A Hybrid Evolutionary Algorithm for Offline UAV Path Planning

Soheila Ghambari[1(✉)], Lhassane Idoumghar[1(✉)], Laetitia Jourdan[2(✉)], and Julien Lepagnot[1(✉)]

[1] University of Haute-Alsace, IRIMAS Institute, Mulhouse, France
{soheila.ghambari,lhassane.idoumghar,julien.lepagnot}@uha.fr
[2] University of Lille, CRIStAL, UMR 9189, CNRS, Centrale Lille, Lille, France
laetitia.jourdan@univ-lille.fr

Abstract. This paper investigates the offline path planning problem of unmanned aerial vehicles (UAVs) for surveillance mission in complex urban environments. A new idea by coupling the differential evolution (DE) with A* algorithm is suggested to address the problem in large urban areas with narrow street and infrastructure of built environment. The proposed method consists of two phase: the first phase adopts DE to divide the straight line between source and destination into several smaller regions, while the second one utilizes A* for each region to find a collision-free and shortest path in parallel. In order to assess the efficiency of the suggested algorithm, a real-world scenario is examined. Evaluations exhibited promising results with proper accuracy and minimum computational time.

Keywords: UAV · Offline path planning · Differential evolution · A* algorithm

1 Introduction

The development of autonomous UAVs is of high interest to many military and civilian applications for various missions. In recent years, more studies focus on one of the essential aspect of UAV autonomy which is the capability for automatic path planning [7]. This process consists of finding an optimal or near-optimal collision-free path between the start and target positions; under specific constraints conditions. As a matter of fact, a suitable path planning strategy should be design not only to improve the effectiveness of the system (e.g., memory consumption and computational time) but also to communicate with other elements in order to comply with the mission requirements. Hence, implementing an effective algorithm entails a deep analysis of various contributing techniques [18].

Previous studies have presented a series of techniques to tackle the aforementioned problem based on different necessities such as performances optimization, collision avoidance, real-time planning, and safety maximization. They took hints from different research fields; like mathematics for graph-based and probabilistic approaches [2,15], physics for potential field algorithm [6], or computer science for artificial intelligence methods [5,17,22]. Generally, we can categorize the existing works into classical techniques (i.e., graph-based search methods, sampling-based approaches, potential field), computational intelligence (CI) methods, and hybrid approaches.

Graph-based searches (e.g., A* and Dijkstra) were developed to find the shortest path between two nodes of connected graphs with a greedy logic. One of the positive characteristic of these methods is their simplicity, which implies reduced computational time. They have deterministic nature and guarantee to find the optimal collision-free path, if it exists. However, the performance of these algorithms depends on the environment's total area due to the fact that they save all explored nodes in memory. Sampling-based methods, such as Probabilistic Roadmaps (PRM) [13] and Rapidly-exploring Random (RRT) [14] have proven to be an effective framework suitable for high-dimensional spaces to produce feasible solutions; nevertheless, they do not guarantee the optimality of the solution [9]. In recent years, CI methods including fuzzy system, neural network, and evolutionary algorithms (EAs) have received most of the research effort for solving UAV path planning problem [10,21]. They attract the attention of researchers because of: (a) their flexibility to solve large-scale complex problems, (b) their ability to apply different learning strategies to perform an effective search towards the global optimum, and (c) employing for both single and multiple UAVs. However, in practice, when the available computation resources and/or time are limited, they are not always the best choice.

These issues motivated us to present a novel hybrid approach inspired from incremental heuristic search which not only scales well with problem size but also speeds up the search process for a high quality path in a reasonable execution time. To do so, A* as an informed heuristic search strategy and DE algorithm as one of the most popular EAs are integrated in order to find the shortest collision-free path in high dimension spaces with minimum computational time. In this method, the search space is limited around the straight line between the start and target locations. This is motivated by the fact that taking into account the whole configuration space can raise the computational cost. Here, the start and the target points are connected to each other via a straight line (as the shortest path) regardless of the obstacles. Then, we apply DE algorithm to divide the straight line into several suitable segments/regions. Thereafter, A* is used as a local path planner to find the shortest path for each region in a parallel manner. Altogether, the suggested method reduce the dimensionality of the search space which enables the presented algorithm to find better topologically distinct paths more rapidly. The performance of the proposed method is compared with A* and standard DE algorithm for a realistic urban environment. Evaluations exhibited desirable run-time performance in finding feasible and safe paths.

