UAV Path Planning Based on Variable Neighborhood Search Genetic Algorithm

Guo Zhang[1,2], Rui Wang[1,2(✉)], Hongtao Lei[1,2], Tao Zhang[1,2], Wenhua Li[1,2], and Yuanming Song[1,2]

[1] College of Systems Engineering, National University of Defense Technology, Changsha 410073, China
[2] Hunan Key Laboratory of Multi-Energy System Intelligent Interconnection Technology, Changsha 410073, China

Abstract. This study proposed a new genetic algorithm with variable neighbourhood search (GAVNS) for UAV path planning in three-dimensional space. First, an 0–1 integer programming mathematical model is established by inspired from the vehicle routing planning model with time window (VRPTW), and then a heuristic rule based on space vector projection is designed to quickly initialize high-quality solutions that meet constraints of upper error limit and minimum turning radius. Second, it improves mutation operator with a reselected mutation strategy, and incorporates Variable Neighborhood Search strategy based on adding and deleting route during the search process; Finally, GAVNS is compared with general Genetic Algorithm on a set of experiments. It is demonstrated that GAVNS algorithm is both effective and efficient. Moreover, the introduction of variable neighborhood search strategy enhances the local search ability of Genetic Algorithm.

Keywords: Path planning · GAVNS · 0–1 integer programming model

1 Introduction

Unmanned Aerial Vehicle (UAV) are widely applied in military fields such as battlefield reconnaissance and long-range strike due to their advantages such as easy concealment, strong maneuverability, as well as unmanned driving, and have become one of the important attack methods for unmanned and intelligent battlefields in the future [1]. In the face of complicated environmental conditions and diversified uncertain factors, advance trajectory planning is particularly important [2]. The problem of fast trajectory planning is an important subject in UAV control. The main algorithms to solve the problem include Artificial Potential Field method [3], Mixed Integer Linear Programming method [4], A* Algorithm [5], Genetic Algorithm [6], Particle Swarm Optimization [7] and Ant Colony Optimization [8], etc. The intelligent optimization algorithm is widely applied in trajectory planning problems such as Chen [9] et al. converted the UAV path optimization problem into a TSP problem and applied the Ant Colony Optimization to solve it; Literature [10] firstly used the Dubins Algorithm to generate the initial feasible path of the UAV, and then used the Genetic Algorithm to improve the path to the optimality;

Y. Tan and Y. Shi (Eds.): ICSI 2021, LNCS 12690, pp. 205–217, 2021.
https://doi.org/10.1007/978-3-030-78811-7_20

Peng[11] et al. proposed an effective hybrid algorithm combining genetic algorithm and variable neighborhood search together for the permutation flow shop scheduling problem. Due to the limitation of the system structure, the positioning system of the UAV cannot accurately position itself, and the accuracy is affected by the accumulated error of the built-in inertial navigation system, which may lead to mission failure once the positioning errors have accumulated to a certain level. Therefore, correcting the positioning error during the flight is an important task in the path planning of UAV.

Considering the cumulative error correction and the minimum turning radius limitation of the UAV, we establish the 0–1 integer programming model by appropriately simplifying the fast path planning problem of the UAV in three-dimensional space under the system positioning accuracy constraint and combine Variable neighborhood Search with Genetic Algorithm to solve it.

2 Model

2.1 Description and Analysis

It is assumed that the size and mass of the UAV are ignored during the flight, and it is considered as a mass point without considering the constraints such as fuel consumption, while its positioning error (i.e., horizontal and vertical error) does not have a delayed situation and can be updated synchronously in real-time with the accumulation of flight distance. Its path's constraints are as follows:

- The UAV needs real-time positioning during space flight, and its positioning error includes vertical error and horizontal error. For one meter of flight, the vertical and horizontal errors will each increase by δ. The vertical and horizontal errors should be less than θ when reaching the endpoint B. And to simplify the problem, it is assumed that the UAV can still fly according to the planned path when both vertical and horizontal errors are less than θ.
- The UAV needs to be corrected for positioning errors during the flight. There are safe locations in the flight area (called correction points) that can be used for error correction, and it will be corrected based on the type of error correction when the UAV reaches. If the vertical and horizontal errors can be corrected in time, the UAV can follow a predetermined path, passing through several correction points to correct the errors and finally reach the endpoint.
- At the start point A, the vertical and horizontal errors of the UAV are both 0.
- When the UAV at the vertical error correction point, its vertical error will be corrected 0 and the horizontal error will remain unchanged.
- When the UAV at the horizontal error correction point, its horizontal error will be corrected 0 and the vertical error will remain unchanged.
- When the UAV's vertical error is no more than α_1 and the horizontal error is no more than α_2, the vertical error correction can be performed.
- When the UAV's vertical error is no more than β_1 and the horizontal error is no more than β_2, the horizontal error correction can be performed.

