The Journal of Systems & Software 204 (2023) 111769

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Application of metamorphic testing on UAV path planning software☆

Lvyuan Wu [a], Zhiyu Xi [a,*], Zheng Zheng [a], Xiaoli Li [b]

[a] Beihang University, Beijing, China
[b] Institute of Systems Engineering Integration, Beijing, China

ARTICLE INFO

Article history:
Received 10 February 2023
Received in revised form 3 May 2023
Accepted 27 May 2023
Available online 5 June 2023

Keywords:
Software test
Path planning
Oracle problem
Metamorphic testing

ABSTRACT

Both the performance and reliability evaluation processes of the unmanned aerial vehicle path planning software rely on the determination of the correctness of the execution results of unmanned aerial vehicle path planning software. However, this task is hindered due to the testing oracle problem. In this paper, a framework is designed to overcome the oracle problem and verify the correctness of path planning software based on the grid searching algorithms. In this framework, a metamorphic testing-based method is proposed, and three operations-based metamorphic relations are proposed and proved towards the target software. While analysis of the software is conducted, the version with manually injected faults as well as the officially released version are both dealt with. It is shown that in the experimental results the injected faults can be effectively revealed by using the methods proposed in this paper. Besides, through the evaluation of different types of metamorphic relations, we find that the composed metamorphic relations have stronger fault detection capability compared to the individual ones.

1. Introduction

The testing problem has always been the bottleneck to improving the reliability of unmanned aerial vehicle (UAV) path planning software. With the increasing complexity of UAV missions, the task of path planning algorithm is also becoming more complex (Liu et al., 2021; Cheng et al., 2022). Therefore, a lot of attention has been attracted to the research on path planning algorithms. At the same time, increasing number of path planning algorithms are embedded into UAV software to realize automatic path planning (Zhao et al., 2018). However, the application scenario of path planning software is complex and highly integrated, which results in more hidden faults in the software (Qadir et al., 2021; Zhang et al., 2019). Therefore, software testing tasks are facing significant challenges. One of the reasons that made the testing of UAV path planning software difficult is the oracle problem, i.e., it is difficult to design reasonable test cases with expected output. Therefore, solving the problem of testing UAV path planning software is of great significance to ensure the reliability and safety of UAV mission, and can also promote the development of software testing technology (Lv and Yin, 2020; Yin and Li, 2006).

Path planning software and other software with Oracle problems share common characteristics, which is the lack of sufficient verification criteria for the test results due to the randomness of the algorithm (Nuortio et al., 2006). Xie proposed a model-specific metamorphic relation to test the implementation of a supervised classifier (Xie et al., 2009). With the enrichment of the functionality of UAV path planning software, the degree of integration of path planning algorithms and the scale of software keep increasing, which results in much heavier workload of the software testing process (Petritoli et al., 2018; Zhang et al., 2010). At the same time, there are random processes in some path planning algorithms, which incurs the possibility that output of these algorithms are different even under the same input (Zhang et al., 2018). All of these have brought great challenges to the test of UAV path planning software. In terms of the state-of-the-art, there is little research on the path planning software with the oracle problem considered, so it is difficult to evaluate the adequacy of the UAV path planning software test during the reliability and safety assess activities.

Metamorphic testing (MT) is a testing technology applicable where the algorithm is uncertain or the correctness of its output cannot be determined. It is understood that MT can alleviate the oracle problem in complex software, including path planning software (Segura et al., 2016). In order to alleviate the oracle problem in the testing of path planning software, this paper is based on the MT method (Bueno et al., 2014). Through analysis of attributes of the algorithm, metamorphic relations (MRs) are defined for the tested program (Bueno et al., 2014; Segura et al.,

☆ Editor: W. Eric Wong.

* Corresponding author.
E-mail addresses: ryanwoo@buaa.edu.cn (L. Wu), z.xi@buaa.edu.cn (Z. Xi), zhengz@buaa.edu.cn (Z. Zheng), z.xi@buaa.edu.cn (X. Li).

https://doi.org/10.1016/j.jss.2023.111769
0164-1212/

2018), then one or more pairs of input and output are assigned to be associated with each MR (Kanewala and Bieman, 2013). While the testing process is completed, analysis is performed to determine if the input and output satisfy the relevant MR, and if there are faults in the software (Xie et al., 2011).

As the application of MT algorithm onto UAV path planning software, a framework which facilitate the establishment of MRs for path planning software is designed. Based on the framework, attributes of MRs can be defined systematically. Then the MT is carried out based on the input/output of the software and the MRs. The graph-based route searching planning algorithm is chosen in this paper to realize path planning and six types of MRs are proposed according to attributes of the algorithm. MT is then performed under different scenarios. It is shown that most of injected faults or native faults in the experimental subjects can all be killed using the proposed methods.

The rest of this paper is organized as follows. Background of the research work is introduced in the second section, including the description of the path planning problem and relevant algorithms. The third section is dedicated to the design the metamorphic relation framework according to the algorithm attributes. Then, details of the experiment are provided in Section 4, while analysis to the experimental results is given in Section 5. Section 6 concentrates on the conclusion and prospect.

2. Background

2.1. The UAV path planning

The UAV path planning refers to the process to plan for the optimal or a feasible trajectory connecting the starting point and the target point for UAV navigation in a feasible space (Pitre et al., 2012). It is required that obstacles along the route are avoided with all other constraints satisfied so that the UAV can reach the target safely along the planned route (Liu et al., 2014). Scope of this paper is restricted to the path planning task in two-dimensional space for simplicity purposes while the results can be easily extended to the cases in three-dimensional space (Özalp and Sahingoz, 2013).

The elements of the path planning algorithm include the starting point S, the target point T and obstacles in the space. Obstacles are modeled as threat areas described by the UAV intrusions. Mathematically, a threat area can be described as a disk featured by its radius and location of the center. In this paper, it is assumed that the value of the cost functions that are associated with mid-points of routes are equal everywhere within such disk. Therefore, the problem is simplified into looking for the shortest route from the starting point S to the target T while avoiding the disks. Based on such assumption, the cost function can be substituted into the route calculation algorithm once defined to facilitate the optimization operation. Result of a path planning algorithm can be expressed as a series of vectors containing the coordinates of trajectory points to be experienced by the UAV, which can be described as $Path = \langle S, p_1, p_2, \ldots, p_n, T\rangle$, p_1, p_2 representing the first and second points on the path respectively,where n represents the number of mid-points involved in the trip from S to T (Zhang and Yin, 2018; Gordon et al., 2012).

At present, many path planning algorithms have been proposed which can find feasible routes from the starting point to the target point automatically (Bast et al., 2016). According to whether the route obtained is an optimal one, these algorithms are categorized as optimal path planning algorithms or feasible path planning algorithms (Roberge et al., 2012). A feasible path planning algorithm can find feasible routes for the problem given, while the optimal path planning algorithms can find a path with the lowest cost based on an explicitly defined cost function. In addition, path planning algorithms can also be categorized as those based on graph search and those based on sampling (Yang et al., 2020).

Optimal path planning algorithms based on graph search adopted in this paper is a typical kind of path planning algorithm (Nikolova et al., 2006). The main task of such algorithm is to find the path(s) from the starting point to the target point in a discrete space, such that performance requirements for the UAV mission are all satisfied while minimizing a comprehensive cost function.

