Evolutionary F1 Race Strategy

Andrea Bonomi*
Evelyn Turri*
Giovanni Iacca
{andrea.bonomi-2,evelyn.turri}@studenti.unitn.it,giovanni.iacca@unitn.it
University of Trento
Trento, Italy

ABSTRACT

Formula 1 is a highly competitive and ever-evolving sport, with teams constantly searching for ways to gain an edge over the competition. In order to meet this challenge, we propose a custom Genetic Algorithm that can simulate a race strategy given data from free practices and compute an optimal strategy for a specific circuit. The algorithm takes into account a variety of factors that can affect race performance, including weather conditions as well as tire choice, pit-stops, fuel weight, and tire wear. By simulating and computing multiple race strategies, the algorithm provides valuable insights and can help make informed strategic decisions, in order to optimize the performance on the track. The algorithm has been evaluated on both a video-game simulation and with real data on tire consumption provided by the tire manufacturer Pirelli. With the help of the race strategy engineers from Pirelli, we have been able to prove the real applicability of the proposed algorithm.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Genetic algorithms**; *Artificial life*; *Planning and scheduling*; • **Theory of computation** → **Models of learning**; **Evolutionary algorithms**.

KEYWORDS

Formula 1, Race Strategy, Evolutionary Computation

ACM Reference Format:
Andrea Bonomi, Evelyn Turri, and Giovanni Iacca. 2023. Evolutionary F1 Race Strategy. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3583133.3596349

1 INTRODUCTION

F1 is one of the most difficult race competitions, and it offers a number of incredibly interesting challenges from a computational perspective. One of these problems is the optimization of the race strategy, that is precisely the focus of the present paper.

As a matter of fact, modern F1 cars have hundreds of sensors that stream data from the car to the team garage. Team engineers need to understand those data and the correlation between them, in order to develop an effective strategy for the race. Therefore, computational intelligence approaches can be a valuable tool in this domain. However, F1 is an extremely closed world, where data are usually kept secret, and it is very difficult to find state-of-the-art works on this topic, with publicly available results and data. Still, it is possible to identify three main approaches that can be used for this type of problem: Monte Carlo (MC) sampling, Reinforcement Learning (RL), and Evolutionary Algorithms (EAs).

According to an article published in 2015 on "F1 magazine" [1], F1 teams mainly use MC simulations in addition to game theory techniques to obtain optimal race strategies given the telemetry data. Typically, each team runs thousands of simulations to get its final strategies. This process, however, can be very time-consuming and demanding in terms of computational resources. Moreover, with MC simulations there is still a problem of human choices, as discussed in [3]. For this reason, [3, 12] introduce the idea to use a machine-only method, namely RL. For instance, in [3] the total reward is computed as the number of positions gained or lost at the end of the race w.r.t. the other drivers. However, as with MC simulations, also these RL-based approaches usually require a high computational time.

A possibly computationally cheaper alternative is to use EAs [2]. To the best of our knowledge, the only work that applied an EA to this problem is [13]. However, that study presented a very basic EA for optimizing the car settings, with rather stringent simplifications e.g. related to the fact that only binary parameters are considered (for this reason, a direct comparison with the method proposed in the present paper is not possible).

In this paper, we attempt to go beyond the previous literature by presenting a custom Genetic Algorithm (GA) with much more complex features and degrees of freedom, which differently from the algorithm proposed in [13] does not work only with binary parameters. Moreover, our algorithm takes in consideration the telemetry data with a more appropriate representation. As we will show in the following, the proposed algorithm has competitive performances both on simulated and real data (the latter provided by engineers from Pirelli) and is very fast in terms of compute time.

The rest of the paper is structured as follows. In the next section, we describe the background concepts of the race model. Then, in Section 3 we describe the proposed approach based on GA. In Section 4 we present the experimental setup, followed by the numerical results in Section 5. Then, Section 6 provides the main conclusions of this study.