- The UAV is limited by the structure and control system during the turn and cannot make the instant turn (the forward direction of the UAV cannot be changed suddenly), assuming the minimum turn radius of the UAV $R = 200\,\text{m}$.

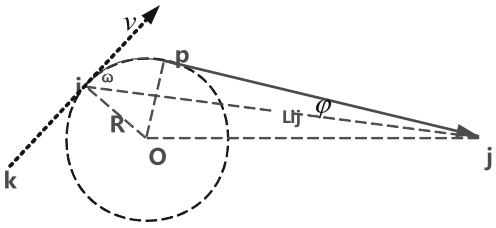

Fig. 1. Diagram of the ideal flight path of the UAV

The UAV come from i to j, the angle between i point's initial velocity $\overrightarrow{v}$ and $\overrightarrow{ij}$ is ω, at that time the trajectory is the arc ip plus line segment pj, where pj is the tangent line of circle O, p is the tangent point. It is clearly that this trajectory is the optimal flight track for the UAV (see Fig. 1).

From the theorem of the cosine of a triangle, it follows that:

$$|Oj|^2 = R^2 + L_{ij}^2 - 2RL_{ij}\sin\varpi \tag{1}$$

In addition,

$$|pj|^2 = |Oj|^2 - R^2 \tag{2}$$

$$|ip| = R(\varpi + \varphi) \tag{3}$$

Where L_{ij} denotes the Euclidean distance from point $i(x_i, y_i, z_i)$ to point $j(x_j, y_j, z_j)$.

$$L_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{4}$$

Since $L_{ij} \gg R$, then $\varphi \to 0$. Thus the distance of the track between any two points $dis(i, j)$ can be obtained as follows:

$$\begin{aligned} dis(i,j) &= |ip| + |pj| = R(\varpi + \varphi) + (|Oj|^2 - R^2)^{1/2} \\ &= R\varpi + (R^2 + L_{ij}^2 - 2RL_{ij}\sin\varpi - R^2)^{1/2} = R\varpi + (L_{ij}^2 - 2RL_{ij}\sin\varpi)^{1/2} \end{aligned} \tag{5}$$

Where $\cos\varpi = \frac{<\overrightarrow{ki}, \overrightarrow{ij}>}{|\overrightarrow{ki}||\overrightarrow{ij}|}$, then

$$dis(i,j) = R\arccos\frac{<\overrightarrow{ki}, \overrightarrow{ij}>}{|\overrightarrow{ki}||\overrightarrow{ij}|} + \left(L_{ij}^2 - 2RL_{ij}\sin\arccos\frac{<\overrightarrow{ki}, \overrightarrow{ij}>}{|\overrightarrow{ki}||\overrightarrow{ij}|}\right)^{1/2} \tag{6}$$

2.2 Mathematical Model

We refer the Vehicle Route Problem with Time Window (VRPTW) [12] and establish the following mathematical model by replacing the time window constraint with an error accumulation constraint and removing the constraint that the vehicle must pass through all customer points.

Firstly, a set of points $N = \{1, 2, ..., n\}$ is created to represent all the correction points except for the start and end of the path; The edge matrix $X = \{x_{ij} | i, j \in N\}$, where x_{ij} is the edge from i to the j, $x_{ij} = 1$ indicates the UAV will come from i to j; The distance matrix $D = \{d_{ij} | i, j \in N\}$ where d_{ij} represents the distance between the correction points i, j; φ_i, γ_i represents the horizontal error and vertical error respectively when the UAV flies to the point i, and σ_i denotes the correction type of the correction point i where $\sigma_i = 0$ indicates that i is the horizontal correction point (otherwise it is the vertical correction point). It is assumed that a feasible trajectory is represented by $M_i = \{m_0 = 0, m_1, m_2, \ldots, m_s, m_{s+1} = n + 1\}$, where $0, n + 1$ denote the point A and B respectively in the problem, as well as $m_1 \sim m_s$ is a series of the ordered correction points selected from N, then the mathematical model of the problem is as follows:

