Hyperparameter Optimization Technique in Autonomous Racing Cars exploiting the Leaky Piece and Conquer Fireworks Algorithm

MyeongJun Kim[1], Gun-Woo Kim[1],*

[1]School of Computer Science, Gyeongsang National University, Jinju, Republic of Korea

Abstract
This paper discusses hyperparameter optimization within existing autonomous driving algorithms used in autonomous racing competitions. Existing autonomous driving algorithms vary significantly in performance depending on the setting of hyperparameters, making it important to find the right hyperparameter. To address these issues, we propose the Leaky Piece and Conquer Fireworks method, which improves the efficiency of hyperparameter optimization. Experimental results indicate that, on average, the general Fireworks algorithm is approximately 35.5 times faster than Random Search. Furthermore, the Piece and Conquer Fireworks algorithm and the Leaky Piece and Conquer Fireworks algorithm are about 69.9 and 77.9 times faster, respectively.

Keywords
Hyperparameter Optimization, Autonomous Driving, Fireworks algorithm

1. Introduction

Recently, the world's top five automakers - Hyundai Motor, Volkswagen, GM, Stellantis Group, and Toyota - have subsidiaries or subsidiaries that research autonomous driving. Even big tech companies such as Google, Apple, and Amazon are conducting self-driving research. In 2010, the VIAC (VisLab International Autonomous Challenge) [1]competition was held, and in 2011, the European Grand Cooperative Driving Challenge (GCDC) [2]was held In line with this trend, self-driving racing competitions such as Indy Autonomous Challenge[3], Roborace[4], and F1TENTH[5] are also gaining popularity. Even ASPIRE, a program development agency under the Abu Dhabi Advanced Technology Research Council (ATRC), will launch the Abu Dhabi Self-Driving Racing League as a car racing competition in the second quarter of 2024, representing the popularity of self-driving racing.

In this paper, we tried to reduce lap time by selecting the classic autonomous driving algorithm FGM (Follow Gap Method), which is widely used in F1TENTH. Classical algorithms such as FGM vary greatly in performance depending on the hyperparameter settings [6][7]. For this reason, we want to reduce lap time, which makes FGM perform through hyperparameter optimization.

ISE 2023: 2nd International Workshop on Intelligent Software Engineering, December 4, 2023, Seoul

*Corresponding author.
gnu_kim98@gnu.ac.kr (M. Kim); gunwoo.kim@gnu.ac.kr (G. Kim)
https://github.com/k0rtal/ (M. Kim); https://sites.google.com/view/gwkim81 (G. Kim)
0009-0008-7454-8251 (M. Kim); 0000-0001-5643-4797 (G. Kim)

FGM is an algorithm for autonomous driving by continuously adjusting the steering wheel toward the largest gap. FGM has several hyperparameters, such as the minimum size gap selected, speed in straight lines and curved sections. In this paper, six hyperparameters were used to construct an autonomous driving algorithm.

Fireworks algorithm was used as an algorithm for hyperparameter search. Fireworks algorithm is a sampling-based search algorithm and shows good performance even if the form of the objective function is complex. In addition, Piece and Conquer (PnC) Fireworks algorithm, Leaky Piece and Conquer (Leaky PnC) Fireworks algorithm was proposed and compared with Random Search, general Fireworks algorithm.

2. Related Works

2.1. Follow Gap Method(FGM)

The Follow Gap Method (FGM)[8] is an algorithm that finds the largest and deepest gap in the data received by LiDAR and performs autonomous driving. FGM can navigate the path in a simple and effective way in a dynamic environment and is suitable for real-time operation. However, sometimes inefficient routes can be chosen because they follow the largest intervals.

The hyperparameters of the FGM are as follows. PREPROCESS_CONV_SIZE is a hyperparameter that sets how many pieces of data should be converted during pre-processing. BEST_POINT_CONV_SIZE is a hyperparameter representing the CONV value used to find the maximum gap size. MAX_LIDAR_DIST is a hyperparameter indicating how far to look. STRAIGHTTS_SPEED is a hyperparameter indicating a speed in a straight line section. CORNERS_SPEED is a hyperparameter indicating a speed in a curved section.

