Spirale at the SBFT 2023 Tool Competiton - Cyber-Physical Systems Track

Domenico De Vivo
Università degli Studi di Napoli Federico II
Napoli, Italy
dom.devivo@studenti.unina.it

Anna Rita Fasolino
Dip. di Ingegneria Elettrica e delle Tecnologie dell'Informazione
Università degli Studi di Napoli Federico II
Napoli, Italy
fasolino@unina.it

***Abstract*—In this paper, we present Spirale, a search-based testing tool designed to generate scenarios for testing Lane-Keeping Assist Systems (LKAS). Spirale took part in the CPS (Cyber-Physical Systems) testing competition held at SBFT 2023.**

***Index Terms*—search-based testing, autonomous driving systems, genetic algorithms**

I. Introduction

Search-based testing is one of the most widely-adopted methodologies for testing ADS (Autonomous Driving Systems) [1]. It consists in searching in the parameter space for specific parameter values that optimize a testing objective. Designing better search algorithms that aim to improve the search efficiency is an open challenge in the field of ADS systems. In this paper, we present the testing tool Spirale that uses a genetic algorithm to generate test scenarios consisting in virtual roads. The code of Spirale is available at: https://github.com/domenico-devivo/cps-tool-competition

II. The Spirale Technique

According to the CPS testing competition requirements, testing scenarios consist of roads that must not self-intersect, be too sharp, or exceed a given map, so in Spirale we made several design decisions to avoid generating invalid roads. Our technique is described by Algorithm 1. It starts building an initial population of roads (Line 3). Then it generates heir roads by applying the crossover (Line 4) and mutation (Line 5) operators and evaluates their fitness value (Line 6). After this step, the algorithm evolves in a while loop (Lines 7-18) in which new roads are searched for applying the *selection*, *crossover* and *mutation* steps of the genetic approach and their fitness values are assessed. Each time the fitness of a new population is worse than the parent one (Line 18) or the population has not enough roads to select, i.e., less or equal than 4 in our implementation (Line 8), the while condition is false and the algorithm restarts by generating a novel initial population (Line 3), actually implementing a multi-start approach.

Algorithm 1: Spirale

```
   Input: map_size, time_budget
1  Roadtition start()
2      while time_budget_is_not_over do
3          init_roads_pop ← init_pop_gen(3)
4          heir_pop ← crossover(init_roads_pop)
5          heir_pop ← mutation(heir_pop)
6          fitness ← fitness_f(heir_pop)
7          do
8              if len(heir_pop > 4) then
9                  new_pop ← selection_f(heir_pop, 5)
10                 heir_pop ← crossover(new_pop)
11                 heir_pop ← mutation(heir_pop)
12                 heir_fitness ← fitness_f(heir_pop)
13                 fitness = fitness − heir_fitness
14             end
15             else
16                 v = False
17             end
18         while (fitness > 0 and v == True);
19     end
20 end
```

A. Initial Population

As most genetic algorithms, Spirale starts by generating an initial population of 3 roads. We encode the roads of the initial population as sequences of points sampled from arcs of a randomly generated Archimedean spiral. In particular, Spirale uses the following parametric equations to describe the spiral curve in the Cartesian plane, where a is a constant, r is the spiral radius, θ is an angle in radians, and c_{x}, c_{y} are the coordinates of the spiral polar:

$$\begin{cases} x = a \cdot r \cdot \cos\theta + c_{\text{x}} \\ y = a \cdot r \cdot \sin\theta + c_{\text{y}} \end{cases} \tag{1}$$

The points that will compose an initial road will be sampled among the points of an arc whose angles range in $[\theta_{\text{in}}...\theta_{\text{fin}}]$. In order to obtain valid roads, i.e., having non-intersecting points and length greater than the minimum allowed one, the values of θ_{in} and θ_{fin} are chosen from predefined disjointed ranges of values. Moreover, during the generation, we alternate clockwise and counterclockwise spiral arcs to obtain diverse sets of points. An example of initial population roads including points from a single spiral arc are reported in Fig.1.

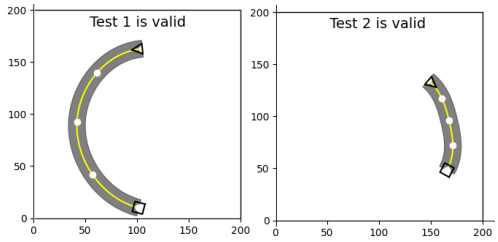

Fig. 1. Two roads defined from clockwise and counterclockwise spiral arcs

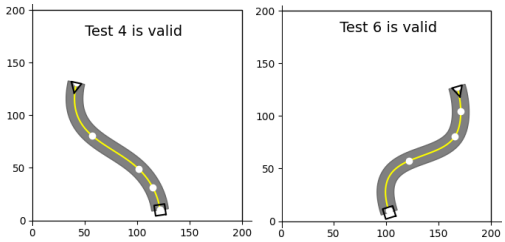

Fig. 2. Examples of Heir Roads

B. Fitness Function

Spirale uses as fitness function the minimum distance between the center of mass of the car and the road margins, achieved during the simulation on the individual, i.e., the road.