$$\min f_1 = \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} x_{m_i m_j} d_{m_i m_j} \tag{7}$$

$$x_{ij} = x_{ji} \ , \ \forall i, j \in N \tag{8}$$

$$x_{ii} = 0, \ \forall i \in N \tag{9}$$

$$x_{ij} \odot x_{jk} = 1, \forall j \in \mathrm{N}, \exists i, k \in \mathrm{N} \ and \ i \neq k \tag{10}$$

$$\sum_{i=1}^{n} x_{0i} = 1, \forall i \in N \tag{11}$$

$$\sum_{i=1}^{n} x_{i(n+1)} = 1, \forall i \in N \tag{12}$$

$$\varphi_{m_i} \leq \alpha_1, \sigma_{m_i} = 1 \, and \, x_{m_i m_{s+1}} = 0 \tag{13}$$

$$\gamma_{m_i} \leq \alpha_2, \, \sigma_{m_i} = 1 \, and \, x_{m_i m_{s+1}} = 0 \tag{14}$$

$$\varphi_{m_i} \leq \beta_1, \sigma_{m_i} = 0 \, and \, x_{m_i m_{s+1}} = 0 \tag{15}$$

$$\gamma_{m_i} \leq \beta_2, \sigma_{m_i} = 0 \, and \, x_{m_i m_{s+1}} = 0 \tag{16}$$

$$\varphi_{m_s} + x_{m_s m_{s+1}} d_{m_i m_{s+1}} \delta < \theta \tag{17}$$

$$\gamma_{m_s} + x_{m_s m_{s+1}} d_{m_s m_{s+1}} \delta < \theta \tag{18}$$

$$|Oj| \geq R, \forall j \in N \tag{19}$$

$$\varphi_{m_0} = \gamma_{m_0} = 0 \tag{20}$$

Equation (7) is the optimization objective of the UAV path planning problem, indicating that the total length of the trajectory path is minimized; Eq. (8) means that the path planning is simplified to an undirected point selection and connection problem, only considering whether two points are connected to each other without the direction of the connection; Eq. (9) means that any correction point cannot be connected with itself to form a loop; Eq. (10) means that for any correction point other than the and the end point, there are only two cases where it is not connected to any point or connects to a different point; Eq. (11), (12) means that the path must contain the start and the end point; Eq. (13), (14) means that if a correction point is a horizontal correction point and not the end in the path, its horizontal and vertical errors must be less than α_1 and α_2 respectively; Similarly Eq. (15), (16) means that if a correction point is a vertical and not the end in the path, its horizontal and vertical errors must be less than β_1 and β_2 respectively; Eq. (17), (18) means that the horizontal and vertical errors of the end point must be less than θ; Eq. (19) means that the turning radius of the UAV should be greater than or equal to the minimum turning radius R; Eq. (20) means that the initial horizontal and vertical errors of the start point are set 0.

3 Variable Neighborhood Search Genetic Algorithm

In the next, we have improved the Genetic Algorithm by design the specific mutation operator and make the best individual in each generation do variable neighborhood search. It enhances the local search ability of the Genetic Algorithm, avoiding the premature convergence of the Genetic Algorithm which leads to a local optimal solution, the steps as follows.

Algorithm 1: Genetic Algorithm with Variable Neighborhood Search
Input: the number of generations $mGen$, the number of population size N
Output: bets solution

```
gen ← 1
P ← initPopulation ();
While ( gen ≤ mGen) do
   parents ← Selection (P)
   child ← Crossover (parents)
   child ← Mutation (child)
   evaluateFitness (child)
   best_solution ← best_Selection (child)
   best_solution ← VNS (best_solution)
   parents ← parents | best_solution
   gen ← gen+1
End
```

3.1 Chromosome Coding

0 and 1 are used to encode the trajectory path, $x = \{x_1, x_2, ..., x_n\}$ representing an individual, where $x_i = 1$ means that the correction point i is selected for position error correction (see Fig. 2). x_1, x_{n+1} is constant to 1 means that the UAV must start from the start point A to reach the end point B. The order in which the UAV passes through the correction points is determined by the size of the projection of the correction points on the track onto the spatial vector. The length of the chromosome is determined by the number of all correction points to be selected in space.

