Navigation*, *11*(1), 212-218.

28. Wu, Z., Fu, W., Xue, R. & Wang, W. (2016). A Novel Global Path Planning Method for Mobile Robots Based on Teaching-Learning-Based Optimization, *Information*, *7*(3), 39-49.

29. Yang, X.-S. (2009). Firefly algorithms for multimodal optimization, *Stochastic Algorithms: Foundations and Applications*, *LNCS*, *5792*, 169-178.

30. Zhang, B. & Duan, H. (2017). Three-dimensional path planning for uninhabited combat aerial vehicle based on predator-prey pigeon-inspired optimization in dynamic environment, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (*TCBB*), *14*(1), 97-107.

31. Zhang, X. & Duan, H. (2015). An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning, *Applied Soft Computing*, *26*, 270-284.

32. Zhang, Y., Gong, D. W. & Zhang, J. H. (2013). Robot path planning in uncertain environment using multi-objective particle swarm optimization, *Neurocomputing*, *103*, 172-185.

33. Zohaib, M., Pasha, S. M., Javaid, N., Salaam, A. & Iqbal, J. (2014). An improved algorithm for collision avoidance in environments having U and H shaped obstacles, *Studies in Informatics and Control*, *23*(1), 97-106.