EVOLUTIONARY APPROACH TO FINDING AN OPTIMAL RACING LINE IN A VEHICLE SIMULATOR

PATRIK MIKČO

2023/2024

INTRODUCTION

WHAT IS THE MAIN GOAL?

- TORCS simulator
- Finding an optimal racing line
- Comparing results between known starting parameters, a randomized set and built-in TORCS simulator drivers
- Getting rid of the guessing game

PROBLEM DESCRIPTION

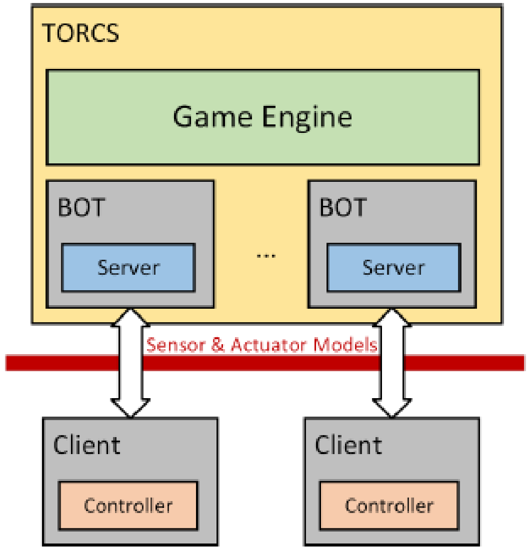
- Simulators fail to provide the best driving line
- Simulators require manual testing of each track, car etc.
- GA will determine the best track line to follow while dealing with speed and steering
- 2 fuzzy sub-controllers to host control logic for outputting the best speed and steering at every moment
- Wall damage is included, player collisions are excluded
- Goal is to drive **fast** and **safely** (fastest lap with minimal damage)
- Trapezoidal membership function defines the boundaries for fuzzy rules

TESTING ENVIRONMENT

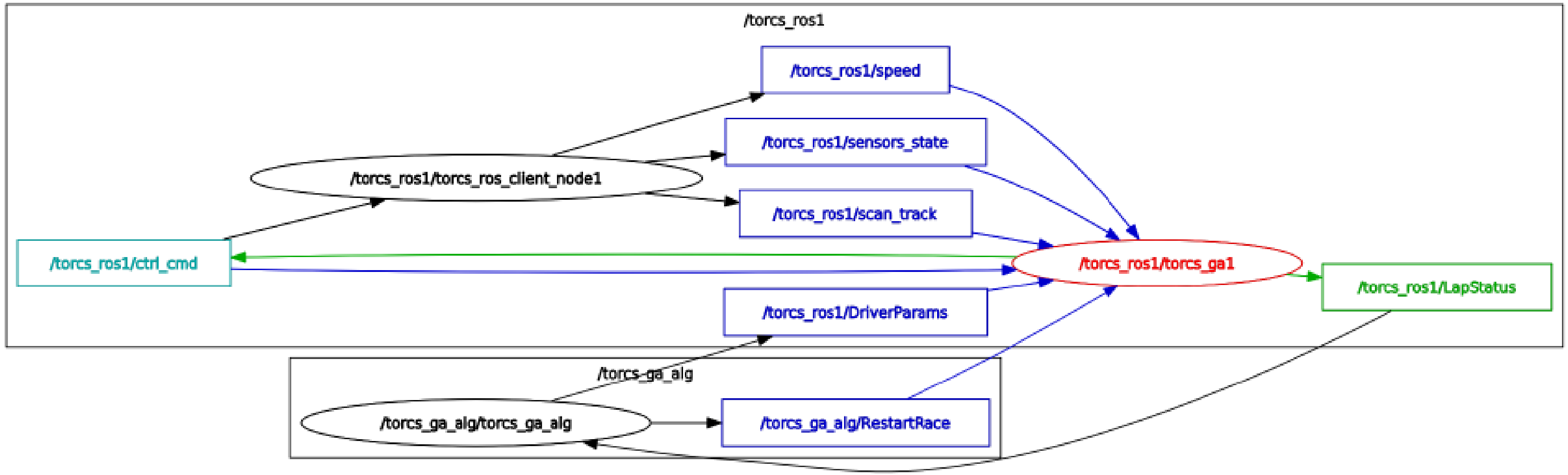
- TORCS simulator
- Robot Operating System (ROS)
- TORCS client represents **node** in ROS
- Nodes communicate via **messages** published to a provided **topic**