The rest of this paper is organized as follows. Section 2 starts with problem definition in Subsect. 2.1. Next, in Subsects. 2.2 and 2.3 basic concepts of A* and DE algorithm are explained, respectively. The description of the proposed method including environment modeling, constraints, solution representation, objective function, and the suggested algorithm are provided in Sect. 3. The simulation results and discussion are presented in Sect. 4. Finally, the paper is concluded in Sect. 5.

2 Background Information

2.1 Problem Definition

Generally, path planning belongs to a class of non-deterministic polynomial-time (NP) hard problems [4] which is much more intensively investigated in robotics (referred as motion planning). Formally, path planning for UAVs defined as an optimization problem aimed at finding the shortest and safest path to reach a goal position, while flying into a high-threat area. Here, some important factors should be taken into consideration such as modeling the environment, the path representation, safety, cost of the path, and computational time. These factors are either integrated directly into the objective functions that require to be minimized/maximized, or in the form of constraints that a path must comply with. The later subsections elaborates these factors with more details.

2.2 A* Search Algorithm

The history of finding the shortest path can be followed as early as 1968, when A* as the most effective direct search method is developed for robot navigation [8,12]. The algorithm acts on the basis of *Dijkstra*, but can avoid blind search to improve search efficiency. It seeks towards the most promising states using a heuristic function in order to save the computational time resource. A detailed explanation of A* can be found in [12].

2.3 Differential Evolution

DE algorithm has been successfully employed in various research and application areas. It has been also utilized in path planning tasks for both single UAV and multiple UAVs [3,19,20]. DE is an iterative procedure which aims at evolving a population (NP) of D-dimensional parameter vectors towards the global optimum. It includes a population of path candidate solutions or individuals which are produced by integrating a parent and other individuals of the same population. Each candidate solution has a set of variables which subjected to mutation and crossover search operators in order to produce new solutions subject to some constraints. The algorithm only accepts the candidate solutions that are better than their parents and accordingly transfers them to the next generation of the algorithm. The algorithmic description is summarized in Fig. 1. In this figure, the five most frequently utilized mutation strategies are listed.

The indices r_1, r_2, r_3, r_4, r_5 are mutually exclusive integers randomly generated within the range $[1, NP]$, which also differ from the index i. These indices

The DE Algorithm Pseudo Code

```
Generate the initial population
Evaluate the fitness for each individual
While the stopping criterion is not satisfied do
For $i$ =1 to NP do
    Select three mutually different individuals $r_1 \neq r_2 \neq r_3$
        $\neq i$
    $j_{rand} = int\,(rand\,[1, D])$
  For $j$ =1 to D do
    If $rand\,[0,1] < CR$ or $j = j_{rand}$ then
      $u^j_{i,G}$ Apply the predetermined strategy
    Else
      $u^j_{i,G} = x^j_{i,G}$
    End if
  End for

 End for
For $i$ =1 to NP do

   Evaluate the offspring
   If the fitness function value of $u_{i,G}$ is no worse than
       $x_{i,G}$ then replace $x_{i,G}$ with $u_{i,G}$
   End if
  End for
End while
```

Different type of strategy applied in DE algorithm
DE/rand/1 $u_{i,G} = x_{r1,G} + F\,(x_{r2,G} - x_{r3,G})$
DE/best/1 $u_{i,G} = x_{best,G} + F\,(x_{r1,G} - x_{r2,G})$
DE/rand-to-best/1 $u_{i,G} = x_{i,G} + F\left(x_{best,G} - x_{i,G}\right) + +F\,(x_{r1,G} - x_{r2,G})$
DE/best/2 $u_{i,G} = x_{best,G} + F\left(x_{r1,G} - x_{r2,G}\right) + +F\,(x_{r3,G} - x_{r4,G})$
DE/rand/2 $u_{i,G} = x_{r1,G} + F\left(x_{r2,G} - x_{r3,G}\right) + +F\,(x_{r4,G} - x_{r5,G})$

Fig. 1. The pseudo code of DE algorithm

are randomly generated once for each mutant vector. The scaling factor F is a positive control parameter for scaling the difference vector. The crossover rate CR is a user-specified fixed within the range $[0, 1)$, which controls the fraction of parameter values copied from the mutant vector. $X_{best,G}$ is the best individual vector with the best fitness value in the population at generation G. j_{rand} is a randomly chosen integer in the range $[1, D]$.