Spirale tries to improve the proposed solutions by minimizing this fitness function. The idea is that the lesser this value, the closer we get to failures. The fitness function is used both for assessing the initial population after the crossover (Line 6) and for evaluating the difference between the fitness of parent and heir populations (Lines 12-13).

C. Crossover

Our crossover operator generates a new road by combining two valid and distinct roads. Applying this operator to the initial population (Line 4), a heir initial population is created. In the do-While loop (Lines 7-18), crossover is applied again for generating the heir population of the roads selected at line 9. Given two roads, the crossover operator joins them by shifting the points of the latter road towards the points of the former. Starting from a population of n roads, this function will generate a set of at most $n * (n - 1)$ new valid roads.

D. Selection

The selection operator chooses the most suitable individuals to generate, through crossover, a new population, i.e., the heir population. This operator selects only the 5 roads of a given population with the smaller fitness values (Line 9).

E. Mutation

Our mutation operator randomly shifts the points of a road, thus increasing the entropy of the road set by seeking new pathways. In Spirale, we exploited the mutation mechanism to improve the validity of the proposed solutions, by using it after the crossover operator (Line 5 and 11). In particular, we considered two mutation operators, named *Halve* and *Reframe*. They are used when a generated road does not fit the map and operate by either removing points from both the road extremes, or shifting the road points, respectively. Fig. 2 shows two roads of the heir population obtained after the application of the crossover and mutation operators to the parent roads illustrated in Fig. 1.

III. Results

In this competition, Spirale tested two different LKAs: BeamNG.AI, the driving agent offered by the BeamNG.tech simulator and DAVE-2, a DL-based driving agent trained by the competition organizers. Spirale was run 6 times on each agent and its effectiveness was assessed in terms of feature map coverage [5].

The results of the competition are reported in [2]. Spirale achieved the best results when testing BeamNG.AI, showing that our tool was able to generate 428 roads on average, with an average percentage of about 80% valid roads. In future work, we intend to investigate solutions for improving the diversity of the generated tests and the fault-exposing capability of Spirale.

IV. Conclusion

Defining motivating approaches for supporting testing learning is a relevant educational challenge [3]. The CPS competition at SBFT 2023 provides an example of learning practice based on gamification having the potential to convey software testing matters in effective and stimulating way. Spirale has been developed in the context of a Computer Engineering Master Degree Final Thesis development task, within the activities of the ENACTEST Project [4]. The experience with Spirale showed us the feasibility and usefulness of alternative learning practices, like competition and gamification, also in the context of search-based testing learning.

V. Acknowledgements

This work received funding from the ERASMUS-EDU-2021-PI-ALL-INNO under grant agreement n. 101055874-ENACTEST (European iNnovation AllianCe for TESting educaTion).

References

[1] S. Tang, Z. Zhang, Y. Zhang, J. Zhou, Y. Guo, S. Liu, S. Guo, Y. Li, L. Ma, Y. Xue, and Y. Liu. 2023. A Survey on Automated Driving System Testing: Landscapes and Trends. ACM Trans. Softw. Eng. Methodol. Just Accepted (February 2023). https://doi.org/10.1145/3579642

[2] M.Biagiola and S. Klikovits and J. Peltomaki and V. Riccio. SBFT Tool Competition 2023 - Cyber-Physical Systems Track. 16th IEEE/ACM. International Workshop on Search-Based and Fuzz Testing, SBFT 2023, Melbourne, Australia, May 14, 2023.

[3] V. Garousi, A. Rainer, Per Lauvås, A. Arcuri, Software-testing education: A systematic literature mapping, Journal of Systems and Software, Volume 165, 2020,110570.

[4] B Marín, T. E. J. Vos, A. C. R. Paiva, A. R. Fasolino, M. Snoeck: ENACTEST - European Innovation Alliance for Testing Education. RCIS Workshops 2022, https://ceur-ws.org/Vol-3144/RP-paper5.pdf

[5] T. Zohdinasab, V. Riccio, A. Gambi, P. Tonella: Efficient and effective feature space exploration for testing deep learning systems. ACM Transactions on Software Engineering and Methodology, 2022