Frenetic at the SBST 2021 Tool Competition

Ezequiel Castellano, Ahmet Cetinkaya, Cédric Ho Thanh, Stefan Klikovits, Xiaoyi Zhang, Paolo Arcaini
National Institute of Informatics
Tokyo, Japan
{ecastellano,cetinkaya,hothanh,klikovits,xiaoyi,arcaini}@nii.ac.jp

***Abstract*—Frenetic is a genetic approach that leverages a curvature-based road representation. Given an autonomous driving agent, the goal of Frenetic is to generate roads where the agent fails to stay within its lane. In other words, Frenetic tries to minimize the "out of bound distance", which is the distance between the car and either edge of the lane if the car is within the lane, and proceeds to negative values once the car drives off. This work resembles classic aspects of genetic algorithms such as mutations and crossover, but introduces some nuances aiming at improving diversity of the generated roads.**

I. Frenetic

Frenetic[1] is a genetic approach that uses curvature values as a means to represent roads. Section I-A explains how a road can be represented by its curvature values. Section I-B shows how Frenetic uses this representation to produce tests following a genetic-based approach and its mechanism to improve diversity.

A. Road representation

We use curvatures associated with smooth planar curves to represent roads. In our characterization of curvatures, we follow Section 1.2 of [1] and use Frenet frames. Frenet frames are commonly used in trajectory planning for self-driving systems (see [3] and the references therein).

Let $\gamma\colon [0,r] \to \mathbb{R}^2$ be a smooth planar curve of length $r > 0$. We use $e_1 = \gamma' : [0,r] \to \mathbb{R}^2$ to denote the velocity vector field along the curve. Let us further assume that γ is parameterized by arc-length, i.e., that $\|e_1(s)\| = 1$ for all $s \in [0,r]$. Let $e_2(s)$ be $e_1(s)$ rotated counterclockwise by $\pi/2$ radians. The pair $(e_1(s), e_2(s))$ forms the so-called *Frenet frame* [1] of γ at s. An illustration of this frame for a planar curve is given in Fig. 1.

There exists a unique smooth function $\kappa : [0,r] \longrightarrow \mathbb{R}$ such that $e_1'(s) = \kappa(s)e_2(s)$. The value $\kappa(s)$ is called the *curvature* of γ at s. By [1, theorem 1.2.2], κ completely determines γ up to rotation and translation.

We assume that the behavior of a vehicle in the driving scenario of the competition depends only on the shape of the curve γ and not its position or orientation. In this context, the curvature κ is well suited for representing roads.

Concretely, in our tool, we represent a road $\gamma\colon [0,r] \to \mathbb{R}^2$ by an array of N curvature values $\kappa_0, \ldots, \kappa_{N-1} \in \mathbb{R}$, which are uniformly sampled across the length of $[0,r]$. The values $\kappa_0, \ldots, \kappa_{N-1}$ cannot be directly passed to the BeamNG driving simulator, since it only accepts road points in Euclidean coordinates. This requires us to transform curvature values to road points. We set the first road point to $\gamma_0 = (0,0)$ and the initial orientation to $\theta_0 = \pi/2$. From there, the transformation is achieved by numerical integration. As an example, Fig. 2 shows road orientation values $\theta_1, \ldots, \theta_{N-1}$ and road points $\gamma_1, \ldots, \gamma_{N-1}$ in Cartesian coordinates for given curvature values $\kappa_0, \ldots, \kappa_{N-1}$.

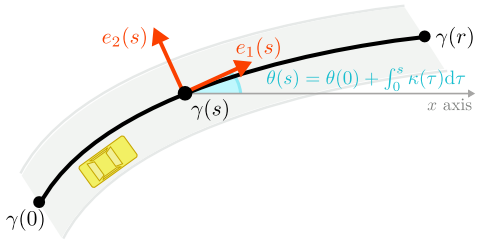

Fig. 1. Planar curve γ, Frenet frame $(e_1(s), e_2(s))$, and orientation θ as the integral of curvature κ given for standard Euclidean coordinates