Datatype	Name
Header	header
uint8	nodeId
float64[]	lapTimes
float64	Damage
float64	maxSpeed
uint8	lapsCompleted

Lap status message definition

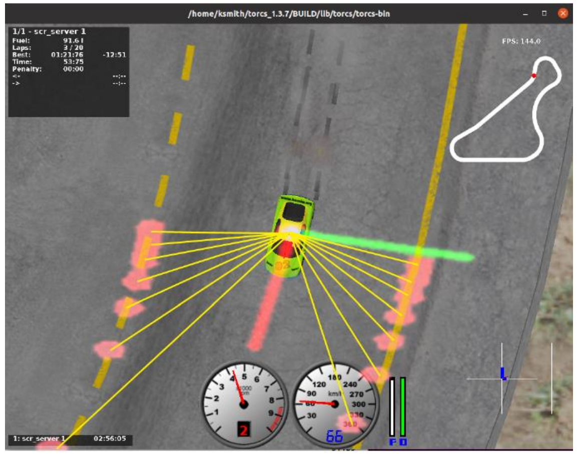


TORCS software architecture



FUZZY CONTROLLERS

- 19 distance sensors providing range from -90 to 90 degrees
- Sensors measure max distance from the vehicle to the edge of the track
- 3 sensors: Front (0 deg), M5 (+- 5 deg) and M10 (+- 10 deg)
- Speed range (0, 200)
- Steering range (-1, 1) full left and full right



Condition	If True
$Front = High$	$Target_{Speed}[0]$
$Front = Med$	$Target_{Speed}[1]$
$Front = Low$ and $M5 = High$	$Target_{Speed}[2]$
$Front = Low$ and $M5 = Med$	$Target_{Speed}[3]$
$Front = Low$ and $M5 = Low$ and $M10 = High$	$Target_{Speed}[4]$
$Front = Low$ and $M5 = Low$ and $M10 = Med$	$Target_{Speed}[5]$
$Front = Low$ and $M5 = Low$ and $M10 = Low$	$Target_{Speed}[6]$
$Front = MAX$ or $M5 = MAX$ or $M10 = MAX$	$Max_{Speed} = 300$

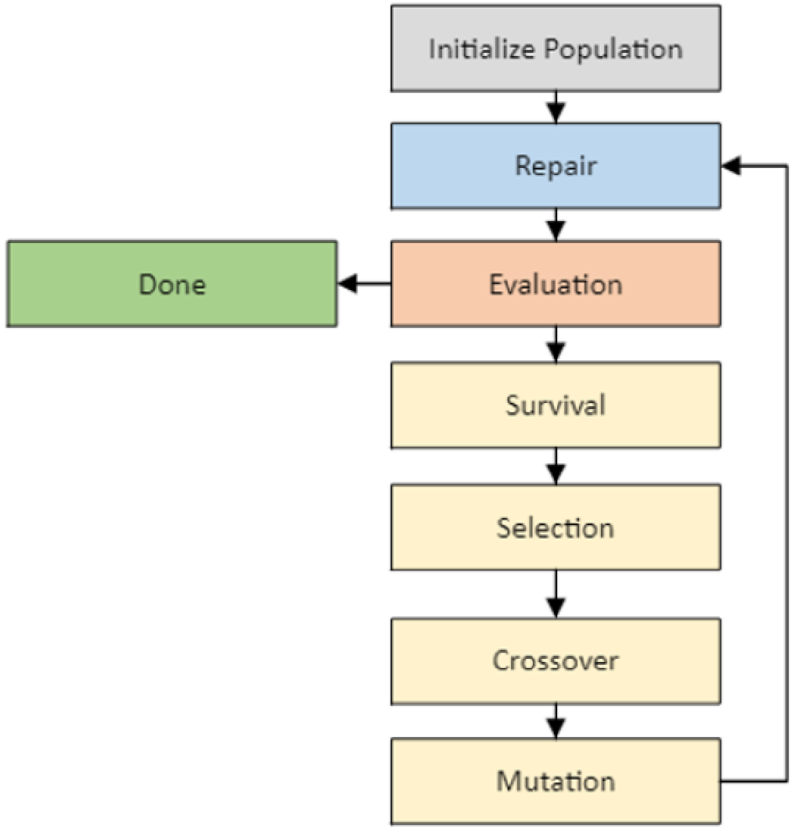
$$Target_{Speed} = [280, 240, 220, 180, 120, 60, 30]$$