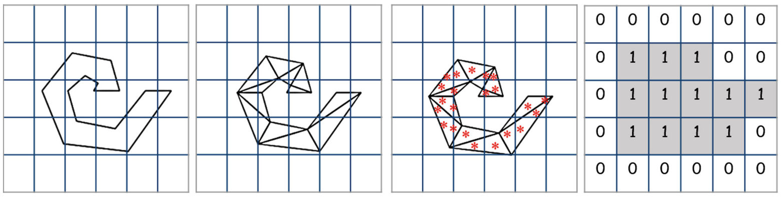
3 The Proposed Approach

First, a clear description of environment modeling, constraints, solution representation, and objective function is presented. Thereafter, the introduced algorithm is described in details.

3.1 Environment Modeling and Constraint

In this work, we considered a grid-based map to represent the environment. The grid map is composed of equal size cells, where each cell is represented by a unique number. An urban environment in a 2-dimensional (2D) form is pre-processed to generate the grid map. In this step, an occupancy matrix is utilized for grid map representation where each cell has two possible values: "0" for a free and "1" for an occupied cell. The buildings with different polygon shapes are considered as obstacles; which are static and known in advance. In order to understand how these polygon shapes occupy the grid cells, polygon triangulation method is used to decompose a polygon area into a set of triangles with pairwise non intersecting interiors. Accordingly, we check whether a grid cell lies inside a triangle or not (see Fig. 2). The occupancy matrix is pre-processed only once and the back-tracking process for making paths does not consume significant computational resources.

The constraint is path safety which means that a path always should satisfy a predefined safety margin (distance) with respect to the obstacles. In this work, the safety margin is the confidence radius of UAV around obstacles which is considered as 1 unit.



0	0	0	0	0	0
0	1	1	1	0	0
0	1	1	1	1	1
0	1	1	1	1	0
0	0	0	0	0	0

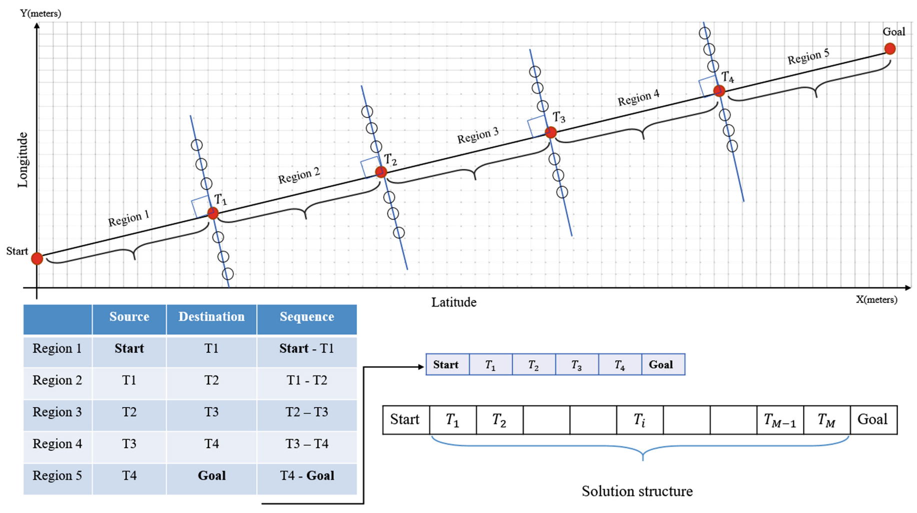
Fig. 2. The obstacle modeling in a grid map representation (The first three figures, on the *left* side, show how an arbitrary polygon shape occupies grid cells using the triangulation method. The occupancy map is represented in the *right* side figure.)

3.2 Solution Representation

The solution representation is an essential element for solving an optimization problem. In this study, each solution/path consists of a sequence of design variables. These variables are adopted based on grid cells that are located in the straight line between the start and target positions; with their unique numbers. In this way, the algorithm focuses on the most promising parts of the search space which can enhance the convergence performance. If a variable did not satisfy the constraint, the perpendicular line that passes through the selected variable is considered and another arbitrary point upon this line which is collision-free and near to the straight line will be chosen. An example of modeling the configuration space and solution representation is displayed in Fig. 3.