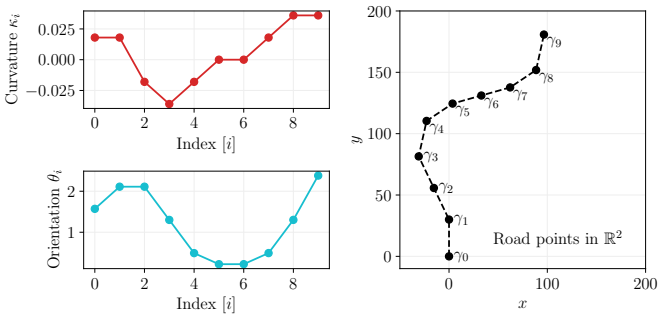

Fig. 2. Road orientations θ_i and positions γ_i for given curvature values κ_i. For example, κ_3 is negative, so the road curves to the right at γ_3. Since κ_5 and κ_6 are close to 0, the corresponding part of the road is mostly straight.

B. The Frenetic algorithm

Frenetic algorithm (see Listing 1) aims to produce short interesting road segments. Thus, the maximum number of initial points for random generation is 30, and the segment length is fixed at 10 meters (line 3-4). To avoid producing overly sharp roads, the curvature values are bounded both globally within the interval [-0.07, 0.07], and w.r.t. the previous curvature value κ_{i-1} by 0.05, i.e., κ_i is in the interval $[\kappa_{i-1} - 0.05, \kappa_{i-1} + 0.05]$.

This is a preprint of the paper submitted to the CPS Competition of SBST 2021. The authors are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST. S. Klikovits is also supported by Grant-in-Aid for Research Activity Start-up 20K23334, JSPS.

[1]Our code is available at github.com/ERATOMMSD/frenetic-sbst21

```
1  def frenetic(map_size, total, random_budget,
2               cross_freq, cross_num, thresh):
3    segment_length = 10
4    points = min(map_size // segment_length, 30)
5    while time < random_budget:
6      gen_rnd_road(points + rnd(-5,5), segment_length)
7    while time < total:
8      if len(candidates_with_min_oob_gt(thresh)) > 0:
9        parent = best_not_visited_candidate(thresh)
10       parent.visited = True:
11       if parent.failed:
12         mutations = [mut_f1, ... , mut_f4]
13       else:
14         mutations = [mut_p1, ... , mut_p6]
15       for mutation in mutations:
16         child = mutation(parent)
17         child.visited = parent.failed
18         if child.failed:
19           break
20     else:
21       gen_rnd_road(points + rnd(-5,5), segment_length)
22     if recent_count > cross_freq:
23       parents = best_candidates(cross_num, thresh)
24       while len(children) < cross_num:
25         parent1, parent2 = rnd.choice(parents, 2)
26         crossover = rnd.choice([Cross_1, Cross_2])
27         children.append(crossover(parent1, parent2))
```

Listing 1. Frenetic Algorithm

Given a list of curvature values and the defined segment length, Frenetic constructs the road following the method of Section I-A. When integrating the curvature values, the Cartesian representation may result in some points outside of the boundaries of the map. In those cases, Frenetic tries to re-frame the test by changing the initial point. However, there are cases in which this does not solve the problem, and in those cases the tests were discarded. This could be further improved by also trying rotational transformations. In addition, we conservatively use a 10-meter margin to account for the width of a road.

As most genetic approaches, Frenetic has three different phases: 1) generate an initial population [lines 5-6], 2) mutate previously generated tests [lines 7-21], 3) perform crossover among those tests [lines 22-27]. It first spends a given time budget for random generation (`random_budget`), which creates the initial population of roads. To introduce some variability, the length of a random road is defined as points ± 5. Frenetic then selects the best candidates for mutations based on the minimum out of bounds distance (MOOBD), which measures 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, which was fixed at -0.5 for the competition. When no suitable candidate is available, the algorithm continues randomly generating new roads. After generating a number of mutants, the crossover is applied between the tests with lowest MOOBD. Our hypothesis is that tests with lower MOOBD are more likely to produce a failure.

