Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-22 (2018) 292–297

Path planner methods for UAVs in real environment

V. Jeauneau*. L. Jouanneau**
A. Kotenkoff***

*Command and Control&Mission Planning Department, MBDA, Le Plessis-Robinson, France (e-mail: vincent.jeauneau@mbda-systems.com).
** Command and Control&Mission Planning Department, MBDA, Le Plessis-Robinson, France (e-mail: laurent.jouanneau@mbda-systems.com)
*** Command and Control&Mission Planning Department, MBDA, Le Plessis-Robinson, France (e-mail: alexandre.kotenkoff@mbda-systems.com)}

Abstract: This paper presents two methods to plan the path of an unmanned aerial vehicle (UAV) in a real 3D environment. Each method has its own purpose: one method is off-line and must allow an external operator to choose a trajectory to fly before the takeoff of the UAV; the other is on-line and must allow an external operator to modify the UAV trajectory in real-time during the fly. This last method is based on the well-known A-Star algorithm whereas the first one is a genetic algorithm (GA). The method based on A-Star provides a single path and the GA provides multiples trajectories using a Pareto front (PF). The paths produced have to satisfy the dynamic properties of the vehicle. Our methods embed the 2D dynamic properties of the vehicle and computes 2D trajectories. The third missing dimension is then recomputed using a recursive algorithm. The characteristics of the optimal path are represented in the form of multi-objectives. We achieve real-time capability for our A-Star based method with an average computation time of 836 ms and we achieve performance and flexibility for our GA.

Keywords: Genetic Algorithm (GA), A-Star, path planning, multi-objectives environment, unmanned aerial vehicles (UAVs).

1. INTRODUCTION

This research is motivated by developing an efficient operational system for UAVs. One key element of such a system is to have autonomous path planning method. It allows to autonomously compute a trajectory from a start point (TSP) to an end point (TEP) in 3D rough terrain environment. LaValle (2006) proposes an extensive overview on path and motion planning techniques in general whereas Goerzen et al. (2009) wrote a specific survey on motion planning for UAVs.

In the past, the quality of the trajectories was associated to the shortest path and deterministic search algorithms, such as A-Star proposed by Hart et al. (1968), were used to find it. Now, an optimal path is defined according to multiple objectives such as trajectory length, average altitude, radar exposure, GPS jammer exposure, fuel consumption, etc. In the literature, UAVs path planning problems have attracted a large variety of optimization algorithms. Szczerba et al. (2000) have developed an algorithm to solve the shortest path UAVs problem based on A-Star. Mittal and Deb (2007) have implemented a multi-objectives version of a GA in which a trade-off has to be done between flying time and safety. To do so, a two dimensional Pareto front (PF) is used to provide a suitable set of solutions at an external operator. Marachet et al. (2010) have used Bezier curves with a GA in order to compute trajectories that better consider the dynamic properties of the UAV. Volkan (2012) has proposed to use a vibrational genetic algorithm to improve the exploration and to avoid local minima. Roberge et al. (2013) have presented a generic cost function for the multi-objectives UAVs problem, they have developed a framework to parallelize their algorithms and they made a comparison between a GA and a particle swarm optimization (PSO). LaValle and Huffner Jr (2001) have developed a method to plan the trajectories of satellites based on the rapidly exploring random trees algorithm (RRT).

In this paper, we use two methods to develop an operational system for fixed wing UAVs. One method is off-line and must allow to select a trajectory before the take off. The other method is on-line and must allow to modify the trajectory in real-time during the fly. We decide to develop a GA for the first method and to base the second one on A-Star. Both algorithms compute a 3D path but contrary to the literature, our algorithms start by building a 2D path and then by computing the third missing dimension.

The remainder of the paper is organized as follow: section 2 provides details of the environment. Section 3 explains how to generate the third dimension from a 2D trajectory. Section 4 introduces the GA. Section 5 presents the method based on A-Star. Finally, experimental results are done in section 6.