3.3 Objective Function

The objective function has to satisfy the constraints while optimizing the flight path and avoiding obstacles. Here, owing to employing grid map representation a feasible flight path can be defined from the start to the destination cell by traversing a certain number of free cells [1]. Hence, the cost of a feasible path is the sum of all costs of the movements along the associated path in all regions. The UAV is assumed to move horizontally or vertically or diagonally from a free cell to another one with fixed flight altitude. Accordingly, there are eight



	Source	Destination	Sequence
Region 1	**Start**	T1	**Start** - T1
Region 2	T1	T2	T1 - T2
Region 3	T2	T3	T2 – T3
Region 4	T3	T4	T3 – T4
Region 5	T4	**Goal**	T4 - **Goal**

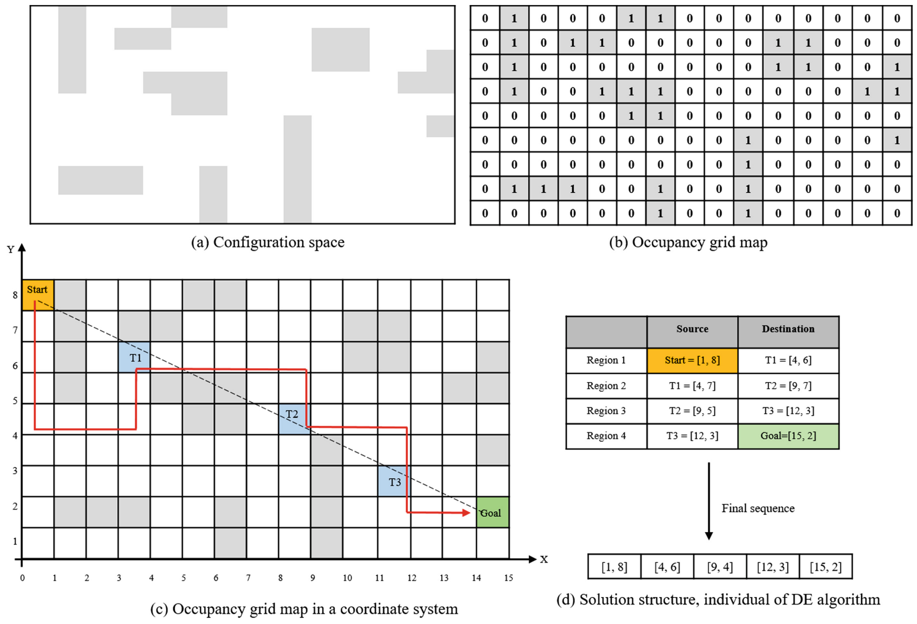
Fig. 3. The solution representation

possible moves from each cell to another one. It is worth mentioning that the main contribution of this work is to apply A* heuristic objective function to find the shortest path length in a desirable run-time.

3.4 ADE Algorithm

As mentioned before, adopting a fast and efficient path planning method is critical for autonomous UAVs which usually operate in large scale urban environments. There are various intelligent optimization methods which have been successfully applied in solving UAV path planning problem [11,21]. In the same direction and without loss of generality, this paper presents a new approach by integrating A* and DE algorithm in order to generate the shortest path with minimum computational time over very long distances.

The introduced hybrid algorithm, named as ADE, contains two main steps. The first step is accomplished with DE which is responsible for intersecting the whole area into conjunct regions. In fact, it determines several intermediate cells for exploring better the configuration space. These cells are DE's design variables which are located on the straight line between the start and target positions. In this way, the algorithm focuses on the most promising parts of the search space which can enhance its convergence performance. As explained in previous subsection, if a variable did not satisfy the constraint, a straight line perpendicular to the selected variable is considered, and another arbitrary point upon this line will be chosen by DE. This point should have two conditions: (a) be in an admissible space, (b) have a minimum distance from the straight line. In such a way, a proper balance between the exploration and exploitation capabilities of DE algorithm is achieved. In the second step, A* algorithm is employed to find the shortest path in each region in parallel. Thus, all the paths obtained from regions are connected to form the global best path. Hence, the algorithm is widely favorable for reducing computational time. An example of modeling the configuration space and the proposed algorithm is displayed in Fig. 4. Moreover, the flowchart of the proposed algorithm is shown in Fig. 5.