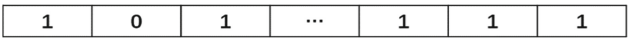

Fig. 2. Diagram of chromosome coding

3.2 Generating the Initial Feasible Solution

The quality of the initial feasible solution has a significant impact on the convergence of speed for algorithm. To address the dilemma that the problem's constraints are too complex to generate difficultly initial feasible solutions, we design a heuristic rule based on space vector projection, which can generate quickly initial feasible solutions with high quality.

From the start point A, the correction points are selected in turn: firstly, the type of the next correction point and the set of candidate correction points are determined, and finally, a specific correction point is selected from the candidate according to the heuristic rule of space vector projection.

$$\sigma_{i+1} = \begin{cases} 0, \ min(\alpha 1 - \gamma_i', \beta 1 - \gamma_i') > min(\alpha 2 - \varphi_i', \beta 2 - \varphi_i') \\ 1, \ otherwise \end{cases} \tag{21}$$

$$\varphi_j = \varphi_i' + dis(i, j)\delta \tag{22}$$

$$\gamma_j = \gamma_i' + dis(i, j)\delta \tag{23}$$

The first step is to determine the next correction point type. Firstly, the remaining horizontal and vertical error margins for the current point are calculated respectively, and the type of the next correction point is determined according to Eq. (21), where $\sigma_{i+1} = 1$ indicates that the type of the next is a vertical correction point, φ_i, γ_i is the horizontal and vertical error of the i point before correction, φ_i', γ_i' is the horizontal and vertical error after correction. The error update formula is shown in Eq. (22), (23).

The second step is to determine the set of candidate correction points C_i. According to Eq. (24), a sphere of radius r is made with the current point as the center of the sphere, then all correction points within the sphere are traversed, and the distance Oj from the candidate point to the center of the sphere O is calculated by according to Eq. (1). If $|Oj| < R$ means that the candidate does not satisfy the minimum turning

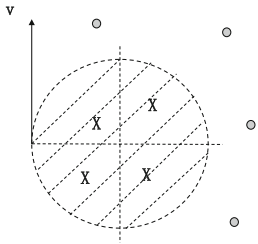

Fig. 3. Diagram of infeasible correction points

radius constraint, then it needs to be rejected. The dashed circular area is the candidate point that does not satisfy the minimum turning radius constraint (see Fig. 3), while it is a circular pipe in the three-dimensional space.

$$r_i = \frac{min(\zeta_1 - \varphi_i', \zeta_2 - \gamma_i')}{\delta} \tag{24}$$

$$K_{c_j} = \frac{\overrightarrow{i_{curr}c_j} \times \overrightarrow{AB}}{\left\| \overrightarrow{AB} \right\|} \tag{25}$$

$$P(c_j) = \frac{K_{c_j}}{\sum K_{c_j}} \tag{26}$$

The third step is to select a specific correction point. In order to identify a point c_j within the candidate point set C_i as the next correction point, we design a heuristic rule based on space vector projection to prioritize the points within the candidate point set. In conjunction with the greedy strategy which the closer to the end the better next point is. The spatial vector from the current point to all candidate c_j and the spatial vector from start point A to end point B are made respectively. By comparing the projections of the vectors on the vectors, if the projection is larger, it indicates that the candidate c_j is more capable of displacement relative to the end point, and the probability of being selected is greater (see Fig. 4). The specific formula is shown in Eq. (25), (26), where K_{cj} denotes the projection of candidate correction c_j on the vector $\overrightarrow{AB}$, and $P(c_j)$ denotes the probability of c_j being selected.

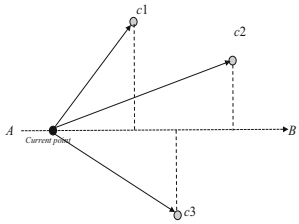

Fig. 4. Diagram of space vector projection

The fourth step, assigning the next point to the current point, it is to determine whether current point can directly reach the end point B without violating the end point error constraint Eq. (17), (18), if not, jump to the first step; if yes, current point is the last correction point of the path and then completes the construction of the initial feasible solution by adding the end point B to the path.