2.2. Fireworks Algorithm

Fireworks Algorithm[9] is one of the search algorithms for solving optimization problems. This algorithm works by mimicking Firework, which models the process of finding solution as various fireworks explode. Fireworks Algorithm is effective in finding global optimal solution and can be applied to a variety of optimization problems.

The Fireworks Algorithm proceeds as follows. 1) Firework Concept: models the solution search process as an object called "firework". Each firework indicates the location of the candidate solutions. 2) Firework Explosion: The key idea of the algorithm is to divide fireworks into several groups, share fireworks among the groups, and improve the solution of discovery. Furthermore, if a better solution is found, more fireworks are dropped at that location, accelerating the process of "exploding" and discovering the solution. 3) Grouping: Fireworks are divided into groups. Each group performs a solution search independently of the other group, and when it finds a new solution, it shares it with the other group. These grouping methods help to explore both global and regional optimal solutions simultaneously. 4) Firework Update: After discovering new solutions, update the location of the firework, and if a better solution is found, drop more fireworks into that location to improve the solution. 5) Termination condition: The algorithm terminates after meeting the termination condition or after a certain number of repetitions.

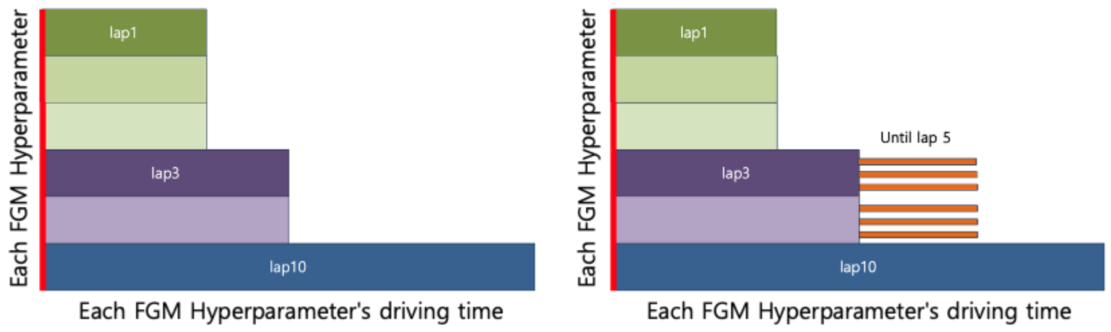

Figure 1: (Left) Piece and Conquer Fireworks Algorithm, (Right) Leaky Piece and Conquer Fireworks Algorithm

3. Proposed Method

In this paper, the Piece and Conquer Fireworks Algorithm and Leaky Piece and Conquer Fireworks Algorithm were proposed based on the existing Fireworks algorithm for explore hyperparameters.

3.1. Piece and Conquer Fireworks Algorithm

The first algorithm proposed in this paper, Piece and Conquer (PnC) Fireworks Algorithm, means to conquer by piece. A typical Fireworks algorithm finds the optimal hyperparameter using only data that has completed 10 laps if the target lap is 10 laps.

In the PnC Fireworks algorithm, as shown in **Figure 1** (Left), the optimal result n end points of the lap1 completion piece become the n start points of the lap3 completion piece, and the optimal result n end points of the lap3 completion piece become the n start points of the lap10 completion piece. This is used for navigation of lap10 targeting data fragments of lap1 and data fragments of lap3. As such, the PnC Fireworks algorithm use more data fragments to perform more efficient hyperparameter optimization.

3.2. Leaky Piece and Conquer Fireworks Algorithm

The second algorithm proposed in this paper, Leaky Piece and Conquer (Leaky PnC) Fireworks Algorithm, takes the similar structure with PnC Fireworks Algorithm.