To reduce the number of similar tests, every time a parent has a child producing a failure, the algorithm stops producing children from it. Besides, different mutations are applied to tests that passed and tests that failed. For instance, reversible mutations are only applied to a test that already failed, and those mutants are marked as already visited. A mutation Mut is reversible if $\mathrm{Mut}(\mathrm{Mut}(T))$ is equal to the original test T.

Given a *passing* test, defined by its curvature values, the following mutations can be performed.

- $\mathrm{Mut}_{\mathrm{P}_1}$: appends 1 to 5 new curvature values;
- $\mathrm{Mut}_{\mathrm{P}_2}$: removes 1 to 5 values randomly;
- $\mathrm{Mut}_{\mathrm{P}_3}$: removes 1 to 5 values from the front;
- $\mathrm{Mut}_{\mathrm{P}_4}$: removes 1 to 5 values from the end;
- $\mathrm{Mut}_{\mathrm{P}_5}$: replaces 1 to 5 random values with new values;
- $\mathrm{Mut}_{\mathrm{P}_6}$: makes the turns sharper by increasing all curvature values by 10% to 20%.

Given a *failed* test, defined by its curvature values $\kappa_0, \ldots, \kappa_{N-1}$, the following mutations are applied.

- $\mathrm{Mut}_{\mathrm{F}_1}$: reverses the Cartesian representation of the road, which also changes the driving lane;
- $\mathrm{Mut}_{\mathrm{F}_2}$: reverses the order of curvature values;
- $\mathrm{Mut}_{\mathrm{F}_3}$: splits curvature values in the middle and swaps front and back, i.e., $\kappa_{\lfloor N/2 \rfloor}, \ldots, \kappa_{N-1}, \kappa_0, \ldots, \kappa_{\lfloor N/2 \rfloor - 1}$;
- $\mathrm{Mut}_{\mathrm{F}_4}$: flips the sign of all curvature values, i.e., $\kappa_0 * -1.0, \ldots, \kappa_{N-1} * -1.0$ to "mirror the road".

As for the crossover, two different operators are applied with equal probability. Given two tests $\kappa_0, \ldots, \kappa_{N-1}$ and $\kappa'_0, \ldots, \kappa'_{M-1}$, the operators are applied as follows.

- Cross_1: chromosome crossover produces 1 child of length $O = \mathit{min}(N, M)$ that has one of the curvature values of its parents in each position, i.e., $c_0, \ldots, c_{O-1}$ with $c_i = \kappa_i \vee \kappa'_i$;
- Cross_2: single point crossover produces 2 children of lengths $\lfloor N/2 \rfloor + \lceil M/2 \rceil$ and $\lceil N/2 \rceil + \lfloor M/2 \rfloor$ by splitting the values in the middle and swapping the front of each list, i.e., the children are obtained as $\kappa_0, \ldots, \kappa_{\lfloor N/2 \rfloor - 1}, \kappa'_{\lfloor M/2 \rfloor}, \ldots, \kappa'_{M-1}$ and $\kappa'_0, \ldots, \kappa'_{\lfloor M/2 \rfloor - 1}, \kappa_{\lfloor N/2 \rfloor}, \ldots, \kappa_{N-1}$.

II. Results and Final Remarks

In the SBST21 competition report [2], Frenetic was highlighted as one of the most effective tools for triggering failures and also outperformed others in terms of diversity. Nonetheless, it was mentioned that Frenetic produced a relatively large number of invalid tests. Given that the definition of a valid test is available, it would have been easy to check if a produced test is valid or not before sending it to the simulator. However, this would imply checking this condition twice, as the provided execution pipeline also checks validity. Given that these checks and the generation of new tests in Frenetic are inexpensive, we decided to not penalize invalid tests.

References

[1] S. Kobayashi. *Differential Geometry of Curves and Surfaces*. Springer, 2019.

[2] S. Panichella, A. Gambi, F. Zampetti, and V. Riccio. Sbst tool competition 2021. In *International Conference on Software Engineering, Workshops, Madrid, Spain, 2021*. ACM, 2021.

[3] S. Zhu and B. Aksun-Guvenc. Trajectory planning of autonomous vehicles based on parameterized control optimization in dynamic on-road environments. *J. Intel. Robot. Syst.*, 100(3):1055–1067, 2020.