3.3 Cross and Mutation Strategy

After initialization of the population, the linear-rank selection is adopted to select individuals to generate a mating pool for crossover and mutation operations.

Crossover Strategy: a number *rand* in the interval [0, 1] is randomly generated, if *rand* is less than the crossover probability *Pc*, then individual i and $i + 1$ are selected as parents to execute single-point crossover operations (see Fig. 5).

Fig. 5. Diagram of chromosome crossover

Reselected Mutation Strategy: mutation increases the diversity of the population, but the multiple constraints make the mutated individuals mostly become infeasible solutions, so we design a specific mutation strategy to increase the probability of mutated individuals satisfying the constraints, with the following steps.

A number *rand* in the interval [0, 1] is randomly generated, and if *rand* is less than the mutation probability *Pm*, then individual i is selected for mutation. Randomly select position1 and reselect a correction point from the hemisphere of the candidate point corresponding to position1. If it there is no correction point of the same type as position1 within the hemisphere, the inverse mutation (common mutation) is taken for the position1. If the second is available, the position1 is reversed, meanwhile the position2 corresponding to the second will become what it was before the position1 mutation, as shown in Table 1.

Table 1. Schematic table of reselected mutation strategy

	State of position1 before mutation	State of position1 after mutation	State of position2 after mutation
Reselected mutation	1	0	1
Common mutation	1	0	/
	0	1	/

3.4 Individual Evaluation

Penalizing infeasible individuals is a common method applied in genetic algorithm's constraint processing. If the crossover and mutation produce offspring individuals that do not satisfy the constraint, the evaluation function $f = f_1 + M$, where M is a large positive penalty value that makes the genetic algorithm to find the optimal solution as closely as possible.

3.5 Variable Neighborhood Search Operator

The best individual of each generation is selected for the variable neighborhood search perturbation to improve the local search ability of the genetic algorithm while ensuring the quality of the solution, the process is shown in Algorithm 2.

Algorithm 2: Variable Neighborhood Search
Input: best solution Sol
Output: new solution Sol'

```
k ← 1
Generate Neighborhood N_1 ~ N_6
While ( k ≤ 6) do
    Sol' ← N_k(Sol)
    If ( f(Sol') < f(Sol)):
        k ← 1
    Else:
        Sol' ← Sol, k ← k+1
End
```

Neighborhood design:

- Neighborhood N_1: for the feasible solution obtained in the current iteration step, a new feasible solution is obtained by changing one correction point in its path. For example, a possible neighborhood solution for the path $0 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 8$ in N_1 is $0 \rightarrow 2 \rightarrow 7 \rightarrow 3 \rightarrow 8$.
- Neighborhood N_2: for the feasible solution obtained at the current iteration step, a new feasible solution is obtained by changing two consecutive correction points in its path. For example, a possible neighborhood solution for the path $0 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 8$ in N_2 is $0 \rightarrow 2 \rightarrow 7 \rightarrow 4 \rightarrow 8$.
- Neighborhood N_3: for the feasible solution obtained at the current iteration step, a new feasible solution is obtained by changing three consecutive correction points in its path. For example, a possible neighborhood solution for the path $0 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 8$ in N_3 is $0 \rightarrow 5 \rightarrow 7 \rightarrow 4 \rightarrow 8$.
- Neighborhood N_4: for the feasible solution obtained in the current iteration step, a new feasible solution is obtained by deleting one correction point in its path. For example, a possible neighborhood solution for the path $0 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 8$ in N_4 is $0 \rightarrow 2 \rightarrow 3 \rightarrow 8$.

- Neighborhood N_5: for the feasible solution obtained at the current iteration step, a new feasible solution is obtained by deleting two consecutive correction points in its path. For example, a possible neighborhood solution for the path $0 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 8$ in N_5 is $0 \rightarrow 2 \rightarrow 8$.
- Neighborhood N_6: for the feasible solution obtained in the current iteration step, a new feasible solution is obtained by deleting three consecutive correction points in its path. For example, a possible neighborhood solution for the path $0 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 8$ in N_6 is $0 \rightarrow 8$.