In the Leaky PnC Fireworks algorithm, as shown in **Figure 1** (Right), the optimal result n end points of the lap1 completion piece are the n start points of the lap3 completion piece, and the optimal result n end points of the lap3 completion piece are the n start points of the lap10 completion piece. This is used for navigation of lap10 targeting data fragments of lap1 and data fragments of lap3. Unlike PnC, Leaky PnC searches for lap3, and even number searches for lap5 when searching for lap3.

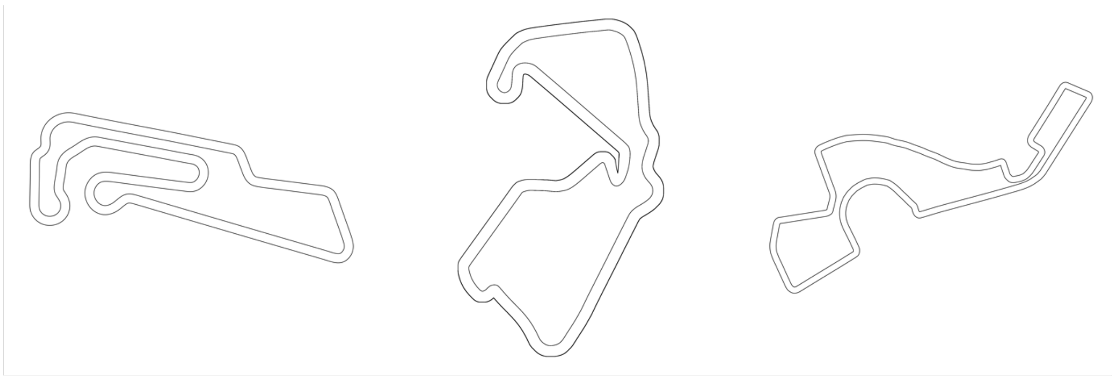
Figure 2: Map image. (Left) Oschersleben, (Middle) Silverstone, (Right) Sochi

Table 1
FGM parameter setting range.

Name of Hyperparameters	Range of Hyperparameter setting range
BUBBLE_RADIUS	90-240 (6 pieces) at 30 interval
PREPROCESS_CONV_SIZE	2-5 (4 pieces) at 1 interval
BEST_POINT_CONV_SIZE	70-120 (6 pieces) at 10 interval
MAX_LIDAR_DIST	2000000-5000000 (4 pieces) at 1000000 interval
STRAIGHTS_SPEED	7.0-11.2 (15 pieces) at 0.2 interval
CORNERS_SPEED	5.0-9.2 (15 pieces) at 0.2 interval

4. Experiments

In this paper, the driving of the autonomous driving algorithm used a python gym-based F1TENTH simulator[10]. This simulator can customize various map and dynamics values. As shown in **Figure 2**, Oschersleben, Silverstone, and Sochi were used for the map. And the basic values of the simulator dynamics values were used for dynamics values.

In this paper, the number of FGM parameters was set to 6, and the range for each parameter was set as shown in **Table 1**. The parameters of FGM can also be applied beyond the ranges specified in Table 1, but we have defined the ranges and intervals as shown in Table 1 to enable the determination of relative parameter quality rankings.

The experimental process is as follows: First, we record the performance results for all possible combinations of FGM parameters corresponding to Table 1. In other words, we save 129,600 records of driving performance, given that there are $6 \times 4 \times 6 \times 4 \times 15 \times 15 = 129{,}600$ possible combinations of FGM parameters. This enables us to establish relative rankings for all 129,600 FGM parameter configurations. Next, based on the recorded performance data, we applied Random search, general Fireworks, PnC Fireworks, and Leaky PnC Fireworks algorithms.

Table 2
Result of Random search, general Fireworks, PnC Fireworks, Leaky PnC Fireworks.