(a) Configuration space

0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	1	1	0	0	0
0	1	0	0	0	0	0	0	0	0	1	1	0	0	1
0	1	0	0	1	1	1	0	0	0	0	0	0	1	1
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	1	0	0	1	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	0	0	0

(b) Occupancy grid map

(c) Occupancy grid map in a coordinate system

	Source	Destination
Region 1	Start = [1, 8]	T1 = [4, 6]
Region 2	T1 = [4, 7]	T2 = [9, 7]
Region 3	T2 = [9, 5]	T3 = [12, 3]
Region 4	T3 = [12, 3]	Goal=[15, 2]

Final sequence

[1, 8]	[4, 6]	[9, 4]	[12, 3]	[15, 2]

(d) Solution structure, individual of DE algorithm

Fig. 4. An example of the proposed path planning algorithm: (a) configuration space, (b) occupancy grid map, (c) an obtained path in a coordinate system, (d) solution representation

4 Experimental Evaluation

This section aims to investigate the efficiency of the presented algorithm through a series of experiments on a realistic urban environment. The selected test case provided the chance to conduct a comprehensive study on the performance of algorithm in terms of path length and computational time. For this purpose, Subsect. 4.1 begins with a description of the test case characteristics. Then, in Subsect. 4.2, the setting parameters are introduced. In order to automatically configure the algorithm's parameters, *irace* package is utilized. Finally, the compared algorithms and statistical results obtained via experiments on urban map are presented and discussed in Subsects. 4.3 and 4.4, respectively. All simulations and evaluations were implemented and conducted within Python library[1], on a computer with Intel Core i5-7440HQ CPU, 2.80 GHz, 8GB RAM, running on Ubuntu OS.

[1] (Atsushi Sakai et al. https://github.com/AtsushiSakai/PythonRobotics).

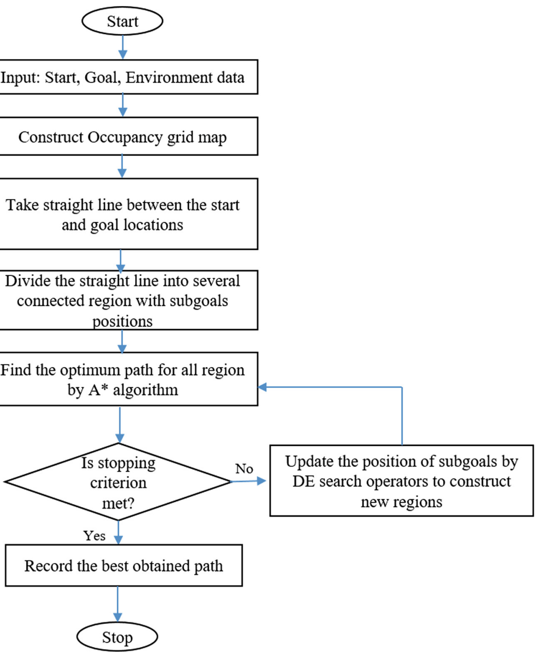

Start
Input: Start, Goal, Environment data
Construct Occupancy grid map
Take straight line between the start and goal locations
Divide the straight line into several connected region with subgoals positions
Find the optimum path for all region by A* algorithm
Is stopping criterion met?
No
Update the position of subgoals by DE search operators to construct new regions
Yes
Record the best obtained path
Stop

Fig. 5. The flowchart of the proposed algorithm

4.1 Test Case

The experiments have been extended with realistic urban environment. The selected environment for evaluating the performance of the algorithm is a partial part of Mulhouse city in France. The map file is extracted from OpenStreetMap, defined by geographical coordinates in terms of latitude and longitude. In this file the buildings tags are filtered. These building are taken into account as obstacles and their modeling is explained in Subsect. 3.1. The characteristics of this map are summarized in Table 1. In addition, Fig. 6 shows the total map and part of its modeling. As can be seen from part of modeling, this map has narrow streets with compressed obstacles.