4 Experiment

The effectiveness of the hybrid algorithm proposed in this paper is tested on an i7–9700@3.0 GHz/16G/Win7 PC. According to the 0–1 integer programming model established in the previous, the hybrid model of Variable Neighborhood Search and Genetic Algorithm is implemented based on the jMetal framework [13], where the JDK version is 1.9 and the jMetal framework version is 5.10. The parameters of the hybrid algorithm are set: population size $N = 100$, mating pool size $poolSize = 100$, the maximum number of iterations $mGen = 500$, crossover probability $Pc = 0.8$, and mutation probability $Pm = 0.25$. The test dataset has 613 correction points and the parameters for each constraint are set as follows: $\alpha_1 = 25$, $\alpha_2 = 15$, $\beta_1 = 20$, $\beta_2 = 25$, $\theta = 30$, $\delta = 0.001$.

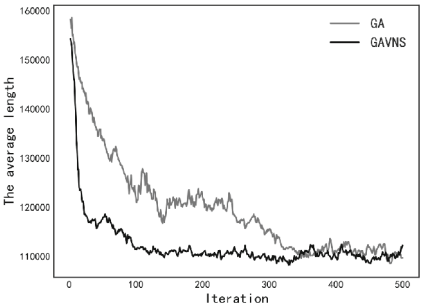

Fig. 6. Comparison of the population's average length by every iterated based on test data

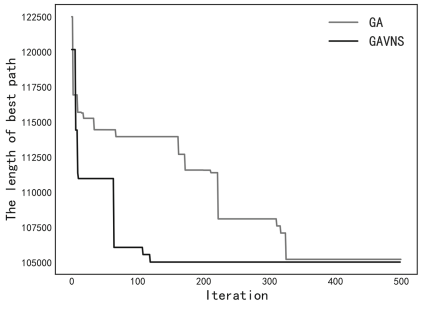

Fig. 7. Comparison of the optimal path's length by every iterated based on test data

The improved Genetic Algorithm with Variable Neighborhood Search (GAVNS) and the traditional genetic algorithm are compared in experiments on the test dataset, and the change curve of the average length for the population of each generation is shown respectively (see Fig. 6), while the length of the optimal path in the population of each generation is shown respectively (see Fig. 7). It can be seen that GAVNS starts to converge at around 100 generations, while GA is still evolving iteratively, which demonstrates that the introduction of variable neighborhood search strategy can strengthen the local search ability of the genetic algorithm.

GAVNS obtains the trajectory path after 500 iterations as $1 \rightarrow 504 \rightarrow 70 \rightarrow 507 \rightarrow 29 \rightarrow 184 \rightarrow 195 \rightarrow 451 \rightarrow 595 \rightarrow 398 \rightarrow 613$, the length is 105059.9m, passing through 9 correction points, the horizontal and vertical error of each correction point is shown in Table 2. GA also obtains the trajectory path after 500 iterations as $1 \rightarrow 504 \rightarrow 201 \rightarrow 81 \rightarrow 238 \rightarrow 283 \rightarrow 599 \rightarrow 562 \rightarrow 449 \rightarrow 486 \rightarrow 613$, the length is 105254.1m, passing through 9 correction points, the horizontal and vertical error of each correction point is shown in Table 3.This experimental comparison shows that the horizontal and vertical errors at each point of the two trajectory paths obtained by GA and GAVNS satisfy the constraint and both can complete the scheduled task, but the length of path obtained by GAVNS is better and has fewer iterations (see Fig. 7).

Table 2. GAVNS's planning path result

The no. of point	Before correction		After correction		The type of point
	Vertical error	Horizontal error	Vertical error	Horizontal error	
1	0.00	0.00	0.00	0.00	A
504	13.39	13.39	13.39	0.00	1
70	22.20	8.81	0.00	8.81	0
507	7.87	16.68	7.87	0.00	1
29	13.86	6.00	0.00	6.00	0
184	11.33	17.32	11.33	0.00	1
195	24.94	13.61	0.00	13.61	0
451	5.98	19.59	5.98	0.00	1
595	24.03	18.05	0.00	18.05	0
398	3.06	21.11	3.06	0.00	1
613	20.03	16.97	20.03	16.97	B