In an optimal path planning algorithm based on graph search (Delling et al., 2009), the continuous space is discretized into a graph composed of node-set p_i and edge set e_{ij} where p_i represents the ith node and e_{ij} represents the edge connecting node p_i to p_j. Each edge $e_{nm} \in \{e_{ij}\}$ is associated with a value representing the cost from node p_m to p_n, which can be depicted as the distance between the two nodes. Node p_n is called the successor of p_m if they are mid-points of a same route and n, m. The starting point S and target point T should also be two nodes in p_i. Then, the task of the graph search algorithm is to search for a series of subsequent nodes from node S until T. The optimal path should subject to Eq. (1):

$$MinC(S,T) = \sum_{0<i\leqslant n+1, j=i+1}^{n} \left|e_{ij}\right| \tag{1}$$

Fig. 1 shows an example of a path planning triplet problem, where S is the starting point, T is the target point, and O is an obstacle. There are n feasible routes from S to T, marked as $Path_1$, $Path_2$, …, $Path_n$. In addition, $Path_0$ is used to represent the route connecting S and T with a straight line. If there is no threat or obstacle, $Path_0$ is also the shortest route (i.e., route with minimum cost). In this example, if C_i is used to represent the cost of ith path $Path_i$ ($i = 1, 2, 3, \ldots, n$), then the task of the graph-based search path planning algorithm is to find mid-points in $Path_j$ so that C_j is the minimum among $\langle C_1, C_2, \ldots, C_n\rangle$. If there are multiple routes with minimum cost, the algorithm will randomly select one of them to return.

Such algorithm can also be expressed as a function $P(S, Env, T)$, which is defined as follows: $P(S, Env, T) = Path_j | C_j = min(C_1\, C_2\, \ldots, C_n)$, where Env represents the environmental factors containing threat areas, and the output of the algorithm is the sequence of mid-points with the lowest cost.

Based on the idea above, various algorithms can be designed. This paper mainly studies three typical path planning algorithms based on graph search, namely the A* algorithm (Soltani et al., 2002), L* algorithm (Niewola and Podsedkowski, 2018), and Jump Point algorithm (Gijbels et al., 1999). The reason for choosing these three algorithms is that they are regraded the most representative in the field of UAV path planning. The A* algorithm is the first path planning algorithm ever proposed, while L* and Jump Point are proposed in recent years to improve A* in terms of computational complexity and pruning strategy. These three algorithms have found extensive applications in this field. However, it should be noted that the methods and techniques proposed in this paper also apply to other path planning algorithms based on graph search. There is still a lot of research space for testing the UAV path planning software, so we apply MT method to the UAV path planning software to alleviate the problem of testing the UAV path planning.

2.2. Metamorphic testing

It is understood that to obtain accurate test result and perform coverage analysis is quite a routine job in many software testing tasks (Myers et al., 2011). However, there are two problems in the

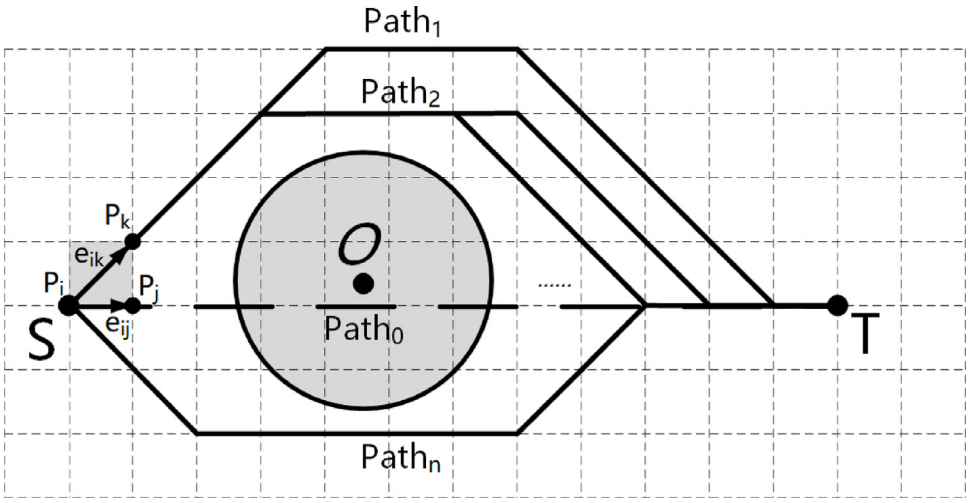

Fig. 1. An example of a UAV path planning probl.

testing of UAV path planning software which make the situation much more complicated, i.e., the oracle problem and the selection of reliable test cases. Oracle problem refers to the scenario that in the process of software testing, it is difficult to determine whether the results based on given test cases are correct (Rovira-Sugranes et al., 2022). And the problem of reliable test set focuses on how to select test cases from the given case set such that faults in the software can be successfully located through the test (Stentz, 1997). In terms of UAV path planning software, it is normally difficult to evaluate the optimality of a path as well as to cover all test cases due to the large volume of the case set. If further investigation is conducted to the mechanism of these two problems, it is found that the oracle problem in the object of the software to be tested is precisely the cause of the difficulty in choosing a reliable test set (McMinn, 2011). The reason is that if there are no explicit rules to perform evaluation to the test result, it will result in the absence of suitable criteria in the selection of effective test cases (Chen et al., 2003).

As a method to alleviate oracle problems, MT has attracted extensive attention in the field of software testing. The idea of MT is that one or more groups of new test inputs can be obtained from the successful test cases by designing the metamorphic relation between them (Hui and Huang, 2013). Then these new inputs are used as test cases with the resulting output compared to those as expected, which are obtained based on the successful test cases and the metamorphic relation. If the new output does not match the expectation, the indication is there may be defects in the program.

In the MT framework, the first thing to do is to find the attributes associated with the input and output through analysis to mechanism of the software and rigorous proof. Then, links should be built between these attributes in the form of explicit mathematical relations. Afterwards, predictions on new outputs can be made considering the relations between the modified test cases and the original ones. MRs are explicit representation of these attributes and forms the core of metamorphic tests. While MRs are established, initial test cases and generated test cases (also called "subsequent test cases") are both executed and verification is performed to show whether two groups of outputs meet the corresponding MRs (Ahmed et al., 2016).

Through MT, oracle problems can be effectively alleviated. Because even if the designer has no information about whether the results obtained from the initial test cases are correct, the software reliability can still be analyzed through comparison between the results obtained based on initial test cases and the metamorphic test cases. Thus, the problem of making direct judgment of the test results is circumvented.

However, the difficulty in MT related algorithms lies in the identification of MRs, which requires professional knowledge or sufficient understanding of software design requirements. For the test of the UAV path planning algorithms/software, it is essential to analyze the nature of the UAV path planning task. After identifying and establishing effective MRs, software testing can be carried out.

3. Methods

3.1. MT for path planning algorithms based on graph search

For the path planning of UAV, the input to the algorithm is the information of each part of the test scene, including the coordinates of the starting point S and target point T, as well as the known and unknown threats in the scene. Meanwhile, the output from the algorithm is the length of the shortest route between the starting point and the target point subject to the test input. In general, the optimal solution of the path planning problem cannot be accurately determined before the execution of software. However, if the MR related to the expected results can be obtained through the analysis of the path planning algorithm, the correctness of software output can be determined by checking on the MR between outputs.

A basic framework for the MT process is designed as shown in Fig. 2. A mathematical model of the path planning process is established in order to improve the clarity of the explanation to the software testing process using MT principle. A basic framework for the MT process is designed as shown in Fig. 2. A mathematical model of the path planning process is established in order to improve the clarity of the explanation to the software testing process using MT principle.

Firstly, the source test case and its follow-up test case are generated for applying MT. Suppose a software $P(x)$ is the implementation of a path planning algorithm $f(x)$, where x denotes a test scenario in Fig. 3.

Let $X = \{x_1, x_2, \ldots, x_n\} (n \geq 2)$ denote a set of test scenarios, where x_i is further defined as a tuple such that Eq. (2).

$$x_i = (S, Env, T), i = 1, 2, \ldots, n \tag{2}$$

where, Env contains two important elements, i.e., the set of threat regions and the coordinate system.

