FreneticV at the SBST 2022 Tool Competition

Ezequiel Castellano
National Institute of Informatics
Tokyo, Japan
ecastellano@nii.ac.jp

Stefan Klikovits
National Institute of Informatics
Tokyo, Japan
klikovits@nii.ac.jp

Ahmet Cetinkaya
National Institute of Informatics
Tokyo, Japan
cetinkaya@nii.ac.jp

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

ABSTRACT

`FreneticV` is a search-based testing tool based on an evolutionary approach that generates roads where an automated driving agent possibly fails the lane-keeping task. It uses a curvature-based road representation and, compared to its predecessor `Frenetic`, considers the validity of the generated roads. In particular, it tries to avoid generating roads with overly sharp turns, it detects self-intersecting roads, and has the capability of rotating and relocating roads so as to fit them in a given map.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; • **Software and its engineering → Search-based software engineering**.

KEYWORDS

search-based testing, autonomous driving, Frenet frame, FreneticV

ACM Reference Format:

Ezequiel Castellano, Stefan Klikovits, Ahmet Cetinkaya, and Paolo Arcaini. 2022. FreneticV at the SBST 2022 Tool Competition. In *Proceedings of The 44th International Conference on Software Engineering (ICSE 2022).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

`Frenetic` [2] is a search-based approach to generate roads for simulation-based testing of autonomous driving systems. Namely, `Frenetic` aims at generating roads in which the ego vehicle drives off the road, so possibly exhibiting failures of the lane-keeping component. `Frenetic` participated in the first competition on CPS testing at SBST'21 [4]. It has been later also used in an empirical study [1] to assess the influence of the road representation in search-based testing of autonomous driving systems.

In the CPS competition at SBST'21 [4], `Frenetic` obtained very good results in terms of diversity of the generated roads, but it also produced a relatively large number of invalid tests (i.e., too sharp, or self-intersecting roads). Therefore, for the second edition of the competition at SBST'22 [3], we extended `Frenetic` with `FreneticV` (i.e., `Frenetic` + `V`alidation), that employs particular strategies to avoid generating invalid roads. `FreneticV` code is available at

https://github.com/ERATOMMSD/freneticV-sbst22

In the following, in Sect. 2, we introduce the generation algorithm employed by `FreneticV`, focusing in particular on the new features introduced to achieve valid roads. Then, in Sect. 3, we provide a critical discussion about the performance of `FreneticV` at the competition, and outline possible future improvements.

2 FRENETICV

`FreneticV` is built upon `Frenetic`. We refer to [2] for a complete description of `Frenetic`; we here provide a short description, and focus on the additional features introduced by `FreneticV`.

Given a list of *curvature values* and *segment lengths*, `FreneticV` builds a road as explained in [2]. Alg. 1 shows how it searches for curvature values describing roads that possibly lead to failures.

`FreneticV` aims at producing roads that are short, but yet useful from a testing perspective. So, the segment length between each road point has been fixed to 5 meters; the number of points for a randomly generated road has been calculated based on the size of the map and the segment length (lines 1-2).

It first randomly generates roads of the initial population for a given period of time *rndBdgt* (lines 3-4); the length of a randomly generated road is defined as the number of points *numPoints* ±5.

Then, for the remaining generation time (line 5), the algorithm keeps on searching for new roads as follows. It first selects the best candidate for mutation based on the *minimum out-of-bounds distance* (MOOBD), i.e., the distance between the center of mass of the car and the center lane. A candidate is suitable for mutation if its MOOBD is lower than a threshold *thMOOBD* (line 6), which was fixed at -0.5 for the competition. If no suitable candidate is available, new roads are generated randomly (line 19).

If a candidate *parent* exists (line 7), different mutations are applied to it (lines 13-17), depending on whether it is a failing (line 10) or passing test (line 12). Refer to [2] for details on the operators.[1]

After generating a number of mutants (line 20), the crossover is applied between the tests having lowest MOOBD (lines 20-24). Two types of crossover are applied; refer to [2] for their description.

[1]Note that we did not apply operator mut_{f1} from [2] as not effective; moreover, differently from [2], in mut_{p6}, the curvature values are modified of 1-5%.

ICSE 2022, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Algorithm 1 `FreneticV`

Require: *mapSize, totalTime, rndBdgt, crossFreq, crossNum, thMOOBD*

$segmentLength \leftarrow 5$
$numPoints \leftarrow \max(20, \min(mapSize/segmentLength, 50))$
while $elapsedTime < rndBdgt$ **do**
  genRndRoad($numPoints + rnd(-5, 5)$, $segmentLength$)
while $elapsedTime < totalTime$ **do**
  if $|$candidatesDeviatingFromMOOBDth($thMOOBD$)$| > 0$ **then**
    $parent \leftarrow$ bestNotVisitedCandidate($thMOOBD$)
    $parent.visited \leftarrow true$
    if $parent.failed$ **then**
      $mutations \leftarrow [mut_{f2}, mut_{f3}]$
    else
      $mutations \leftarrow [mut_{p1}, mut_{p2}, mut_{p3}, mut_{p4}, mut_{p5}, mut_{p6}]$
    for mutation $\in mutations$ **do**
      $child \leftarrow$ mutation($parent$)
      $child.visited \leftarrow parent.failed$
      if $child.failed$ **then**
        break
  else
    genRndRoad($numPoints + rnd(-5, 5)$, $segmentLength$)
  if $recent\_count > crossFreq$ **then**
    while $|children| < crossNum$ **do**
      $parent_1, parent_2 \leftarrow$ bestCandidates($thMOOBD$)
      crossover $\leftarrow$ rndChoice($[cross_1, cross\_2]$)
      $children$.append(crossover($parent_1, parent_2$))

2.1 Road validation

`FreneticV` uses several mechanisms to generate realistic roads that fit into given maps with fixed sizes, while avoiding excessively sharp turns and self-intersections.