Table 3. GA's planning path result

The no. of point	Before correction		After correction		The type of point
	Vertical error	Horizontal error	Vertical error	Horizontal error	
1	0.00	0.00	0.00	0.00	A
504	13.39	13.39	13.39	0.00	1
201	14.25	0.87	0.00	0.87	0
81	15.75	16.61	0.00	16.61	0
238	4.63	21.24	4.63	0.00	1
283	18.31	13.68	0.00	13.68	0
599	11.11	24.79	11.11	0.00	1
562	22.06	10.96	0.00	10.96	0
449	5.15	16.11	0.00	16.11	0
486	5.73	21.84	5.73	0.00	1
613	29.30	23.57	29.30	23.57	B

5 Conclusion

In this paper, an 0–1 integer programming model is established for the path planning problem under several constraints such as the upper limit of horizontal and vertical error at the correction point as well as the minimum turning radius of the UAV. A genetic algorithm with improved population initialization and variation operator is proposed, and a two-level variable neighborhood search strategy for adding and deleting route points is introduced to strengthen the local search ability of the genetic algorithm. Finally, experiments are conducted on the above trajectory path planning problem and compare with the traditional genetic algorithm to demonstrate the stability and effectiveness of the proposed GAVNS hybrid algorithm.

The handling of unsatisfied constraint solutions is relatively straightforward, whereas the constraints on real-world optimization problems such as the fast path planning of the UAV are usually numerous and complex, improvement of the constraint handling approaches will follow so that GAVNS can be applied to more complex problems.

Acknowledgement. This work is supported in part by the National Nature Science Foundation of China under Grant 61773390 and Grant 61973310, the key project of National University of Defense Technology (ZK18–02–09), the Hunan Youth elite program (2018RS3081) and the key project ZZKY–ZX–11–04.

References

1. Aggarwal, S., Kumar, N.: Path planning techniques for unmanned aerial vehicles: a review, solutions, and challenges. Comput. Commun. **149**, 270–299 (2020)

2. Li, F., Li, Z.: Research on rapid planning of intelligent aircraft trajectory under multiple constraints. J. Phys. Conf. Ser. **1592**, 012021 (2020)
3. Chen, Y.-b., Luo, G.-c., Mei, Y.-s., Yu, J.-q., Su, X.-l.: UAV path planning using artificial potential field method updated by optimal control theory. Int. J. Syst. Sci. **47**, 1407–1420 (2016)
4. Radmanesh, M., Kumar, M.: Flight formation of UAVs in presence of moving obstacles using fast-dynamic mixed integer linear programming. Aerosp. Sci. Technol. **50**, 149–160 (2016)
5. Dong, Z., Chen, Z., Zhou, R., Zhang, R.: A hybrid approach of virtual force and A∗ search algorithm for UAV path re-planning. In: 2011 6th IEEE Conference on Industrial Electronics and Applications, pp. 1140–1145. IEEE (2011)
6. Arantes, M.d.S., Arantes, J.d.S., Toledo, C.F.M., Williams, B.C.: A hybrid multi-population genetic algorithm for UAV path planning. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, pp. 853–860 (2016)
7. Wang, Y., Bai, P., Liang, X., Wang, W., Zhang, J., Fu, Q.: Reconnaissance mission conducted by UAV swarms based on distributed PSO path planning algorithms. IEEE Access **7**, 105086–105099 (2019)
8. Li, B., Qi, X., Yu, B., Liu, L.: Trajectory planning for UAV based on improved ACO algorithm. IEEE Access **8**, 2995–3006 (2019)
9. Chen, J., Ye, F., Li, Y.: Travelling salesman problem for UAV path planning with two parallel optimization algorithms. In: 2017 Progress in Electromagnetics Research Symposium-Fall (PIERS-FALL), pp. 832–837. IEEE (2017)
10. Eun, Y., Bang, H.: Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm. J. Aircr. **46**, 338–343 (2009)
11. Peng, K., Wen, L., Li, R., Gao, L., Li, X.: An effective hybrid algorithm for permutation flow shop scheduling problem with setup time. Procedia CIRP **72**, 1288–1292 (2018)
12. Dixit, A., Mishra, A., Shukla, A.: Vehicle routing problem with time windows using meta-heuristic algorithms: a survey. In: Yadav, N., Yadav, A., Bansal, J.C., Deep, K., Kim, J.H. (eds.) Harmony search and nature inspired optimization algorithms. AISC, vol. 741, pp. 539–546. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-0761-4_52
13. Durillo, J.J., Nebro, A.J.: jMetal: a Java framework for multi-objective optimization. Adv. Eng. Softw. **42**, 760–771 (2011)