TUMB at the SBFT 2024 Tool Competition – CPS-UAV Test Case Generation Track

Shuncheng Tang
University of Science and Technology of China
Hefei, China
scttt@mail.ustc.edu.cn

Zhenya Zhang
Kyushu University
Fukuoka, Japan
zhang@ait.kyushu-u.ac.jp

Ahmet Cetinkaya
Shibaura Institute of Technology
Tokyo, Japan
ahmet@shibaura-it.ac.jp

Paolo Arcaini
National Institute of Informatics
Tokyo, Japan
arcaini@nii.ac.jp

ABSTRACT

TUMB is a generator of scenarios for UAV testing, that participated in the UAV Testing Competition at SBFT 2024. TUMB relies on Monte Carlo Tree Search (MCTS) to search for different placements of obstacles of different sizes in the mission environment. The *exploration* phase of MCTS consists in increasing the number of obstacles in the environment, while the *exploitation* phase consists in the optimisation of the placement and of the dimensions of the last added obstacle.

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**; **Search-based software engineering**.

KEYWORDS

unmanned aerial vehicle, test generation, simulation-based testing, Monte Carlo Tree Search

ACM Reference Format:
Shuncheng Tang, Zhenya Zhang, Ahmet Cetinkaya, and Paolo Arcaini. 2024. TUMB at the SBFT 2024 Tool Competition – CPS-UAV Test Case Generation Track. In *Proceedings of 46th International Conference on Software Engineering (ICSE 2024).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs), embedded with onboard cameras and sensors, allow to perform autonomous flights, with a large range of application scenarios (e.g., food delivery). To test UAVs, simulation-based testing is an efficient technique, that allows generating virtual scenarios to evaluate the UAV under test in different conditions, without the cost of real-world testing. Aerialist (unmanned AERIAL vehIcle teST bench) is a framework that facilitates the whole process of simulation-based testing for UAVs [2, 3].

ICSE 2024, April 2024, Lisbon, Portugal

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The UAV Testing Competition at SBFT 2024 [4] called for test generators able to generate scenarios that are challenging for the UAV under test (i.e., PX4), in particular the component responsible for obstacle detection and avoidance (i.e., PX4-Avoidance). A *scenario* is given by the placement of some *obstacles* (up to four) in the environment in which the UAV is expected to complete its mission; each obstacle is characterized by a position in the environment, and by its length, width, and height. Obstacles cannot overlap.

TUMB (**T**esting of **U**AVs by **M**CTS and Path **B**locking) is a tool that participated to the competition. TUMB relies on Monte Carlo Tree Search (MCTS) [1] to explore different placements of obstacles in the environment. The *exploration* phase of MCTS consists in adding a new obstacle to the environment, while the *exploitation* phase consists in the optimisation of the placement and of the dimensions of the last added obstacle. TUMB is available at

https://github.com/MayDGT/UAV-Testing-Competition

2 TUMB

At a high level, TUMB generates scenarios by iteratively simulating the UAV under test and placing obstacles on its trajectories with the goal of making its mission challenging and, possibly, making it collide with an obstacle.

Specifically, TUMB relies on Monte-Carlo Tree Search (MCTS) to balance between *exploration*, which consists in trying to add an increasing number of obstacles in the environment; and *exploitation* which consists in optimising the position and dimension of the last placed obstacle. TUMB is presented in details in Alg. 1. The algorithm uses a *word* (i.e., a sequence of *letters*) to identify a placement of obstacles in the environment (i.e., a *scenario*), where each letter denotes an obstacle in a given position and with a given dimension; in the following, we will use the terms *word* and *scenario* interchangeably. During the search, these words are organized by using a *tree structure*, where each word corresponds to a path (thus a node) of the tree. For each node w of the tree, the algorithm stores three attributes related to the corresponding scenario: (i) the trajectory $traj(w)$ followed by the UAV during the simulation over the scenario; (ii) the *reward* $R(w)$ of the node, that tells how good the scenario is from a testing perspective, i.e., how close was the UAV from colliding with an obstacle during the simulation; (iii) the *(sub-)tree* $T(w)$ that has the node w as its root.

Initially, the algorithm considers the scenario with no obstacles, which corresponds to the empty word ϵ (i.e., the root of the

Algorithm 1 TUMB – MCTS-based test generation for UAVs

Require: hyperparameters $C = [C_1, C_2, C_3, C_4]$ and α for progressive widening
Require: hyperparameter c for UCB1
1: $TS \leftarrow \emptyset$
2: $traj(\epsilon), dist_{\min}(\epsilon) \leftarrow$ SIMULATE(ϵ)
3: $R(\epsilon) \leftarrow 0$
4: $T(\epsilon) \leftarrow \{\epsilon\}$
5: **while** budget is available **do**
6: MCTS(ϵ)