2. ENVIRONMENT DETALS

The environment is represented with digital terrain elevation data (DTED). This representation is a standard which consists of a matrix of terrain elevation values (see Fig. 1). In our environment a radar is defined with a position, a height above the ground at which it is located and a range. An UAV is under radar exposure if it is in range of the radar and if it is in the line of sigh of the radar (see Fig. 2). The environment also allows us to define a GPS jammer with a position, a range defining the jamming heart and a range defining the jamming lock. Inside the jamming heart the GPS is down whereas inside the jamming lock the GPS stays down if it was down and up if it was up (see Fig. 3).

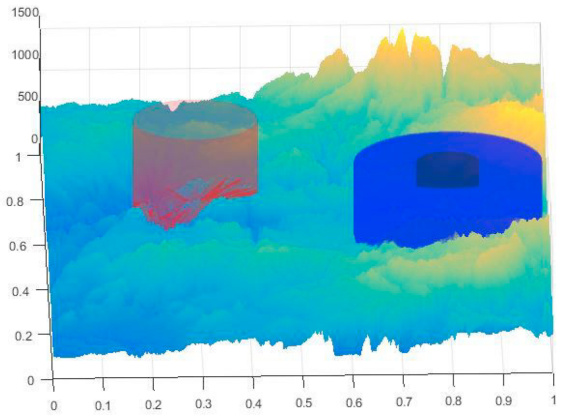

Fig. 1. DTED representation of the 3D environment with a radar in red (on the left) and a GPS jammer in blue (on the right).

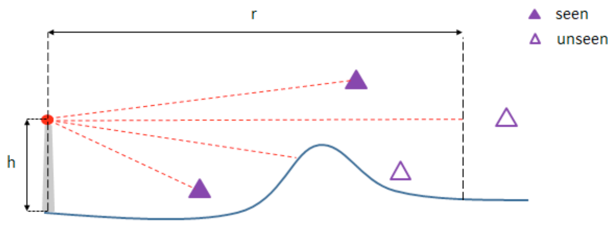

Fig. 2. Radar exposure representation

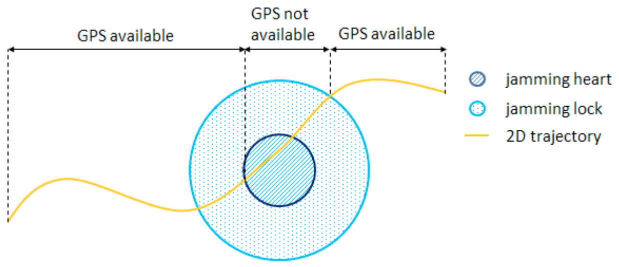

Fig. 3. 2D GPS jammer representation

3. THIRD DIMENSION COMPUTING

As mentioned before, our algorithms build only 2D trajectories and then compute the missing third dimension. To do so, we apply mathematical formulas (1) corresponding to the dynamic properties of the UAV in terms of slopes and vertical curvatures along the 2D trajectory. Where, z(i) is the UAV altitude at the ith point of the trajectory, where $slope^-$ (resp. $slope^+$) is the minimal negative (resp. maximal positive) slope than the UAV can fly, where $vcurv^-$ (resp. $vcurv^+$) is the minimal negative (resp. maximal positive) vertical curvature that the UAV can fly and where cst(i) is a constant upper which the UAV has to fly at the ith point of the trajectory. Usually cst(i) is used to define a safety height above the ground at any point of the trajectory and is used to define TSP and TEP altitudes. In formulas (1), we note that updating z(i) might have an impact on the altitude of points i-2 to i+2. Therefore, we have to apply these formulas recursively along the points of the trajectory until stability is reach.

$$z(i) = \max \begin{cases} cst(i) \\ z(i-1) + slope^- \times \Delta s \\ z(i+1) - slope^+ \times \Delta s \\ 2 \times z(i-1) - z(i-2) + vcurv^- \times \Delta s^2 \\ 2 \times z(i+1) - z(i+2) + vcurv^- \times \Delta s^2 \\ \dfrac{z(i-1) + z(i+1) - vcurv^+ \times \Delta s^2}{2} \end{cases} \tag{1}$$