It is observed that each set of test scenario is composed of an initial test scenario and a corresponding subsequent test

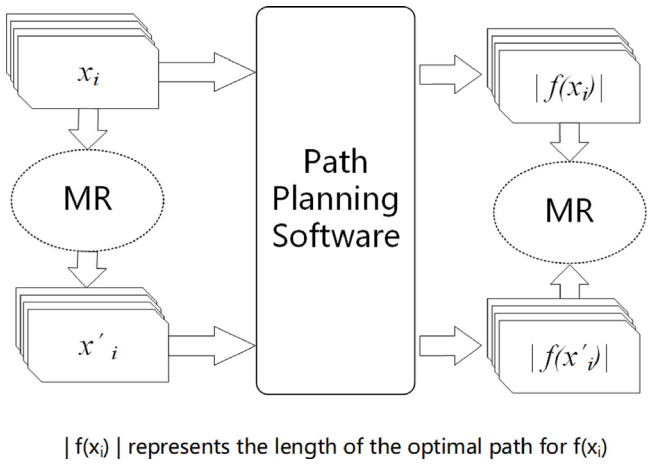

Fig. 2. Overview of conducting MT for a UAV path planning algorithm.

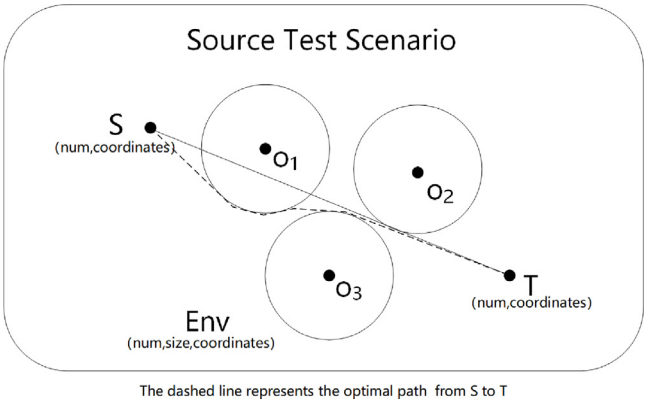

Fig. 3. Elements of test scenario.

scenario. The subsequent test scenarios are transformed from the initial test scenarios. In order to obtain a series of subsequent scenarios based on the initial scenario, a series of MRs which help to obtain expected results of the path planning problem is established (Burnstein, 2006).

Although unexpected results from MT are neither sufficient nor necessary condition of the existence of faults in the software, it can be used as a reference in locating software faults if there is any. Further tests may be performed to verify the MT that produces inconsistent results to find out whether faults do exist in the software.

3.2. Metamorphic relations design and tests

3.2.1. The framework for designing of metamorphic relations

Scenery information in path planning algorithm includes the coordinate system of the scene, information of threats with respect to the coordinates, i.e., flight environment, and other information related to the UAV mission (Wei et al., 2013). Therefore, there are three elements involved in the whole scenario, i.e., the starting point S, the environment Env and the target point T. For S and T, two attributes are obtained through analysis for each of them, i.e., number of points and coordinates of points. And for threats modeled as circular areas in a two-dimensional flight scene, three attributes are obtained, i.e., "number of threats", "coordinates of the center" of each threat area and "radius" of each threat area. A UAV mission scenario is reflected in Fig. 3. According to the scene division and attributes extracted, the definition framework of metamorphic relations for the UAV path planning algorithm can be established (Birk et al., 2011).

In our framework, each MR contains two elements, i.e., the relation R_r between the input test cases and the relation R_f between the corresponding outputs. Each input variable of path planning algorithm includes three elements, i.e., S, Env, T. Relation R_r between the input test cases can be obtained after manipulating some attributes of these elements. Then relation R_f between the corresponding outputs can be obtained by comparing the lengths of the shortest route before and after the transformation defined by the metamorphic relation, i.e., $|f(x_i)|$ and $|f(x'_i)|$. In this way, a two-tiers framework of the MR for path planning software including an input layer and an output layer can be designed, as shown in Fig. 4. For each atomic operation, we modify the starting position, terminal position, or environment to generate.

Next, a framework to design metamorphic relations for a path planning program based on graph search is proposed. Based on the three elements of the initial test scenario and their attributes, three basic operations with corresponding operators are designed to generate subsequent test scenarios. Details of the operations are as below.

Starting point ($O-S$): Make changes to the initial test case by changing the properties of the starting point. Two atomic operations are defined as below

- Starting point operation 1-1: Insert starting point (Insert S). Inserting one or more new starting points into the initial test scenario is equivalent to changing the number of points attribute of the starting point, so that the UAV can begin its mission from one out of a number of optional starting points in the subsequent test scenario.
- Starting point operation 1-2: Move starting point (Move S). In the initial test scenario, change the coordinates attribute of the starting point so that the UAV can start from a different point in the subsequent test scenario.

Environment ($O-Env$) :Make changes to the initial test case by changing the environmental information attribute. Three atomic operations are defined:

- Environment operation 2-1: Add/Remove environmental threat (Add/Remove Env). Adding or removing threats in the initial test scenario is equivalent to changing the number of threats attribute of the environment such that there are a different number of threats in the subsequent test scenario.
- Environment operation 2-2: Move environmental threats (Move Env). By changing the coordinates of the center of a threat, movement of the environmental threat can be achieved.
- Environment operation 2-3: Increase/Decrease the size of environmental threat (Increase/Decrease Env). Size of environmental threat can be adjusted by changing its dimension parameters. In the two-dimensional plane, Increase/Decrease Env means essentially to change the radius of a threat modeled as a circle.

Target point ($O-T$) : Make changes to the initial test case by changing the attributes of the target point. Two atomic operations are defined as below:

- Target point operation 3-1: Insert target point (Insert T). Inserting new target points into the initial test scenario is equivalent to changing the number of points attribute of the target points, so that the UAV can complete its mission at one among a number of target points in the subsequent test scene.

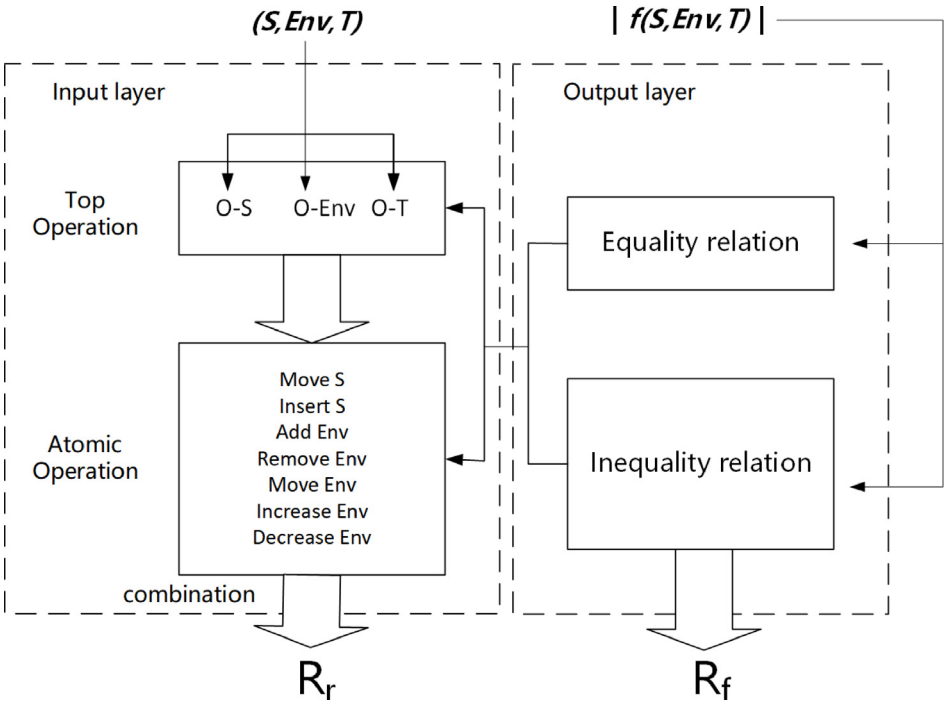

Fig. 4. Two-tier framework of MRs.

- Target point operation 3-2: Move target point (Move T). In the initial test scenario, change the coordinate attribute of the target point so that the UAV can target at a different location in the subsequent test scenario.

Based on the framework shown in Fig. 4, seven kinds of operations are defined to generate subsequent test scenarios based on initial scenarios. Then, six metamorphic relations can be defined, which are further categorized as point-related, threat environment-related, and geometric transformation-related according to arguments the transformations work on.