2.1.1 Avoiding overly sharp turns and self-intersections. For a planar curve, the magnitude of the curvature and the radius of the curve are reciprocals. Both `Frenetic` and `FreneticV` use curvature-based road representations. This allowed us to exploit the reciprocal relationship to identify a global upper bound for the curvature values so that the radius of the generated road does not go below the threshold, and thus the turns are not overly sharp.

In `FreneticV`, we also developed a method to check if a generated road is self-intersecting. In this method, we accounted for the road width, because a road with a positive width may be self-intersecting even if the curve corresponding to its center-line is not. In particular, we considered left and right edges of the road and checked if those edges are intersecting with themselves or with each other. We discard the roads that are detected to be self-intersecting.

2.1.2 Road rotation and relocation to ensure validity. Mutation and crossover operators of `FreneticV` involve randomness in assignment of curvature values. As a result, after curvature values are transformed into Cartesian coordinates, a road generated by `FreneticV` can cross the boundaries of a given map, even if the initial point of the road is inside the map. In `FreneticV`, we rotate and relocate road points so as to fit the entire road in the map.

Specifically, we iterate over a set of orientation values, and for each orientation ϑ, we rotate the road points $\gamma_0, \ldots, \gamma_{N-1}$ around the center of the road's bounding box by ϑ degrees. Furthermore, in each iteration, we also calculate the convex hull of the rotated road points and relocate the entire rotated road so that the minimum x and y coordinates of the exterior points of the convex hull both equal m, a small margin value that we use to avoid road edges being too close to map boundaries. If the rotated-and-relocated road points γ_i' are all contained in the map with margin m, then we stop iteration, and take $\gamma_0', \ldots, \gamma_{N-1}'$ as a valid road that can be considered as a test scenario (see Fig. 1).

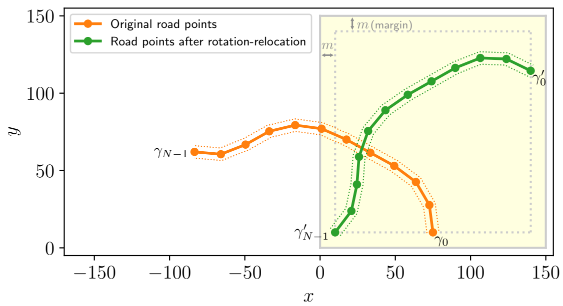

Figure 1: Original road points and road points after rotation and relocation operations to ensure validity on a square map.

3 RESULTS AND DISCUSSION

The *effectiveness* score of `FreneticV` provided in SBST'22 competition report [3] indicates the usefulness of the newly developed validity-checking mechanisms. However, as there is a trade-off between checking road validity and generating more roads, `FreneticV` obtained a relatively lower *efficiency* score. The *diversity* score of `FreneticV` is among the top. To further increase diversity, we believe that directional coverage of the roads can be improved. Specifically, the method discussed in Sect. 2.1.2 can be modified to generate new valid roads with extra rotations (e.g., 90, 180, and 270 degrees for square maps). This would increase directional coverage and it would be particularly useful for testing driving agents whose correctness could be affected by the direction of driving (e.g., DL-based agents such as Dave2).

ACKNOWLEDGMENTS

The authors are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST; Funding Reference number: 10.13039/501100009024 ERATO. S. Klikovits is also supported by Grant-in-Aid for Research Activity Start-up 20K23334, JSPS. A. Cetinkaya is also supported by Grant-in-Aid for Early-Career Scientists 20K14771, JSPS.

REFERENCES

[1] Ezequiel Castellano, Ahmet Cetinkaya, and Paolo Arcaini. 2021. Analysis of Road Representations in Search-Based Testing of Autonomous Driving Systems. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. 167–178. https://doi.org/10.1109/QRS54544.2021.00028

[2] Ezequiel Castellano, Ahmet Cetinkaya, Cédric Ho Thanh, Stefan Klikovits, Xiaoyi Zhang, and Paolo Arcaini. 2021. Frenetic at the SBST 2021 Tool Competition. In *2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST)*. 36–37. https://doi.org/10.1109/SBST52555.2021.00016

[3] Alessio Gambi, Gunel Jahangirova, Vincenzo Riccio, and Fiorella Zampetti. 2022. SBST Tool Competition 2022. In *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2022, Pittsburgh, PA, USA, May 9, 2022*.

[4] Sebastiano Panichella, Alessio Gambi, Fiorella Zampetti, and Vincenzo Riccio. 2021. SBST Tool Competition 2021. In *2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST)*. 20–27. https://doi.org/10.1109/SBST52555.2021.00011