Table 1. The test case characteristics

Map	Latitudes	Longitudes	No. obstacles
Mulhouse	Minlat = 47.7250	Minlon = 7.3001	4099
	Maxlat = 47.7538	Maxlon = 7.3466	

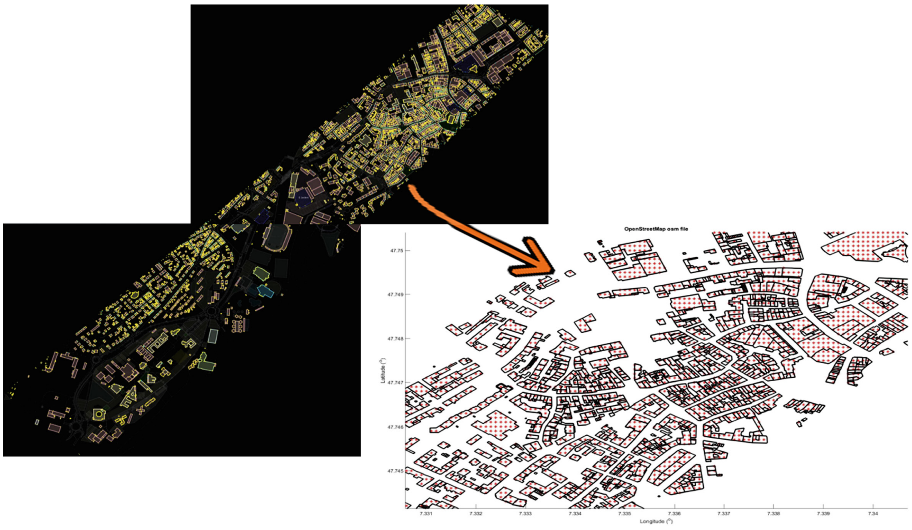


Fig. 6. Illustration of the selected map and employed polygon triangulation method

4.2 Experimental Setup

The configuration parameters of the algorithm can be divided into two categories: environment and algorithm parameters. Environment parameters include: grid size, start point, target point, number of obstacles and their coordination, and the boundary of the search space which are the minimum and maximum of latitudes and longitudes. The algorithm parameters are population size, crossover probability (CR), scaling factor (F), type of DE strategies, number of design variables, and number of iterations. The number of variables, which is assumed as dimension of the problem or regions, is an integer within the range $[1, 9]$. If the algorithm adopts 1, it means A* algorithm is applied for the total configuration space. Also, the maximum number of iterations and runs for this work are 100 and 20, respectively.

Table 2 describes the configurable settings of the proposed algorithm. As mentioned above, some parameter settings including population size, CR, F, and type of DE strategy are significant for a certain value and have a great impact on the performance of algorithm. Hence, instead of using a trial-and-error approach to identify good values for these parameters, we utilized *irace* software [16] as an automated algorithm configuration tool for obtaining very high-performing algorithmic variants. A maximum budget of 200 experiments is applied for each run of *irace* and it is repeated 20 times to assess the variability of the automatic configuration process. According to the obtained results, the best configuration uses $[80, 1, 0.2, 1]$ values for population size, F, CR, and the type of strategy parameters, respectively. The related DE mutation strategy, labelled by the number 1 during the parameter setting, is DE/rand/1.

Table 2. The setting parameter of the algorithm

Parameter	Value
Grid space	461 * 286
Start	[x = 84, y = 40]
Goal	[x = 387, y = 251]
Grid size	1
Population size	[1, 100]
CR	[0.1, 1]
F	[0.1, 2]
No. strategy	[1, 5]
Max iteration	100
Max run	20

4.3 The Effect of Different Number of Regions

As was mentioned before, the presented approach divides the distance between the start and target locations into several regions. The number of these regions which are taken into account as the number of dimension are very important to be determined. Thus, to investigate whether this number has a positive effect on the performance of the algorithm in terms of precision of path length and computational time, a comparison using different number of regions is performed. Parameter configurations for this experiment are similar to the settings explained in the previous subsection. Figure 7 exhibits the average and standard deviation of path length and computational time for different number of regions over 20 independent runs. As can be seen, by increasing the number of dimension the computational time significantly decreased; while as expected the precision of path length is approximately reduced. Furthermore, standard deviation of the results shows the stability of the presented method. As a matter of fact, it clearly confirms that the difference between regions can considerably affect the computational time which is very important factor especially in large scale environment. Another interesting observation that can be concluded from these results is that this approach makes the problem as a low dimensional problem using less number of decision variables for dividing the regions.

4.4 Results and Discussion

The presented algorithm was executed over 20 independent runs. The results are presented by the best, mean, and standard deviation (S.D.) of cost values obtained in all runs. To provide a meaningful comparison of A*, DE, and ADE, the mean and S.D. of the path length and computational time are compared with each other. All experimental results are reported in Table 3.