3.2.2. Definition of metamorphic relations

Point-related MRs: this kind of MRs are mainly used to manipulate the attributes of the starting point or target point ($O-S$ or $O-T$), which were firstly proposed and tested in one of our previous work in Zhang et al. (2019). A single metamorphic relation refers to a metamorphic relation in which there is only one atomic operation, that is, when a metamorphic relation is defined, one top operator is selected at first, and then a corresponding bottom operator is selected as the definition of the metamorphic relation. A composite metamorphic relation refers to the situation where two or more atomic operations are relative to a single metamorphic relation. Illustration to these two MRs are given below:

- ***MR-1.1*** Swapping the starting and the target points should not change the length of the planned path, denoted as (Move S & Move T). Formally, if the initial test case is $x = (S, Env, T)$, the subsequent test case is $x' = (T, Env, S)$, one should have $|f(x)| - |f(x')| = 0$.
- ***MR-1.2*** Dividing the source path planning problem into two sub-problems by inserting an intermediate point, denoted as (Insert T/Insert S). Let P denote a point on the optimal path for $x = (S, Env, T)$, $x_1 = (S, Env, P)$, and $x_2 = (P, Env, T)$, then should have that $|f(x_1)| + |f(x_2)| = |f(x)|$.

Threat environment-related metamorphic relations: manipulating the threat environment is equivalent to manipulating the attributes of threats ($O-Env$). Two kinds of environment-related MRs are defined in this paper, which are:

- ***MR-2.1*** Make changes to the threat attributes without affecting the optimal route (Add/Remove Env, Increase/Decrease Env). There are many ways to change attributes of threats, such as increasing or decreasing the number of threats (Add/Remove Env), and adjusting the radius of a threat (Increase/Decrease Env). Suppose the initial test case is $x = (S, Env, T)$, aim of MR-2.1 in this paper is to change the threat attributes so that the threat is closer to the optimal route subject to $x = (S, Env, T)$ such that the optimality of the original optimal route is maintained. Denote this original optimal route as $f(S, Env, T)$, then $f(S, Env, T)$ should also be the optimal route subject to the subsequent test cases $x' = (S, Env', T)$. It is then straightforward that $|f(x)| - |f(x')| = 0$.
- ***MR-2.2*** Make changes to the attributes of a threat area to cause changes to the optimal route (Add/Remove Env, Increase/Decrease Env). MR-2.2 is similar to MR-2.1 in terms of their arguments. But the aim of MR-2.2 is to change the attributes of a threat close the original optimal route, so that the optimal route in the subsequent test case should be different from the original one in an expected fashion. For example, by adding a threat or increasing the radius of an existing threat, $|f(x)| - |f(x')| > 0$ is expected when there is unique shortest route in the initial test case. However, if there are multiple optimal routes in the initial test case, the optimal route remains the same after MR-2.2, i.e., $|f(x)| - |f(x')| = 0$, unless Add Env or Increase Env is performed to all optimal routes. Combining the two above cases, output relation $|f(x')| - |f(x)| \geq 0$ is achieved. Similarly, if one wants to have a shorter optimal route in subsequent test case by deleting a threat or reducing the radius of a threat near the original optimal route, there are also two cases for the results of subsequent test cases. And $|f(x')| - |f(x)| \leq 0$ should be observed.

Each of MR-2.1 and MR-2.2 can be extended to two metamorphic relations, where MR-2.1.1 and MR-2.1.2 keep the original result of optimal route unchanged, while MR-2.2.1 and MR-2.2.2 result in longer shortest routes. MR-2.1 and MR-2.2 can be easily designed according to the user's expectations. It can be observed that MR-2.2 is more complex than MR-2.1 because the number of original optimal routes in the has to be considered. We keep the operators Add Env and Increase Env to implement this metamorphic relation.

Geometric transformation-related metamorphic relations: geometric transformations of the whole scene actually change all three elements of the scenario ($O-S\&O-Env\&O-T$). Two simple and effective geometric transformation-related MRs are described below.

- ***MR-3.1*** Scene rotation (Move S & Move Env & Move T). If we rotate the source scene a certain angle, the planned path length should be unchanged, because the position of the starting point, the target points, and the threats are relatively unmoved. This operation changes the coordinate attributes of T and S, and the coordinate of center attribute of the environmental threat. For the initial test case $x = (S, Env, T)$, the subsequent test case is $x' = (S', Env', T')$, with $|f(x)| - |f(x')| = 0$.
- ***MR-3.2*** Scene symmetry (Move S & Move Env & Move T). If a symmetric transformation is performed about a straight line to the whole initial test scenario while keeping the relative positions of the starting point, target point, and environmental threats unchanged, then the result of path planning will remain unchanged. Similar to MR-3.1, the coordinate attributes of T and S, and the coordinate of center attribute of the environmental threat are changed with MR-3.2. Let $x = (S, Env, T)$ and $x' = (S', Env', T')$ denote the initial and subsequent test cases, respectively. Then, $|f(x)| - |f(x')| = 0$ should hold.

Only when all MRs mentioned above are the necessary properties of a path planning algorithm, they can be used for validation. Next, validity for each MR will be proved.

3.2.3. Proof of correctness of metamorphic relations

To prove the correctness of the MRs introduced previously, some definitions relevant to path planning problem are provided first. Let the UAV path planning triplet problem be denoted as (S,Env,T), the set containing all feasible routes of the UAV be written as $Path_i$(S, Env, T), i=0,1,2,…,n, and the length $Path_i$ be referred to as$|Path_i$ (S,Env,T)|. The optimal route obtained by an algorithm is denoted as $f(S, Env, T) = \{Path_i | Path_i(S, Env, T)|\}$, whose length is denoted as $|f(S, Env, T)|$. Immediately, Eq. (3).

$$\forall i, i = 0, 1, 2, \ldots, n, |f(S, Env, T)| \leq |Path_i(S, Env, T)| \tag{3}$$

Proof of **MR-1.1**: In the two-dimensional plane, the shortest route length from the starting point to the target point is $|f(S, Env, T)|$, and the optimal route from the target point to the starting point is $|f(T, Env, S)|$. Suppose $|f(S, Env, T)| \neq f(T, Env, S)|$, then if $|f(S, Env, T)| > |f(T, Env, S)|$, the condition that $f(S, Env, T)$ is the optimal route for the original test case is contradicted. Else if $|f(S, Env, T)| < |f(T, Env, S)|$, then the condition that $f(T, Env, S)$ is the optimal route for the subsequent test case is contradicted. Therefore, $|f(S, Env, T)| \neq |f(T, Env, S)|$ is invalid, i.e., there is $|f(S, Env, T)| = |f(T, Env, S)|$ or $|f(S, Env, T)| - |f(T, Env, S)| = 0$.

Proof of **MR-1.2**: Take a point P on path $f(S, Env, T)$, then according to Eq. (4), ∀ i, i=1,2,…,n

$$0 \leq |f(S, Env, P)| \leq Path_i(S, Env, P) \tag{4}$$

$$0 \leq |f(P, Env, T)| \leq Path_i(P, Env, T) \tag{5}$$

$$Path_i(S, Env, P) + Path_i(P, Env, T) \in \{Path_i(S, Env, T)\} \tag{6}$$

From Eqs. (4) and (5), it can be further obtained that for ∀ i, i=1,2,…,n

$$f(S, Env, P) + |f(P, Env, T)| \leq |Path_i(S, Env, P)| + \left|Path_j(P, Env, T)\right| \tag{7}$$

$$f(S, Env, P) + f(P, Env, T) \leq |f(T, Env, S)| \tag{8}$$

Furthermore, from Eq. (8) one has:

$$f(S, Env, P) + f(P, Env, T) \in \{Path_i(S, Env, T), i = 0, 1, 2, \ldots, n\} \tag{9}$$

From Eqs. (3) and (9):

$$f(S, Env, P) + f(P, Env, T) \geq |f(T, Env, S)| \tag{10}$$

It can be proved that from Eq. (8), 9 and 10 that:

$$f(S, Env, P) + f(P, Env, T) = |f(T, Env, S)| \tag{11}$$

Proof of **MR-2.1**: Suppose in the initial test case, the optimal route obtained is $f(S, Env, T)$, and the optimal route after performing MR-2.1 is $f(S, Env', T)$. As the aim of MR-2.1 is to let a threat move closer to the optimal route without affecting the result of path planning, then both $|f(S, Env, T)| \geq |f(T, Env', S)|$ and $|f(S, Env, T)| \leq |f(S, Env', T)|$ hold. Therefore, one has $|f(S, Env, T)| = |f(S, Env', T)|$ hold.