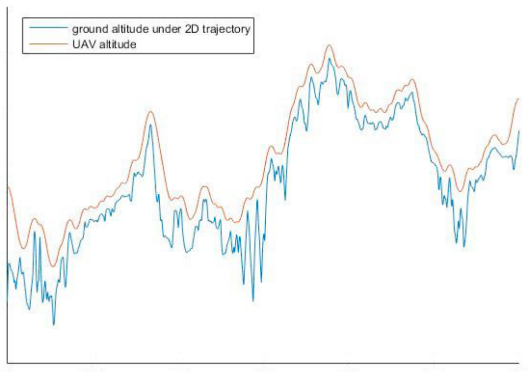

Fig. 4. Illustration of the UAV fly altitudes after computing its third dimension. The TSP is at 450 meters above the ground, TEP is at 125 meters above the ground and others points have a safety height.

4. GENETIC ALGORITHM

The GA is a bio-inspired optimization method presented by Holland (1975). GA is a nondeterministic method based on the theory of Darwin evolution which claims, by natural selection, that the most fittest to survive will pass to the next generation their genetic characteristics which make them fittest to survive.

To develop a GA, the first decision is to determine how to encode a trajectory to fit in chromosome representation. We decide to define a trajectory with waypoints. This defining allows us to fully determine a trajectory with only few waypoints. A waypoint is defined with its position, its heading and with an optional height if we want to force the trajectory to start and/or end at a specific height above the ground. To determine the trajectory associated to a waypoints representation, we simply consider waypoints two by two. For two consecutive waypoints, we build a straight line after (resp. before) the first (resp. second) waypoint. Those straight lines are used to stabilize the heading. Then we make a turn at each extremity of the straight lines and we link the turns with an optional straight line between them (see Fig. 5). With our chromosome representation of a trajectory, it is easy to define crossover and mutations strategies. For the crossover we choose to have a single point crossover which generates two children from two parents (see Fig. 6). In the reproduction phase, the selection of the parents is done using the tournament selection method. For mutations, we elaborate fifth types of mutation: add a waypoint, remove a waypoint, move a waypoint, decouple a waypoint and change the heading of a waypoint (see Fig. 7).

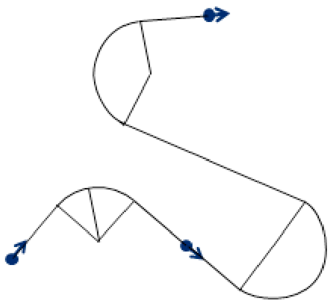

Fig. 5. Trajectory computing from waypoints representation

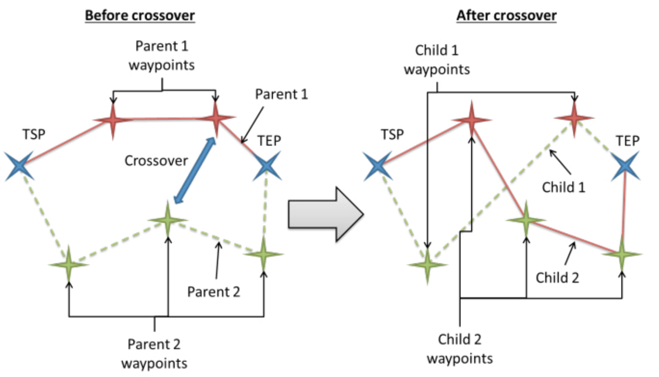

Fig. 6. Illustration of the single point crossover

The final step in our GA is to evaluate a solution. In order to determine the quality of a solution, we need to have the 3D trajectory associated to a chromosome. We obtain it by successively computing the 2D trajectory and then building the missing third dimension as it is mentioned before. Once we have the 3D trajectory, we evaluate it on all the objective criteria separately. It a give us a point in an R^n objective space where n is the number of criteria to optimize. Then solutions are ranked inside a Pareto Front (PF) where each rank contains several non-dominated solutions regarding to their rank. The Pareto Dominance is defined as follow: a solution $(a_1,\ldots,a_n)$ is dominated by a solution $(b_1,\ldots,b_n)$ if for all $i \in \{1,\ldots,n\}$, we have $a_i \geq b_i$ and $a_j > b_j$ for at least one $j \in \{1,\ldots,n\}$. A rank contains all non-dominated solutions excluding solutions of higher rank (see Fig. 8).