Condition	If True
$Front = High$	$Target_{Steer}[0]$
$Front = Med$ and $M10 = High$	$Target_{Steer}[1]$
$Front = Med$ and $M5 = Med$ and $M10 = Med$	$Target_{Steer}[1]$
$Front = Med$ and $M5 = Low$ and $M10 = Med$	$Target_{Steer}[2]$
$Front = Low$ and $M10 = High$	$Target_{Steer}[2]$
$Front = Low$ and $M5 = Med$ and $M10 = Med$	$Target_{Steer}[3]$
$Front = Low$ and $M5 = Low$ and $M10 = Med$	$Target_{Steer}[3]$

$$Target_{Steering} = [0, 0.25, 0.5, 1]$$

GENETICKÝ ALGORITMUS

ŠTRUKTÚRA ALGORITMU



BIT ENCODING

- 10 racing vehicles in one generation
- Enough damages means removing the vehicle from the race
- Chromosome is made up of 3 parts (one for each sensor), each consists of 6 data points describing shape of trapezoid function that define the values of Low, Medium and High
- x0 and x7 are not considered in chromosome encoding since they represent endpoints (0/200)
- Colors for x1 - x6 represent fuzzy value (blue = low, red = medium, yellow = high)

Front						M5						M10					
x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}

INITIALIZATION

- The population was initialized by two different methods in testing

1. Take parameters of a working set as the base of the population and keep the base individual as our first population member. Then mutate the other 9 individuals using the built-in polynomial mutation. The **mutation rate** for this change was **30%**, with an η value of 3. This method ensures **at least one finishing driver** and faster found solution.
2. Random integer sampling function allowed better exploration of other solutions instead of relying on base individual to start

SELECTION, CROSSOVER & MUTATION

- To determine what population members become parents, we will be going with a **tournament selection**
- Since the size of tournament will be 2, it is **binary** tournament selection
- The individual that has the **highest fitness** will be the one selected to move on to the next generation

- For the crossover operator, chosen method was simulated **binary crossover**
- This will produce new parameters in the offspring, also inheriting the parents' old parameters
- The determined crossover rate was set to 0.7

- The mutation operator uses **integer polynomial mutation** for selecting genes to mutate in our chromosomes
- Mutations allow us exploration across search space so that we **do not converge to a local minimum**
- Chosen mutation rate was 0.3

REPAIR & FITNESS FUNCTION

- The repair function ensures that the fuzzy logic parameters meet the constraint for each sensor

$$0 = x_0 \leq x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 = 200$$

- If the values do not follow the constraint, the value is replaced with a random integer value between the points that surround this value