Proof of **MR-2.2**: Suppose in the initial test case, the optimal route obtained is $f(S, Env, T)$, and the optimal route after performing MR-2.2 is $f(T, Env', S)$. If MR-2.2 is carried out to increase the optimal route, that is $|f(T, Env', S)| < |f(S, Env, T)|$, then this conflict with the condition that $f(S, Env, T)$ is an optimal route. Therefore, $|f(T, Env', S)| \geq |f(S, Env, T)|$ should be true. Similarly, while MR-2.2 is carried out to decrease the length of the optimal route, it can be shown that $|f(T, Env', S)| \leq |f(S, Env, T)|$ should be true.

Proof of **MR-3.1**: Suppose in the initial test case, the optimal route obtained is $f(S, Env, T)$, and the optimal route obtained after rotating the input scene by an angle a is $f(S', Env', T')$. While MR-3.1 is performed, the whole scene is rotated around the same point at a same angle, which means that the relative positions of the starting point, threats, and the target point have not changed. Assume $|fa(S', Env', T')| = |f(S, Env, T)|$ is not true, then one of them should not be the optimal route in its associated test scenario, resulting in a contradiction. Therefore, the $|fa(S', Env', T')| = |f(S, Env, T)|$ statement is true.

Proof of **MR-3.2**: Suppose that in the initial test case, the optimal route obtained is $f(S, Env, T)$, and the optimal route after a symmetric transformation about a straight line l is $fl(S', Env', T')$. As illustrated in the proof of MR-3.1, since the relative position of the whole scene remains unchanged after MR-3.2, there is $|fl(S', Env', T')| = |f(S, Env, T)|$.

It should be noted that all of the above proofing processes are specific to MR. Presently, a fast and automated method to generate MR is unavailable.

3.2.4. Analysis of the nature of the metamorphic relations

Classification of MRs previously discussed will be carried out from different perspectives in this subsection. Results of classification are recorded in Table 1, which reveal a number of properties of metamorphic relations. Each MR is a description of manipulations on the initial test scenario. Therefore, MRs are categorized according to their relevance to the initial output

Table 1
Properties of metamorphic relations.

MR	Attributes			
	Input relation			Output relation
	Relevant to initial output	Relevant to user expectation	Argument	Relation
$MR-1.1$	N	N	S, T	E
$MR-1.2$	Y	N	T	E
$MR-2.1$	Y	Y	Env	E
$MR-2.2$	Y	Y	Env	NE
$MR-3.1$	N	N	S, Env, T	E
$MR-3.2$	N	N	S, Env, T	E

(if the results of optimal route are the same before and after the MR operation) and relevance to the designer's expectations. In Table 1, "Y" and "N" are used to represent "relevant" and "irrelevant" respectively. "E" and "NE" indicate that the result of performing associated MR is to be described in the form of an equality or inequality respectively (Liu et al., 2013).

Among all the MRs, MR-2.1 and MR-2.2 are more complex than others. Because in the implementation of these two MRs, especially MR-2.2, selections of specific manipulations for the MR to realize are to be made and relevant settings are to be determined, in which the expectations of the designer are to be taken into account. For example, in MR-2.2, whether there are unique or multiple optimal paths in the initial test scenario has to be considered in addition to the selection of specific manipulation to carry out. In order to further clarify the mechanisms of MR-2.1 and MR-2.2, they are further categorized as MR-2.1.1& MR-2.1.2 and MR-2.2.1& MR-2.2.2 respectively. The purpose of MR-2.2.1& MR-2.2.2 is chosen as to increase the length of the optimal route as an example.

4. Experimental setups

In this section, A* algorithm, L* algorithm, and Jump Point (JP) algorithm are taken as the research objects under framework. At the same time, an platform to automatically generate sets of test scenarios have been established such that batch testing can be realized while experiments are carried out (Harabor and Grastien, 2014). In this section, typical path planning software are tested through metamorphic relations. From a large number of algorithms integrated in the software, three representative ones, i.e., A*, L*, and JP algorithms are chosen for testing purpose.

4.1. Scenario design

The test scenarios adopted in this paper include three elements, i.e., the starting point S and target point T of the UAV mission, and the information of threat areas in the plane. Also, according to the scenarios to be tested, a set of initial test scenarios containing randomly generated inputs are designed. Without loss of generality, it is assumed that for each task, the coordinates of S, T, and center of the threat areas are randomly distributed in the two-dimensional plane. The ranges of horizontal and vertical coordinates are both [0, 100]. The radius of each threat region is within [10, 20], and the maximum number of threats in each scenario is 5. Besides, two assumptions are made such that the test scenarios are more reflective of UAV path planning tasks, i.e., there exists at least one feasible route from S to T, and there must be at least one threat encountered by each feasible route. In the experiment, all metamorphic relations are tested using each path planning algorithm with the same initial test scenario.

4.2. Implementations of metamorphic relations

(a) Point related operations. $MR-1.1$ is achieved by exchanging the coordinates of S and T. $MR-2.1$ can be achieved by selecting a point on the existing optimal route to be the center of the additional threat.

(b) Threat related operations. In $MR-2.1$ and $MR-2.2$, although lengthening and shortening of the optimal route are two opposite ways of affecting $f(S, Env, T)$, expected results of performing MRs which tend to increase or decrease $f(S, Env, T)$ should be the same, i.e., to end up with $|f(x)| - |f(x')| = 0$. Therefore, MRs which tend to increase $f(S, Env, T)$ are implemented in this section as examples. Results of verification of effectiveness of these MRs should also apply to MRs which tend to decrease $|f(S, Env, T)|$. In the process of implementing $MR-2.1.1$ and $MR-2.2.1$, a new threat is to be inserted into the scene, and the radius of inserted threat can be randomly generated within a certain range. Therefore, determination of the center of the threat to be inserted seems to be the real critical issue. Since the algorithm will search for mid-points of the optimal path from S to T in the whole plane, points on both sides of the straight-line connecting S and T are to be covered by the algorithm. In order to fully explore the effectiveness of the MRs, center of the inserted threat is located on the side of the straight-line connecting S and T containing the longest line segment of the optimal route obtained for the initial test scenario, as shown in Fig. 5. Suppose P_1 denotes the optimal route obtained for the initial test scenario, P_0 is the straight-line connecting S and T, and it is below all line segments of P_1. Therefore, the center of the inserted threat region denoted as O_3 is located below P_1. Additionally, in $MR-2.1.2$, an existing threat centered at O_1 which is located slightly away from P_1 should be chosen and its radius is to be increased such that it is closer to P_1. Similarly, for $MR-2.2.2$, a threat on the same side of P_1 with P_0 will be selected. In Fig. 5, the threat centered at O_2 is selected and its radius is increased.

(c) Geometric transformation-related operations. It is easily understood that performing geometric transformations to S and T simply refers to performing transformations to associated coordinates. Meanwhile, since environmental threats are modeled as circular regions in the space, geometric transformations essentially work on coordinates of their center points. That means, to conduct geometric transformations of the whole scene actually end up with playing around several critical coordinates therein.