7: **function** MCTS(w)
8: $layer \leftarrow |w| + 1$
9: $children \leftarrow \{x \mid w \cdot x \in T(w)\}$
10: **if** $|children| \leq C_{layer} \cdot (|T(w)|)^{\alpha}$ or $|w| \geq 3$ **then**
11: $a \leftarrow$ GENERATE($w, traj(w)$)
12: $traj(w \cdot a), dist_{\min}(w \cdot a) \leftarrow$ SIMULATE($w \cdot a$)
13: **if** soft or hard failure occurs **then**
14: $TS \leftarrow TS \cup \{w \cdot a\}$
15: $R(w \cdot a) \leftarrow -1.0 * dist_{\min}$
16: $T(w \cdot a) \leftarrow \{w \cdot a\}$
17: **else**
18: $a \leftarrow \underset{x \in children}{\arg\max} \left(R(w \cdot x) + c\sqrt{\frac{2 \ln |T(w)|}{|T(w \cdot x)|}} \right)$
19: MCTS($w \cdot a$)
20: $R(w) \leftarrow \max_{x \in children} (R(w \cdot x))$
21: $T(w) \leftarrow T(w) \cup \bigcup_{x \in children} T(w \cdot x)$

search tree). TUMB simulates the UAV with the empty scenario ϵ, and obtains the UAV's trajectory $traj(\epsilon)$ with the minimum distance $dist_{\min}(\epsilon)$ to any obstacle[1] (line 2). Then, it iteratively calls the MCTS function from the root node, as long as the budget is available (lines 5-6).

The function MCTS takes as argument a node of the tree w, i.e., a scenario. MCTS works as follows. It first decides whether a new child of w should be created (lines 8-10): (i) according to the *progressive widening* condition that decides to add a new child if the current number of children $|children|$ is less that the depth of the tree $|T(w)|$ parametrized with hyperparameters C_{layer} and α; or (ii) if there are already three obstacles in w (the additional obstacle added as child reaches the maximum number of four obstacles in the scenario). If at least one of the two conditions is satisfied:

- it generates a new obstacle a with a given dimension, by placing it on the trajectory $traj(w)$ obtained when simulating w (line 11); the intuition is that we try to block the path of the UAV with the aim of letting it collide with the obstacle. In order to add the new obstacle, we need to avoid overlapping it with the existing obstacles; to do this in an efficient manner, we use multiple disks that over-approximate the coverage area of the existing rectangular obstacles, and place the new obstacle outside these disks;
- a is added as new child of w, and the UAV is simulated over the scenario $w \cdot a$, obtaining the trajectory $traj(w \cdot a)$ and the minimum distance to any obstacle $dist_{\min}(w \cdot a)$ (line 12).

[1]In this case, as there are no obstacles in ϵ, $dist_{\min}(\epsilon)$ is ∞.

- if the simulation leads to a *soft* or *hard failure* [4], the scenario $w \cdot a$ is collected in the test suite TS (lines 13-14);
- it updates the reward R of scenario $w \cdot a$, which indicates how good the scenario is from a testing perspective (line 15); it is defined as the negative minimum distance from any obstacle, where reward 0 indicates a collision;
- it updates the search tree with the new sub-tree $T(w \cdot a)$ (line 16).

Instead, if neither of the two conditions is satisfied, the visit of the tree continues to lower layers as follows:

- it selects an existing child a of w by using the *UCB1* policy (line 18); the policy tries to balance (using hyperparameter c) between selecting the nodes that have the highest reward and those that have been visited less frequently;
- it recursively calls MCTS with the selected child node (line 19).

In both cases, the reward of w is updated as the maximum reward of the children (line 20), and the sub-tree with the root w is updated to include the newly created nodes (line 21).

2.1 Setting of MCTS hyperparameters

For the competition, TUMB has been initialised with the following values for the hyperparameters: $c = 1/\sqrt{2}$ for UCB1, and $C = [0.4, 0.5, 0.6, 0.7]$ and $\alpha = 0.5$ for progressive widening. As shown in Alg. 1, for each level of the tree (i.e., number of obstacles in the scenario), there is a different value of hyperparameter C, meaning that the level of exploitation is different for different types of scenarios. Since we want to generate failing scenarios having the minimum number of obstacles, we set the values of C in an increasing way; this guarantees that, as the number of obstacles in the scenario increases, the search focuses more in exploring scenarios with the same number of obstacles, rather than adding a new obstacle.

ACKNOWLEDGMENTS

P. Arcaini is supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST, Funding Reference number: 10.13039/501100009024 ERATO; and by Engineerable AI Techniques for Practical Applications of High-Quality Machine Learning-based Systems Project (Grant Number JPMJMI20B8), JST-Mirai. A. Cetinkaya is supported by JSPS KAKENHI (Grant Numbers JP20K14771 and JP23K03913). Z. Zhang is supported by JSPS KAKENHI Grant No. JP23K16865 and No. JP23H03372.

REFERENCES

[1] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (2012), 1–43.
[2] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2023. Simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flights. In *2023 16th IEEE International Conference on Software Testing, Verification and Validation (ICST)*.
[3] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2024. Simulation-based Testing of Unmanned Aerial Vehicles with Aerialist. In *International Conference on Software Engineering (ICSE)*.
[4] Sajad Khatiri, Prasun Saurabh, Timothy Zimmermann, Charith Munasinghe, Christian Birchler, and Sebastiano Panichella. 2024. SBFT Tool Competition 2024 - CPS-UAV Test Case Generation Track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024*.