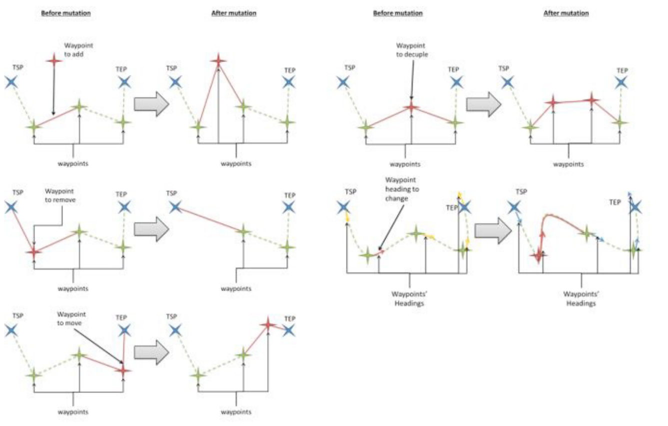

Fig. 7. Principal of each type of mutation

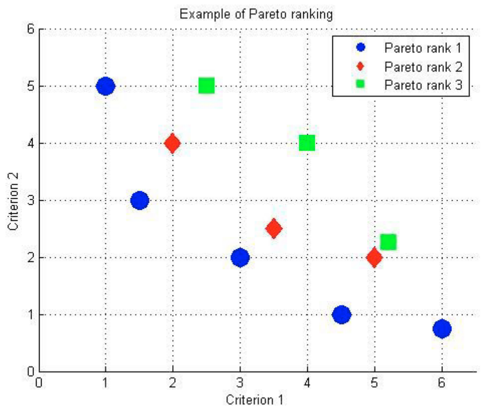

Fig. 8. Example of two objectives Pareto Front and Pareto rank.

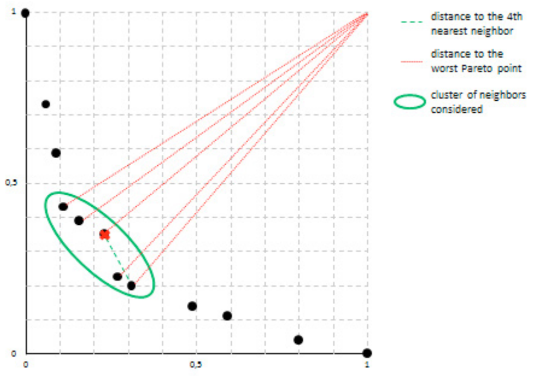

Fig. 9. PF reduction with a strategy of the nearest 4th neighbor.

When GA deals with a fitness function to evaluate solutions, it will provide the best solution in term of fitness at the end of the process. Therefore, with the PF structure, our GA provides no only one solution but all the solutions of rank 1 (i.e non-dominated solutions). This set of solutions might be too large for an external operator and might have a lot of at most identical solutions. Thus, we decide to provide only a subset (of fixed number) of Pareto rank 1 solutions. To make sure that the resulting subset fit the most the initial set, we remove point one by one as follow: for each point we compute the Euclidean distance to the kth nearest neighbours, then we consider the point with the minimal distance and its k nearest neighbours. In this cluster, the point closest to the worst point is removed (see the point with the red cross on Fig. 9). The method is repeated until the desired size is reached. An overview of our GA is illustrated in Fig. 10.