4.3. Automatic generation of test scenarios

It is widely understood that test cases with higher quality contribute to making MT faster and more efficient in detecting faults in software. There are many existing techniques to generate initial test cases for MT algorithms, among which random generation method is the most commonly used. The idea of random generation is to randomly generate a coordinate system and variables in this system in a sequential manner, which has been proved to be quite effective. Therefore, it is embedded in the platform which generates initial test cases in this paper. Moreover, in order to improve the efficiency of MT, the following two constraints on the basis of the random generation method are imposed to ensure that scenarios with certain complexity and rationality are generated.

- Complexity constraint. There is at least one threat on the optimal route to avoid case that the optimal route is a line from one starting point to a target.

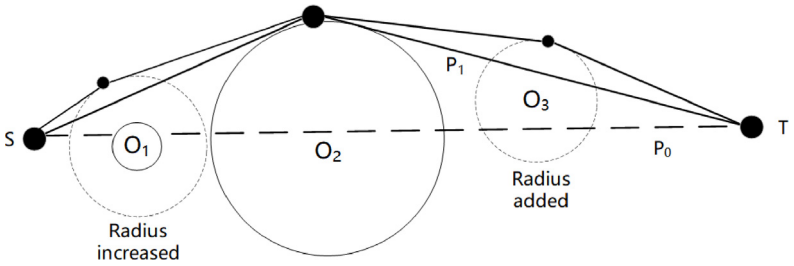

Fig. 5. Insertion and enlargement of threat region.

- Rationality constraint. For each generated scenario, the starting point must be connected with the target point to ensure that the algorithm can find at least one optimal route.

Each generated scenario includes one starting point, one target point, and flight environment including threat areas. Each threat is modeled as a disk in the two-dimensional plane. Via setting the radius of threat areas and the number of threats, the process of randomly generating an initial test scenario under complexity constraint and rationality constraint above can be depicted as follows:

(a) Generate an integer $p \in [n_1, n_2]$ as the number of threats in this scenario;
(b) A threat with radius $r \in [n_1, n_2]$ is randomly generated within the specified range of a two-dimensional coordinate system, then increase the number of existing threats q by 1, i.e., $q = q + 1$;
(c) Go to (d) if $q = P$, go back to (b) otherwise;
(d) Randomly generate a point S as the starting point in the two-dimensional coordinate system;
(e) Return to (d) if S is within a threat area, go to (f) otherwise;
(f) Randomly generate a point T as the target point in the two-dimensional coordinate system;
(g) Return to (f) if T is within a threat area, go to (h) otherwise;
(h) Go to (i) if there is a feasible route from S to T, return to (f) otherwise;
(i) Go to (j) if the straight-line connecting S and T goes through at least one threat areas, return to (f) otherwise;
(j) Output the generated test scenario;

For each initial test scenario, the corresponding subsequent test scenario will be obtained through transformations involved in a certain MR. Each MR consists of two parts, i.e., the input relation R_r and the output relation R_f. R_r could be used to generate subsequent test scenarios for MRs which only takes attributes of the initial test case as arguments. While for MRs which are based on the output of the initial test case, the initial test scenario and its output should both be given to the MR to obtain subsequent test scenarios. Meanwhile, R_f can be used to obtain the result of test, i.e., if the test case pass the test depends on if relation between initial and subsequent outputs matches the expectation defined in the MR. Mechanism of MT can be found in Fig. 6.

Next, the criteria to determine if the test case pass the MT will be described. Because output from algorithms based on graph search are discrete routes with unity step size, so if the output $|f(x)|$ of the two routes is close enough, then these two routes are regarded "the same". Generally speaking, bias values for $MR-1.1$, $MR-1.2$, $MR-3.1$, and $MR-3.2$ are set to 0.5. In other words, if $|f(x)| - |f(x')| < 0.5$, such a scenario will be labeled as pass. The bias value is also set to 0.5 for $MR-2.1$, considering that specific threat attributes are manipulated in a way that would leave the original optimal route left unchanged. For $MR-2.2$, if $|f(x)| \leq |f(x')|$, then such a test scenario is marked as pass. A failed test case will be double checked to confirm if it violates a tester's expectation.

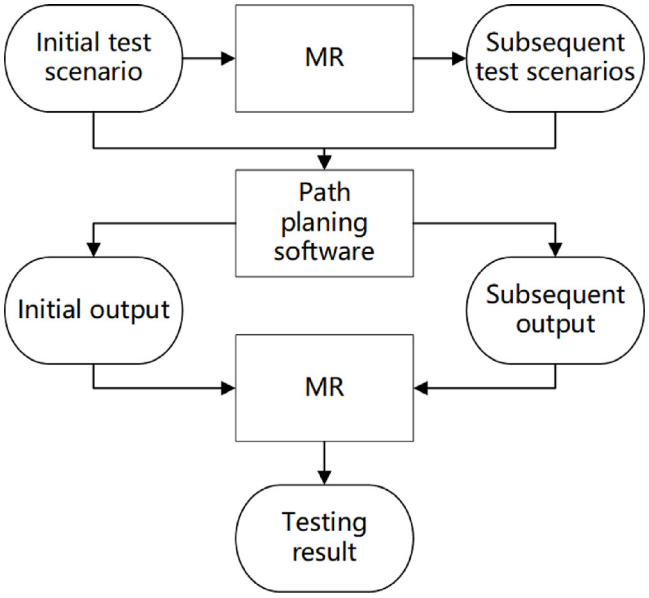

Fig. 6. Metamorphic testing process.

5. Results

5.1. Analysis results of the three algorithms

By testing the designed six groups of metamorphic relations, a total of 100 test scenarios have been generated and three path planning algorithms have been implemented under each of them. Then test results will be analyzed in this section.

It is noticed that severe oscillation is exhibited in the error of test results while MR-1.1 is applied. Therefore, to guarantee that the most reasonable scenarios pass the tests, the threshold in MR-1.1, MR-1.2, MR-2.1.1, MR-2.1.2, MR-3.1, and MR-3.2 (in which equality between $|f(x)|$ and $|f(x')|$ are expected) is set to 0.5. It is shown that the value of threshold is not to be concerned with while MR-2.2.1 and MR-2.2.2 are applied as the inequality structure of the expected relation between $|f(x)|$ and $|f(x')|$ make the algorithm more tolerant to errors.

While value of the threshold has been determined, results of 100 tests under each metamorphic relation from A*, L* and Jump Points algorithm are recorded. Summary of analysis to these test results are provided in Table 3. They are represented by the percentage of numbers of test scenarios which pass the test out of the total of test scenarios, i.e., 100% means that all scenarios have passed the test with no faults reported. A lower percentage

Table 2
Analysis to test results.

MRs	Test results from algorithms			
	A*		L*	Jump points
	Original version	Defect repaired version		
MR-1.1	33%	99%	100%	100%
MR-1.2	69%	100%	100%	100%
MR-2.1.1	61%	100%	100%	100%
MR-2.1.2	66%	91%	94%	95%
MR-2.2.1	77%	100%	100%	100%
MR-2.2.2	84%	98%	99%	98%
MR-3.1	100%	100%	100%	100%
MR-3.2	100%	100%	100%	100%

indicates a higher possibility of existence of faults in the software as more cases failed the test.

Through the tests using geometric transformation-related MRs, it is found that there may be faults in the implementation of A* algorithm. So, the test is repeated after the software is gone through and faults therein are fixed, with results summarized in Table 2. In addition, an in-depth inspection will be conducted to the test cases that fail the test to investigate whether expectations set by the MRs are really violated or there are other reasons which incurred the failure.

The experimental statistical results show that the implementation of A* algorithm violates some MRs both before and after defects are eliminated. It is noticed that in the version before the correction, a large number of scenes failed the test while MR-1.1 is applied, while fewer failures are reported under MR-1.2, MR-2.1, and MR-2.2, whereas only a few scenes failed the test under MR-1.1, MR-2.1.2, and MR-2.2.2. In addition, in the L* and Jump Points algorithm, some test cases failed under MR-2.1.2 and MR-2.2.2. An in-depth investigation of relevant test cases is conducted based on information of these violations collected from the experiment. It is found out that violations in the modified A* algorithm, L*, and Jump Points are incurred by specific characteristics of associated scenarios rather than errors in the implementation. In addition, it is found that for all algorithms, all test cases can pass the tests if MR-3.1 or MR-3.2 are adopted, which means no fault was found in the program.