---

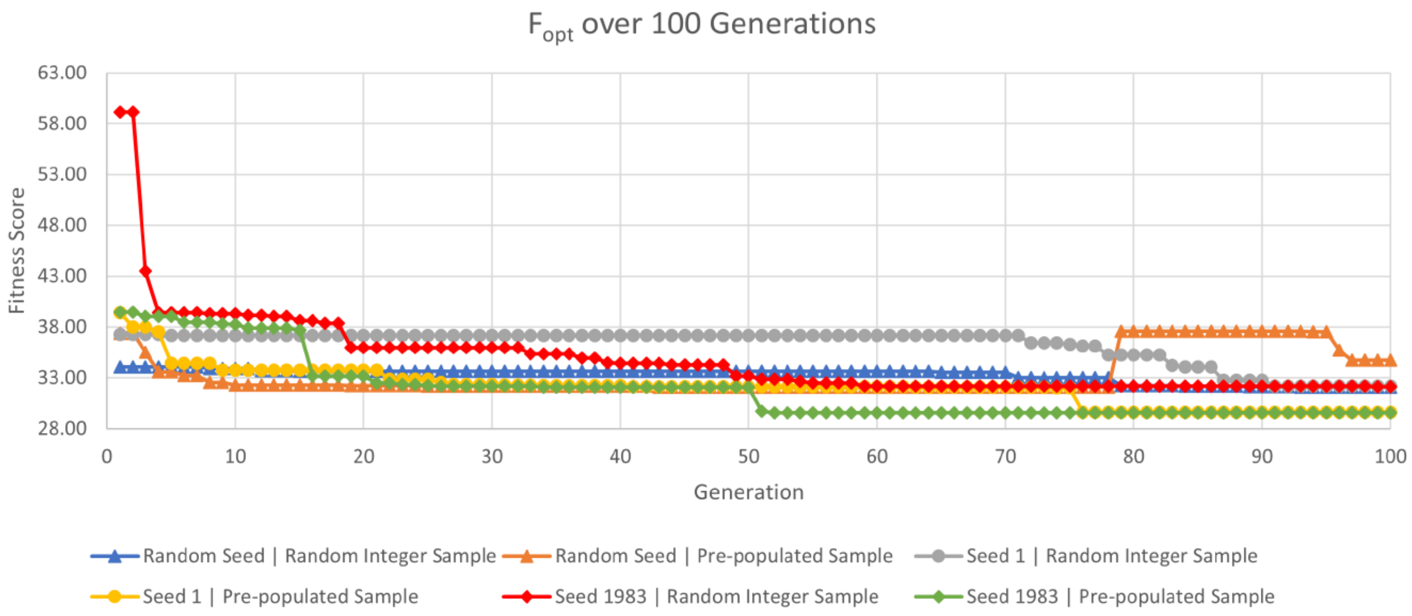
- Goal is to drive fast and safely – function takes in each driver set of lap times in the current generation, along with the damage they received
- The driver with the **lowest score** would be the **fittest** individual

- Scalar variabla **a** gives importance of achieving lower lap time over taking lower damage
- Damage is treated as a time penalty (other solution could be multi-objective optimization)

$$F = D + (\alpha\overline{L})$$

L – lap time
D - damage

RESULTS



Driver	Average Lap Time (10 laps)
Berniw 3	29.66
Olethros 3	32.00
Bt 3	31.78
Tita 3	29.67
Inferno 3	29.67
Lliaw 3	29.67

RESULTS

Table 6: GA Best solution results

Test Case	Seed	Num. of Generations	Initial Pop. Set	Best Solution Value [F]
1	1	50	Pre-populated Set	29.58
2	1	50	Random Integer Sample	37.39
3	1	50	Pre-populated Set with 70% mutation rate	34.44
4	1	50	Random Integer Sample with 70% mutation rate	32.44
5	1983	50	Pre-populated Set	32.17
6	1983	50	Random Integer Sample	31.99
7	1	100	Pre-populated Set	29.59
8	1	100	Random Integer Sample	32.10
9	1983	100	Pre-populated Set	29.54
10	1983	100	Random Integer Sample	32.14
11	Random	100	Pre-populated Set	32.02
12	Random	100	Random Integer Sample	32.09

CONCLUSION

- Driver was able to successfully reduce its lap time through the generations and decrease overall damage
- Results indicate that experiment is right in line, if not better than the built-in AI drivers

- Improvements:
  - mechanism makes sure that the driver stays close to the wall while on a straight segment of the track. Keeping car in its boundary means that the driver made minor steering not to leave our defined boundary. This caused the driver to wobble slightly.
  - extending the project to consider fuel, tire friction, and other vehicle dynamics in the GA calculation could help further lower our lap time.

THANK YOU FOR YOUR ATTENTION!

Odkaz na článok