Map	Algorithm	Median time(sec) of 30 samples	Comparison with Random search (multiple)
Oschersleben	Random search	5,265,512	-
Oschersleben	general Fireworks	19,723	43.9
Oschersleben	PnC Fireworks	58,361	90.2
Oschersleben	**Leaky PnC Fireworks**	50,629	**104**
Map	Algorithm	Median time(sec) of 30 samples	Comparison with Random search (multiple)
Silverstone	Random search	6,879,384	-
Silverstone	general Fireworks	341,182	20.1
Silverstone	PnC Fireworks	160,371	42.8
Silverstone	**Leaky PnC Fireworks**	148,949	**46.1**
Map	Algorithm	Median time(sec) of 30 samples	Comparison with Random search (multiple)
Sochi	Random search	4,614,436	-
Sochi	general Fireworks	108,361	42.5
Sochi	PnC Fireworks	60,158	76.7
Sochi	**Leaky PnC Fireworks**	55,164	**83.6**

5. Result

The experiment was performed 30 times for each search algorithm, and the median value of the results executed 30 times was used. This increases the statistical reliability of search-based algorithm results. **Median time(sec) of 30 samples** means the time it took to find the optimal parameter. The simulator shows the median value of the results executed when the corresponding algorithm is tested 30 times, and if the simulator accelerates, it can be executed 8 to 10 times faster. **Comparison with Random Search (multiple)** is a value that indicates how many times faster the corresponding algorithm is compared to Random Search.

Table 2 shows that the Leaky PnC Fireworks algorithm is the best for all three maps: Oschersleben, Silverstone, and Sochi. In the Oschersleben map, the general Fireworks algorithm is 43.9 times faster, the PnC Fire-works algorithm is 90.2 times faster, and the Leaky PnC Fireworks algorithm is 104 times faster than Random Search. In the Silverstone map, the general Fireworks algorithm is 20.1 times faster, the PnC Fireworks algorithm is 42.8 times faster, and the Leaky PnC Fireworks algorithm is 46.1 times faster than Random Search. In the Sochi map, the general Fireworks algorithm is 42.5 times faster, the PnC Fireworks algorithm is 76.7 times faster, and the Leaky PnC Fireworks algorithm is 83.6 times faster than Random Search.

6. Conclusion

In this paper, the objective was to enhance the performance of FGM through hyperparameter optimization, in other words aiming to improve lap times. Overall, the PnC Fireworks algorithm is about twice as fast as the general Fireworks algorithm, and the Leaky PnC Fireworks algorithm is about 1.1 times faster than the PnC Fireworks algorithm. These experimental results demonstrated that the two types of PnC Firework algorithms proposed in this paper efficiently search for hyperparameters and achieve innovative results in simulations. Furthermore, it is expected to provide important developments in the field of hyperparameter optimization and contribute to the performance improvement of future autonomous driving systems and other complex systems.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, Science and Technology (NRF-2021R1G1A1006381).

References

[1] M. B. et al, Viac: An out of ordinary experiment, 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany (2011) pp.175–180.
[2] V. Nunen, Ellen, et al, Cooperative competition for future mobility, IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 3 (Sept. 2012) pp.1018–1025.
[3] Indy Autonomous Challenge. URL: https://www.indyautonomouschallenge.com/.
[4] Roborace. URL: https://www.roborace.com/.
[5] F1TENTH. URL: https://www.f1tenth.org/.
[6] Badue, Claudine, et al, Self-driving cars: A survey., Expert Systems with Applications 165 (2021) 113816.
[7] Moll, M. et al, Hyperplan: A framework for motion planning algorithm selection and parameter optimization., 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2021).
[8] Sezer, Volkan, M. Gokasan, A novel obstacle avoidance algorithm:"follow the gap method"., Robotics and Autonomous Systems 60.9 (2012) 1123–1134.
[9] Tan, Ying, , Y. Zhu, Fireworks algorithm for optimization., Advances in Swarm Intelligence: First International Conference, ICSI 2010, Beijing, China, June 12-15, 2010, Proceedings, Part I 1. Springer Berlin Heidelberg (2010).
[10] O'Kelly, M., Zheng, H., Karthik, D., F1tenth: An open-source evaluation environment for continuous control and reinforcement learning., Proceedings of Machine Learning Research 123 (2020).