5.2. Evaluation and analysis of MRs' effectiveness

Through the test of different single defect versions of the algorithm, it is found out that application of MRs can realize detection of all types of faults although with different passing rates. Normally, results of MT belong to one of following three categories:

(a) There are no faults in the implementation of the algorithm. The initial and subsequent test cases have been both executed correctly with their outputs matching the expectation imposed by corresponding MR, which indicates the test case passes the test.
(b) There are faults in the implementation of the algorithm. Results of the initial and subsequent test cases do not match the expectation imposed by corresponding MR. The test case does not pass the test with faults detected.
(c) There are faults in the implementation of the algorithm. And results of the initial and subsequent test cases match the expectation imposed by corresponding MR with faults undetected.

Normally, the metamorphic relations should be constructed to increase the proportion of results belonging to category (a) and (b), while decreasing the proportion of those belonging to category (c). This means that when there are faults in the implementation of the program, an MR with lower passing rate of test cases is more desirable. On the other hand, MRs which can detect more faults are regarded more effective. As a result, the combination of passing rate of test cases and the number of defects revealed should be reflected in the evaluation criteria of MRs.

Table 3
Evaluation values using different algorithms.

MRs	A*	L*	Jump points	Average
MR-1.1	0.48	0.44	0.51	0.48
MR-1.2	0.19	0.35	0.14	0.23
MR-2.1.1	0.13	0.03	0.18	0.11
MR-2.1.2	0.13	0.02	0.16	0.10
MR-2.2.1	0.08	0.04	0.18	0.10
MR-2.2.2	0.18	0.13	0.29	0.23
MR-3.1	0.18	0.23	0.29	0.23
MR-3.2	0.17	0.28	0.39	0.28

In the current research, evaluation of error detection ability of MRs is generally performed using performance index $MS(t, r)$ or $FDR(t, f, r)$. They could be calculate as follows:

$$MS\,(t, r) = \frac{M_{kr}}{M_t - M_e} \tag{12}$$

$$FDR\,(t, f, r) = \frac{T_{kr}}{T_r} \tag{13}$$

In Eq. (12), T represents the set of test cases, r represents the metamorphic relation to be evaluated, M_t represents the total number of variables in the program, M_e represents the number of equivalent variables and M_{kr} represents the number of times that the variable is detected, that is, the number of times that the test case fails the test. In Eq. (13), f represents the variable implanted with a specific defect, T_{fr} represents the number of times variable f is appears in the set of test cases, T_r represents the total number of test cases. When $MS(t, r)$ is adopted, multiple faults need to be implanted into the program. After the MT, the evaluation values of the MR are obtained by counting the number of test cases which failed. When $FDR(t, f, r)$ is adopted, detection ability of the MR for a specific defect is evaluated. When a single defect is implanted into the program, $M_e = 0$ and $M_t = 1$. In this case, $MS(t, r)$ is the number of failed test cases, and $FDR(t, f, r)$ uses the ratio of failed tests out of all test cases to reflect the detection ability of the MR.

Let T_n represent the number of invalid test cases, then $FDR'(t, f, r)$ as in Eq. (14) can be calculated as a revised version of $FDR(t, f, r)$.

$$FDR'(t, f, r) = \frac{T_{fr} - T_n}{T_r - T_n} \tag{14}$$

Considering that multiple faults can be detected by MRs, weighted average E_r as in Eq. (15) can be used as an index in the evaluation of MRs.

$$E_r = \frac{\sum_{i=1}^{n} FDR_i}{n} \tag{15}$$

Based on Eq. (15) and experimental data, evaluation values of MRs are calculated and recorded in Table 3.

5.3. Discussion

Table 3 shows evaluation values of each metamorphic relation under different path planning algorithms. It can be seen that MR-1.1 has the highest evaluation value and is therefore the most effective in fault detection. Also, it is shown that MR-2.1 and

MR-2.2 are both weak in fault detection. It is shown from the comparison between MR-2.1 and MR-2.2, that they can detect defects simultaneously, however, the test case pass rates for MR-2.1.1 and MR-2.1.2 were equal to or lower than MR-2.2.1 and MR-2.2.2. MR-3.1 and MR-3.2 are given relatively high grades, comparing to 2.1 and 2.2, due to advantage in revealing specific types of faults. Moreover, it is also observed that composed metamorphic relations such as MR-1.1, MR-3.1, MR-3.2 are more effective in fault detection than individual MRs, such as MR-2.1 and MR-2.2.

6. Conclusions

In this work, we propose a technique to validate and verify the graph search-based path planning algorithms for UAVs based on metamorphic testing. The main contributions of the paper can be summarized as follows: First, an initial test scenario generation algorithm and metamorphic relations are designed to realize the MT process of path planning software. Then test of actual UAV path planning software is performed, and the test data are analyzed in detail. Some faults in the software are detected, which proves the effectiveness of certain metamorphic relations in terms of fault detection. Second, mutation analysis to implanted faults in the program is carried out and tests are conducted afterwards in order to investigate the effectiveness of metamorphic relations. It is found that all faults can be detected with metamorphic relations adopted. Third, evaluation method of the metamorphic relations are studied and improved with new evaluation index proposed. Evaluation and analysis of MRs are carried out based on data collected from experiments, which provide strong support for the validation of MRs at data level. However, it is inefficient to use random MT in practical systems.

Although the MT technology of UAV path planning software is studied in detail and some conclusions are arrived at based on theoretical analysis and experimental results in this paper, this issue can still be further explored. Specifically, development and improvement of relevant testing and analyzing methods are still to be devoted to. First, framework of MT on path planning software needs to be further improved. Although an effective framework has been designed in this paper, integration of more expectations on the output of the algorithm need to be considered such that more effective metamorphic relations can be extracted. Second, it should be noticed that the same set of test cases was used in the experiment in this paper, which caused strong limitation to the verification of efficiency of MRs. Further study on the generation of MT cases needs to be performed so that a more effective set of test cases can be generated for each specific metamorphic relation.

CRediT authorship contribution statement

Lvyuan Wu: Methodology, Software, Formal analysis, Writing – original draft. **Zhiyu Xi:** Supervision, Reviewing and editing. **Zheng Zheng:** Conceptualization, Supervision. **Xiaoli Li:** Data curation, Software, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

References

Ahmed, S., Mohamed, A., Harras, K., Kholief, M., Mesbah, S., 2016. Energy efficient path planning techniques for UAV-based systems with space discretization. In: 2016 IEEE Wireless Communications and Networking Conference. IEEE, pp. 1–6.

Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., Werneck, R.F., 2016. Route planning in transportation networks. In: Algorithm Engineering. Springer, pp. 19–80.

Birk, A., et al., 2011. Safety, security, and rescue missions with an unmanned aerial vehicle (UAV). J. Intell. Robot. Syst. 64 (1), 57–76.

Bueno, P.M., Jino, M., Wong, W.E., 2014. Diversity oriented test data generation using metaheuristic search techniques. Inform. Sci. 259, 490–509.

Burnstein, I., 2006. Practical Software Testing: A Process-Oriented Approach. Springer Science & Business Media.

Chen, T.Y., Tse, T.H., Zhou, Z.Q., 2003. Fault-based testing without the need of oracles. Inf. Softw. Technol. 45, 1–9.

Cheng, Y., Li, D., Wong, W.E., Zhao, M., Mo, D., 2022. Multi-UAV collaborative path planning using hierarchical reinforcement learning and simulated annealing. Int. J. Perform. Eng. 18.

Delling, D., Sanders, P., Schultes, D., Wagner, D., 2009. Engineering route planning algorithms. In: Algorithmics of Large and Complex Networks. Springer, pp. 117–139.

Gijbels, I., Hall, P., Kneip, A., 1999. On the estimation of jump points in smooth curves. Ann. Inst. Statist. Math. 51, 231–251.