As it was expected, the results of ADE shows the impact of adopting different number of regions in accuracy of path length and computational time. ADE

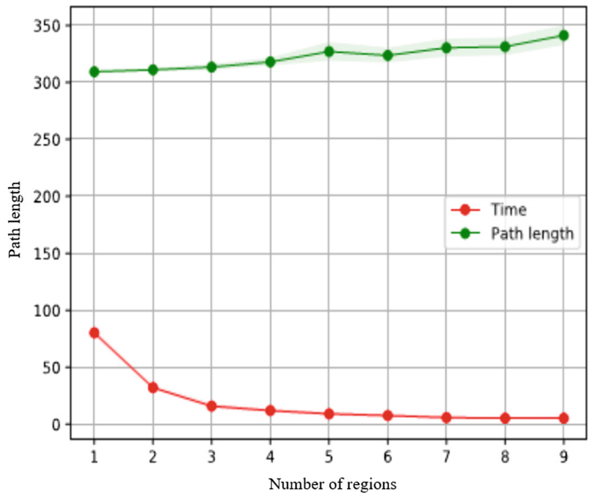


Fig. 7. The effect of different values for number of regions

with two regions has a smaller path length than the other dimension. However, its computational time is greater. The other dimensions have a close competition in accuracy of path length where by increasing the number of regions, the computational time surprisingly reduced. Also, the result of original DE shows that this algorithm was not able to find the shortest path in a reasonable time. One of the reason for such bad performance is the small number of iterations that makes it hard for DE to find the best set of grid cells. Finally, the results of

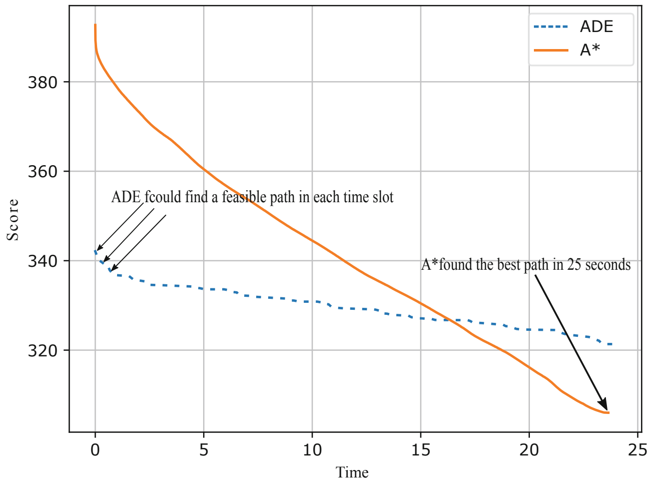


Fig. 8. The obtained feasible path in a predetermined time slot for both A* and ADE algorithm

Table 3. Results obtained for 20 independent runs of algorithms for offline path planning

Algorithm	Path length	Path length	Computational time (s)
	Best	Mean ± S.D.	Mean ± S.D.
ADE (D = 1)	308	309.12 ± 0.76e+00	38.60e+00 ± 2.43e+00
ADE (D = 2)	309	310.05 ± 0.76e+00	32.10e+00 ± 1.25e+00
ADE (D = 3)	310	313.40 ± 1.95e+00	16.00e+00 ± 1.40e+00
ADE (D = 4)	312	316.00 ± 2.67e+00	12.00e+00 ± 0.72e+00
ADE (D = 5)	315	326.05 ± 7.56e+00	9.14e+00 ± 0.80e+00
ADE (D = 6)	317	322.40 ± 5.80e+00	7.70e+00 ± 0.65e+00
ADE (D = 7)	324	327.40 ± 6.51e+00	3.98e+00 ± 0.73e+00
ADE (D = 8)	331	329.40 ± 4.42e+00	3.46e+00 ± 0.52e+00
ADE (D = 9)	339	342.40 ± 2.60e+00	3.03e+00 ± 0.08e+00
DE	2170	1180.00 ± 4.09e+03	66.65e+00 ± 3.81e+00
A*	307	307.00 ± 0.00e+00	13.80e+00 ± 0.16e+00

A* algorithm is reported in the last row of this table. A* could find the shortest path with high accuracy but with more computational time when compared to the presented ADE algorithm. There is a close competition between A* and ADE with dimension 4.

Also, Fig. 8 shows that the proposed method can give more accurate solutions in the early iterations, while A* finds the best possible flyable path using more computational resource. Indeed, this is the main properties of ADE which allows us to make a trade-off between these two conflicting objectives; as previously explained in Sect. 4.3.