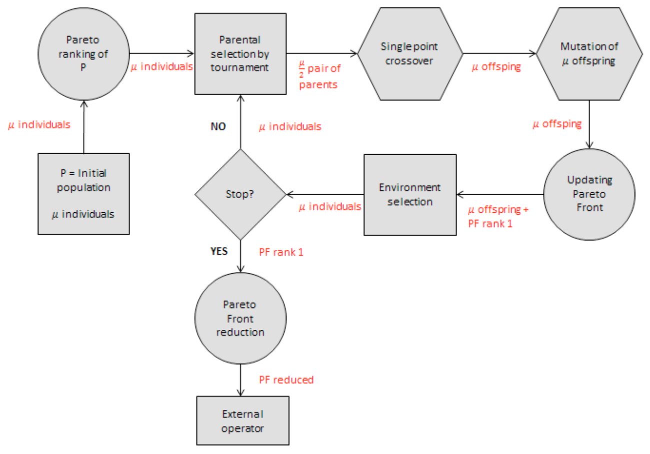

Fig. 10. Overview of the genetic algorithm.

5. A-STAR BASED ALGORITHM

The A-Star algorithm has been proposed by Hart et al. (1968). This algorithm is a deterministic algorithm which was first used to find the shortest path in a 2D grid. It does it iteratively by opening the node with minimal cost. The cost of a node is equal to the cost (the distance in the case of the shortest path) from the start node to the current plus the estimated cost from the current node to then end node. More this estimation is accuracy fastest the algorithm is. The algorithm guarantee to find an optimal solution if the heuristic used to estimate the further cost is a lower bound to the true cost.

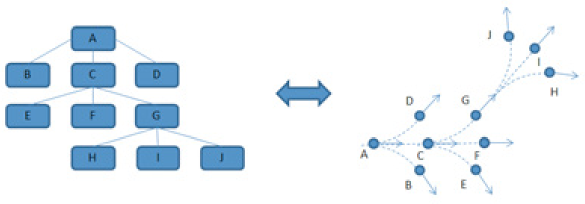

Fig. 11. Expansion pattern and associated exploring tree.

Our A-Star based algorithm only has to build a 2D trajectory. But the 2D trajectory has to respect the dynamic properties of the UAV in term of horizontal curvatures. Therefore, we decide to embed this requirement inside the moving pattern of the method. So, from the current node defined as a position (L (°), G (°)) and a heading (Ψ (°)) only three moves are possible (see Fig. 11): straight line, left turn and right turn of maximum horizontal curvature. With this kind of expansion features, it will be very unlikely to have two identical nodes. Whereas, in its classical form, A-Star prune a branch of its exploring tree when two branches lead to the same node. Indeed, the cost of the optimal path from the current node to the final node is the same for both branches but one of them has a cost from the start node to the current node greater than the other. Therefore, this branch will have a total cost (from start to end) greater than the other and so it can be prune. In our algorithm, we define equality between two nodes $n_a(L_a, G_a, \Psi_a)$ and $n_b(L_b, G_b, \Psi_b)$ with equations (2) where $dist(n_a,n_b)$ is the Euclidian distance between the nodes, where $\Delta(\Psi_a, \Psi_b) \in [0,\Pi]$ is the angular gap between nodes heading and where $dist_{\lim}$ and $\Psi_{\lim}$ are constants. When two nodes are equals, we prune the branch of the exploring tree associated to the node with the higher cost. Note that to determining if a node has reached the goal we used the same equality definition.

$$n_a = n_b \Leftrightarrow \begin{cases} dist(n_a, n_b) \le dist_{\lim} \\ \Delta(\psi_a, \psi_b) \le \psi_{\lim} \end{cases} \tag{2}$$

To have a running A-Star based algorithm, we need to define a heuristic to estimate the remaining cost for each criterion. For the criterion of minimizing the trajectory length, we could have used the Euclidian distance as heuristic as it is usually done. But, as mentioned before, more accurate the estimator is, fastest the algorithm is. Except that the Euclidian distance is not very accurate for our problem because it does not take any kind of heading and dynamic properties of the UAV. Therefore, instead of the Euclidian distance we decide to use the length of the trajectory that we will have obtained if we will have used a waypoint representation of the current node and the final node (see Fig. 5). For any other criterion, we decide to run our A-Star based algorithm with a shortest path objective, from the current node to the final node, then the third dimension is computed (see Fig. 4) and finally the trajectory is evaluated according to the criterion considered (see Fig 12). Thus, when we deal with the multi-objectives problem, for each node, we have an estimated cost from the start node to the end node $(f_1, \ldots, f_n) \in R^n$ where n is the number of objectives to optimize. To transform this cost into R, we use equation (3) where $\max_j f_i$ is the maximum of the cost of the criterion f_i amount all nodes. Basically, equation (3) is a weighted sum over normalized data.