Gordon, M.S., Fedorov, D.G., Pruitt, S.R., Slipchenko, L.V., 2012. Fragmentation methods: A route to accurate calculations on large systems. Chem. Rev. 112 (1), 632–672.

Harabor, D., Grastien, A., 2014. Improving jump point search. In: Proceedings of the International Conference on Automated Planning and Scheduling. Vol. 24. pp. 128–135.

Hui, Z.-W., Huang, S., 2013. Achievements and challenges of metamorphic testing. In: 2013 Fourth World Congress on Software Engineering. IEEE, pp. 73–77.

Kanewala, U., Bieman, J.M., 2013. Using machine learning techniques to detect metamorphic relations for programs without test oracles. In: 2013 IEEE 24th International Symposium on Software Reliability Engineering. ISSRE, IEEE, pp. 1–10.

Liu, G., Shu, C., Liang, Z., Peng, B., Cheng, L., 2021. A modified sparrow search algorithm with application in 3D route planning for UAV. Sensors 21 (4), 1224. http://dx.doi.org/10.3390/s21041224.

Liu, W., Zheng, Z., Cai, K., 2013. Adaptive path planning for unmanned aerial vehicles based on bi-level programming and variable planning time interval. Chin. J. Aeronaut. 26 (3), 646–660.

Liu, X.-f., et al., 2014. An optimization model of UAV route planning for road segment surveillance. J. Cent. South Univ. 21 (6), 2501–2510.

Lv, S., Yin, B., 2020. Testing DNN-based path planning algorithms by metamorphic testing. In: 2020 7th International Conference on Dependable Systems and their Applications. DSA, IEEE, pp. 515–526.

McMinn, P., 2011. Search-based software testing: Past, present and future. In: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops. IEEE, pp. 153–163.

Myers, G.J., Sandler, C., Badgett, T., 2011. The Art of Software Testing. John Wiley & Sons.

Niewola, A., Podsedkowski, L., 2018. L* algorithm—a linear computational complexity graph searching algorithm for path planning. J. Intell. Robot. Syst. 91 (3), 425–444.

Nikolova, E., Brand, M., Karger, D.R., 2006. Optimal route planning under uncertainty. In: Icaps. Vol. 6. pp. 131–141.

Nuortio, T., Kytöjoki, J., Niska, H., Bräysy, O., 2006. Improved route planning and scheduling of waste collection and transport. Expert Syst. Appl. 30 (2), 223–232.

Özalp, N., Sahingoz, O.K., 2013. Optimal UAV path planning in a 3D threat environment by using parallel evolutionary algorithms. In: 2013 International Conference on Unmanned Aircraft Systems. ICUAS, IEEE, pp. 308–317.

Petritoli, E., Leccese, F., Ciani, L., 2018. Reliability and maintenance analysis of unmanned aerial vehicles. Sensors 18 (9), 3171.

Pitre, R.R., Li, X.R., Delbalzo, R., 2012. UAV route planning for joint search and track missions—An information-value approach. IEEE Trans. Aerosp. Electron. Syst. 48 (3), 2551–2565.

Qadir, Z., Ullah, F., Munawar, H.S., Al-Turjman, F., 2021. Addressing disasters in smart cities through UAVs path planning and 5G communications: A systematic review. Comput. Commun. 168, 114–135. http://dx.doi.org/10.1016/j.comcom.2021.01.003.

Roberge, V., Tarbouchi, M., Labonté, G., 2012. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. IEEE Trans. Ind. Inform. 9 (1), 132–141.

Rovira-Sugranes, A., Razi, A., Afghah, F., Chakareski, J., 2022. A review of AI-enabled routing protocols for UAV networks: Trends, challenges, and future outlook. Ad Hoc Netw. 130, 102790. http://dx.doi.org/10.1016/j.adhoc.2022.102790.

Segura, S., Fraser, G., Sanchez, A.B., Ruiz-Cortés, A., 2016. A survey on metamorphic testing. IEEE Trans. Softw. Eng. 42 (9), 805–824.

Segura, S., Towey, D., Zhou, Z.Q., Chen, T.Y., 2018. Metamorphic testing: Testing the untestable. IEEE Softw. 37 (3), 46–53.

Soltani, A.R., Tawfik, H., Goulermas, J.Y., Fernando, T., 2002. Path planning in construction sites: Performance evaluation of the Dijkstra, A*, and GA search algorithms. Adv. Eng. Inform. 16 (4), 291–303.

Stentz, A., 1997. Optimal and efficient path planning for partially known environments. In: Intelligent Unmanned Ground Vehicles. Springer, pp. 203–220.

Wei, Y., Blake, M.B., Madey, G.R., 2013. An operation-time simulation framework for UAV swarm configuration and mission planning. Procedia Comput. Sci. 18, 1949–1958.

Xie, X., Ho, J., Murphy, C., Kaiser, G., Xu, B., Chen, T.Y., 2009. Application of metamorphic testing to supervised classifiers. In: 2009 Ninth International Conference on Quality Software. IEEE, pp. 135–144.

Xie, X., Wong, W.E., Chen, T.Y., Xu, B., 2011. Spectrum-based fault localization: Testing oracles are no longer mandatory. In: 2011 11th International Conference on Quality Software. IEEE, pp. 1–10.

Yang, Y., Juntao, L., Lingling, P., 2020. Multi-robot path planning based on a deep reinforcement learning DQN algorithm. CAAI Trans. Intell. Technol. 5 (3), 177–183.

Yin, L., Li, X., 2006. Research on evaluation of path planning with span of planet detector. In: 2006 IEEE Conference on Robotics, Automation and Mechatronics. IEEE, pp. 1–6.

Zhang, J., Yin, B.-B., 2018. A design of charge-discharge circuit based on lithium battery for small UAVs. In: 2018 IEEE CSAA Guidance, Navigation and Control Conference. CGNCC, IEEE, pp. 1–6.

Zhang, M., Zhang, Y., Zhang, L., Liu, C., Khurshid, S., 2018. DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In: 2018 33rd IEEE/ACM International Conference on Automated Software Engineering. ASE, IEEE, pp. 132–142.

Zhang, D.Q., Zhao, J.F., Wang, M.H., Niu, G.H., 2010. Grey evaluation and optimization of UAV's path planning method. In: 2010 2nd International Conference on Electronic Computer Technology. IEEE, pp. 85–88.

Zhang, J., Zheng, Z., Yin, B., Qiu, K., Liu, Y., 2019. Testing graph searching based path planning algorithms by metamorphic testing. In: Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing. Vol. 12. PRDC, pp. 158–167. http://dx.doi.org/10.1109/PRDC47002.2019.00046.

Zhao, Y., Zheng, Z., Liu, Y., 2018. Survey on computational-intelligence-based UAV path planning. Knowl.-Based Syst. 158, 54–64. http://dx.doi.org/10.1016/j.knosys.2018.05.033.

Lvyuan Wu received a B.S. degree in Automation Science from Beihang University, in 2008. He is currently pursuing a Ph.D. in Automation Science and Electrical Engineering at Beihang University. His research interests include artificial intelligence software testing and UAV formation algorithms and software.

Zhiyu Xi (M'87) received Ph.D. degree in Electrical Engineering from University of New South Wales (UNSW), Australia, in 2010. From 2010 to 2012, she was a Postdoc Fellow at University of New South Wales where she was then an Associate Lecturer. From March 2015 until now, she joined the Beihang University as an Associate Professor. Her research interests include sliding mode control, networked control systems, and fuzzy control systems.

Zheng Zheng received his Ph.D. degree in computer software and theory from the Chinese Academy of Sciences, Beijing, China, in 2006. He is currently a Full Professor in Control Science and Engineering with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. In 2014, he was with the Department of Electrical and Computer Engineering, Duke University, working as a Research Scholar. His research interests include softare dependability, unmanned aerial vehicle path planning, articial intelligence applicatons, and softare fault localization.

Xiwoli Li received a B.S. degree in Computer science and theory from Beihang University, in 2009. Her research focuses on software testng.