5 Conclusion

This study concerns the development of a new path planning algorithm for UAVs so as to avoid obstacles in realistic urban environment for surveillance mission. The problem is modeled in a static 2D space with constraint single objective function. The suggested ADE approach integrated a heuristic search function with DE for large-scale environments. For this purpose, DE is employed to divide the configuration space into several conjunct regions. Then, each region is explored by A* algorithm as a local path planner in a parallel manner. The presented algorithm tries to search around the straight line between the start and destination which results in increasing the convergence performance and decreasing the computational time. The performance of the algorithm is evaluated through a series of experiments. The *irace* software package is also utilized in order to find the best configurations for the algorithm. The obtained results illustrated the efficiency of ADE in finding optimal solutions with proper accuracy and minimum computational time.

Acknowledgment. This work is part of a project funded by the French *Agence Nationale de la Recherche* under grant number ANR-16-SEBM-0004.

References

1. Alajlan, M., Koubâa, A., Châari, I., Bennaceur, H., Ammar, A.: Global path planning for mobile robots in large-scale grid environments using genetic algorithms. In: 2013 International Conference on Individual and Collective Behaviors in Robotics (ICBR), pp. 1–8. IEEE (2013)
2. Bauso, D., Giarré, L., Pesenti, R.: Multiple UAV cooperative path planning via neuro-dynamic programming. In: 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601), vol. 1, pp. 1087–1092. IEEE (2004)
3. Brintaki, A.N., Nikolos, I.K.: Coordinated UAV path planning using differential evolution. Oper. Res. **5**(3), 487–502 (2005)
4. Canny, J.: The Complexity of Robot Motion Planning. MIT Press, Cambridge (1988)
5. Cekmez, U., Ozsiginan, M., Sahingoz, O.K.: A UAV path planning with parallel ACO algorithm on CUDA platform. In: 2014 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 347–354. IEEE (2014)
6. Chen, Y., Luo, G., Mei, Y., Yu, J., Su, X.: UAV path planning using artificial potential field method updated by optimal control theory. In. J. Syst. Sci. **47**(6), 1407–1420 (2016)
7. Choi, Y., Choi, Y., Briceno, S., Mavris, D.N.: Energy-constrained multi-UAV coverage path planning for an aerial imagery mission using column generation. J. Intell. Robot. Syst. **97**(1), 125–139 (2019). https://doi.org/10.1007/s10846-019-01010-4
8. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959)
9. Gammell, J.D., Srinivasa, S.S., Barfoot, T.D.: Informed RRT*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2997–3004. IEEE (2014)
10. Ghambari, S., Lepagnot, J., Jourdan, L., Idoumghar, L.: A comparative study of meta-heuristic algorithms for solving UAV path planning. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 174–181, November 2018. https://doi.org/10.1109/SSCI.2018.8628807
11. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous UAV guidance. J. Intell. Robot. Syst. **57**(1–4), 65 (2010)
12. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
13. Kavraki, L., Svestka, P., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**, 566–580 (1994)
14. LaValle, S.M., Kuffner Jr., J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**(5), 378–400 (2001)
15. Li, J., Sun, X.: A route planning's method for unmanned aerial vehicles based on improved a-star algorithm. Acta Armamentarii **7**, 788–792 (2008)
16. López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Technical report, TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles (2011)

17. Ji, X.-T., Xie, H.-B., Zhou, L., Jia, S.-D.: Flight path planning based on an improved genetic algorithm. In: 2013 Third International Conference on Intelligent System Design and Engineering Applications, pp. 775–778. IEEE (2013)
18. Yang, P., Tang, K., Lozano, J.A., Cao, X.: Path planning for single unmanned aerial vehicle by separately evolving waypoints. IEEE Trans. Robot. **31**(5), 1130–1146 (2015)
19. Zhang, X., Duan, H.: An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. Appl. Soft Comput. **26**, 270–284 (2015)
20. Zhang, X., Chen, J., Xin, B., Fang, H.: Online path planning for UAV using an improved differential evolution algorithm. IFAC Proc. Volumes **44**(1), 6349–6354 (2011)
21. Zhao, Y., Zheng, Z., Liu, Y.: Survey on computational-intelligence-based UAV path planning. Knowl.-Based Syst. **158**, 54–64 (2018)
22. Zhu, Y., et al.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3357–3364. IEEE (2017)