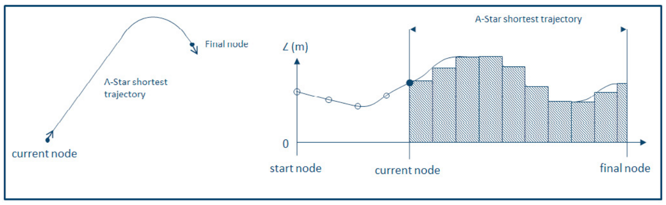

Fig. 12. Heuristic estimating the cost of altitudes overflown

$$f = \sum_{i=1}^{n} \frac{w_i \times f_i}{\max_j f_i} \tag{3}$$

Earlier we mentioned that to have an optimal path, the heuristic which estimates the cost has to be a lower bound to the true cost. It is obvious, it is not the case of our heuristics. Thus, a branch of the tree with a poor cost may lead to an optimal path. To try to still explore such a branch, we decide to open nodes with the help of a batch instead of doing it one by one. It means that, for each iteration, the *p* nodes with the lowest cost are placed into the batch (*p* is the size of the batch) and the children of the nodes are only considered at the next iteration when the batch is empty (a node is removed from the batch after being opened). Fig. 13 illustrates the batch opening process and Fig. 14 is an overview of our A-Star based algorithm.

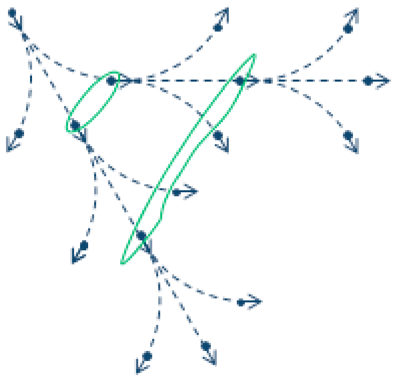

Fig. 13. Batch opening process with a batch of size two

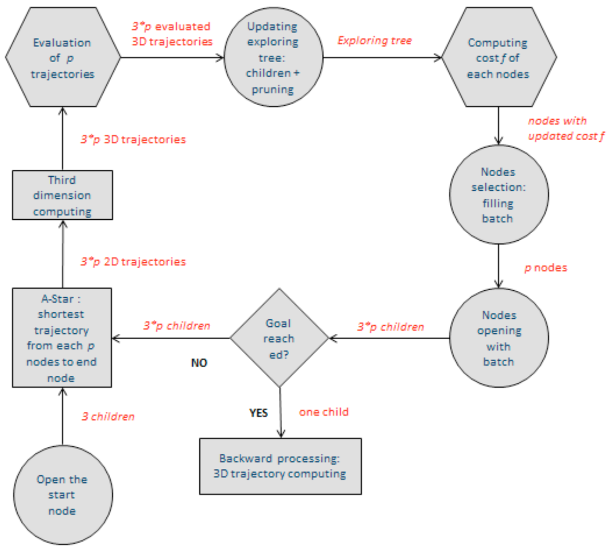
Evaluation of *p* trajectories
3*p evaluated 3D trajectories
Updating exploring tree: children + pruning
Exploring tree
Computing cost *f* of each nodes
nodes with updated cost *f*
Nodes selection: filling batch
p nodes
Nodes opening with batch
3*p children
Goal reached?
NO
3*p children
YES
one child
Backward processing: 3D trajectory computing
A-Star : shortest trajectory from each *p* nodes to end node
3*p 2D trajectories
Third dimension computing
3*p 3D trajectories
3 children
Open the start node

Fig. 14. Overview of the A-Star based algorithm

6. EXPERIMENTAL RESULTS

Finally, we decide to test our algorithms against 16 different scenarios. Each scenario was defined on a real terrain elevation map. The tactical situation of each scenario contains 1 to 4 radars and 2 to 8 GPS jammers. Optimization criteria considered are: minimizing the length, minimizing the mean altitude, minimizing radar exposure and minimizing the GPS jamming. The main parameters of each algorithm are presented in Table 1. Experimental results were computed on a Inter(R) Core(TM) i5-4570 CPU running at 3.20 GHz. To compare GA and A-Star, we compute the corresponding A-Star fitness (Equation (3)) of each GA solution. Table 2 illustrates the main experimental results. According to the fitness function, these results show that GA gives better solutions than solution obtained with our A-star based algorithm. In average, the best solution of GA is an improvement of 44.5% of the solution obtained by A-Star. But our A-Star based method has an average computation time of 836 ms whereas this average time is over 35 s for GA.

Table 1. Algorithms parameters

Algorithm	Parameters	Values
A-Star	Step size between two nodes	2000 m
A-Star	dist_{lim}	1000 m
A-Star	Ψ_{lim}	30 °
A-Star	Batch size	3
A-Star	Weight of length criterion	1
A-Star	Weight of mean altitude criterion	2
A-Star	Weight of GPS jammers criterion	2
A-Star	Weight of radars criterion	4
GA	Number of generations	128
GA	Number of chromosomes	128
GA	Tournament size	2
GA	Mutation rate	20%
GA	Neighbors size	1
GA	Final PF size	4

Table 2. Main results

	A-Star	GA
Mean fitness	1.714	0.9511
Mean CPU Time	836 ms	35632 ms
Best Solution	0/16	16/16

Fig. 15 and Fig. 16 highlight two scenarios. These scenarios are representative of the complexity we used for all the scenarios. For the scenario on Fig. 15, GA proposes two solutions without any radar exposure and any GPS jamming. One of them (yellow one) is 202 km long and used a valley to fly low (446 m in average). The other (green one) is shortest (154 km) but has to fly over higher ground (697 m in average). The remaining two trajectories are 132 and 142 km long with no GPS jamming but radar exposure (40 km) or not radar exposure but GPS jamming (33 km). When we look at the solution proposed by A-Star on this scenario, the trajectory obtained is 140 km long with no radar exposure and 34 km of GPS jamming. This trajectory is very close to the fourth trajectory proposed by GA. On this scenario, the best solution of GA (yellow one) is an improvement of 52% of the fitness value of the trajectory found by A-Star. For the scenario on Fig. 16, there is no solution without radar exposure because the goal is placed on the radar visibility domain. GA proposes one trajectory (yellow one, 181 km) with not GPS jamming and a low radar exposure (6 km). One important thing illustrated on Fig. 16 is that this trajectory is able to use the narrow path inside the radar visibility to avoid to be exposed too much (GPS jammers has been voluntary removed from the picture to see this narrow path). The second trajectory (green one) is a shortest solution (154 km) which exploits as well the failure inside the radar visibility but trades its shortest length against GPS jamming (37 km). The last two remaining solutions are a trade-off between a straight line and minimizing the radar exposure. On this scenario, the trajectory obtained by A-Star is also very close to the fourth trajectory of GA. This trajectory is 126 km long and has an exposure to radar of 37 km. As for the previous scenario, the fitness value of the best trajectory of GA is an improvement of 52% compare to the trajectory of A-Star.

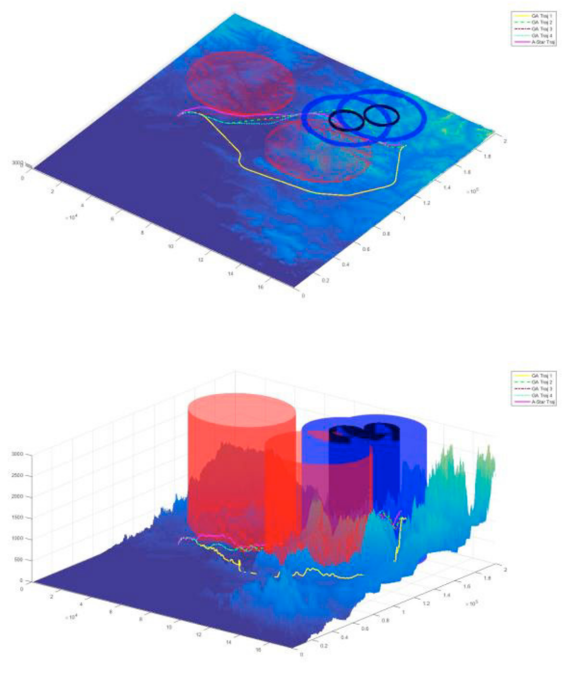

Fig. 15. 2D and 3D visualization of computed paths (Rhone Valley, France)

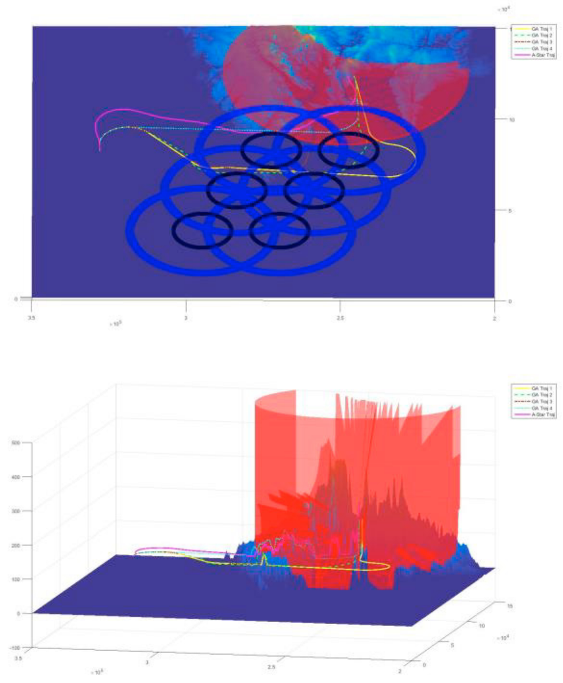

Fig. 16. 2D and 3D visualization of computed paths (Bretagne, France)

7. CONCLUSIONS

This paper presents two methods for the path planning of UAVs. These methods compute 3D trajectories in a real environment and consider the dynamic properties of the UAV. In this paper, we show that our A-Star based algorithm is suitable for real-time path planning. This paper also point out that our GA is performant and flexible. Indeed, our GA is able to find and exploit failures inside the tactical situation and proposes a diversity of solutions. These two methods can be combined to have a fully operational system. In such a system, an operator can select a trajectory before the UAV take off and then can modify it in real-time during the fly with our A-Star based method.

Experimental results show that the best solution obtained with GA is an improvement of the fitness of the A-Star solution by 44.5% in average. This result is mainly due to the poor quality of our estimator. Further works will focus on this area and on parallelizing our methods.

REFERENCES

Goerzen, C., Kong, Z., and Mettler, B. (2009). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems,* volume (57), number (1-4), page 65-100.

Hart, P.E. Nilsson, N.J. Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetic,* volume (4), issue (2), page 100-107.

Holland, J.H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology*. The University of Michigan Press.

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press.

LaValle, S.M., Kuffner Jr, J.J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research,* volume (20), issue (5), page 378-400.

Macharet, D.G., Neto, A.A. and Campos M.F.M. (2010). Feasible UAV path planning using genetic algorithms and Bezier curves. In Springer, *Advances in Artificial Intelligence – SBIA 2010,* page 223-232, Berlin, Germany.

Mittal, S., and Deb, K. (2007). Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms. *IEEE Congress on Evolutionary Computation,* volume (7), page 3195-3202.

Roberge, V., Tarbouchi, M., and Labonte, G. (2013). Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-time UAV Path Planning. *IEEE Transactions on Industrial Informatics,* volume (9), page 132-141.

Szczerba, R.J., Galkowski, P., Glicktein, I.S., and Ternullo, N. (2000). Robust algorithm for real-time route planning. *IEEE Transactions on Aerospace an Electronic Systems,* volume (36), issue (3), page 869-878.

Volkan, P.Y. (2012). A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV. *Aerospace Science and Technology,* volume (16), page 47-55.