*Equal contributions.

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07…$15.00
https://doi.org/10.1145/3583133.3596349

2 BACKGROUND

In the following, we describe the background concepts at the basis of the race strategy optimization problem.

2.1 Tires

A tire compound defines the type of tires that are mounted on the car. Compounds are divided into dry compounds and wet compounds, that are used in dry and wet track conditions respectively. Dry compounds are in turn divided into:

- Soft (S): it has the highest grip to increase the performances, but it wears out very quickly;
- Medium (M): it has a good grip and it does not wear out as quickly as the Soft compound;
- Hard (H): it has the lowest grip and it reduces the performances w.r.t. the Medium compound. However, it wears out slowly.

Instead, wet compounds are divided into:

- Intermediate: it is the fastest and is usually preferred;
- Wet: it is used when the track is very wet and it is the only choice given these weather conditions.

The time loss due to tire wear is computed as the time difference at each lap. This computation is done after removing all possible losses, such as the fuel weight. The time loss between tire compound is computed as the time difference between the first valid lap-time made with a tire compound and the first valid lap-time made with the Soft compound (every lap-time is made with the same amount of fuel).

2.2 Fuel

Every car loads a different quantity of fuel and cannot refuel during a race. The main aspect to consider is that the more fuel is loaded, the slower the car will be. The goal is to find a trade-off between loading enough fuel to finish the race (without exceeding the maximum fuel load of 110 kg) and loading the minimum possible amount of fuel to be competitive. The fuel consumption is computed from the telemetry data as the slope of the linear interpolation between the starting and the final fuel load. Finally, the time loss due to the fuel load is computed given the fuel consumption and the lap time difference at each lap.

2.3 Weather

Weather is obviously not known a priori with certainty, but it is a crucial factor for an optimal strategy. The information given by the weather forecast for the race day is added in our algorithm. This information is added through a value for each lap, i.e., the amount of rain on the track. In particular, this quantity is an integer value in $[0, 100]$, where 0 represents 0 mm of water, while 100 is for those cases where the rain level is equal to or over 5 mm. A graphical representation of the weather data is shown in Fig. 1.

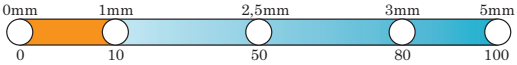

Figure 1: Range for weather data. Orange is for dry weather, while blue indicates rain.

2.4 Rules and constraints

During a race there are different rules to follow, which must be taken in consideration in the algorithm as constraints: more specifically, two rules are for the tire, and one for the fuel.

Regarding the first constraint for tires, during a race weekend there is a limited amount of tire compounds that the tire manufacturer provides at the beginning of the weekend. It is up to the team to choose carefully which tire compounds to use in free practices, because that set of tires is not made available for the race. Indeed, teams can plan their weekend according to the race strategy: if the predicted strategy needs two Soft and two Medium compounds, then the team can organize the weekend to save, or use the least possible, those tire sets for the race. This constraint is not added.

The second constraint for tires imposes that, during a race, the car has to mount at least two different compounds, consequently there has to be at least one pit-stop. This constraint is added in the algorithm, and every strategy is valid only if it has at least one compound change. There is only one case where this constraint can be violated: when a car mounts a wet compound in all laps.

Regarding the fuel, as said earlier a car cannot refuel during a race. This constraint plays an important role in the algorithm, because the strategy should be able to have enough fuel in order to finish the race, but the minimum possible to be competitive.

3 PROPOSED GENETIC ALGORITHM

In this section the different aspects of the GA are explained.

3.1 Individual representation and fitness

Every strategy is considered as an individual in the GA, with its corresponding genotype and fitness. The genotype is composed by the following variables:

- `TireCompound`: indicates which tire compound is mounted at each lap of the race;
- `FuelLoad`: contains, for each lap, the weight of the loaded fuel;
- `PitStop`: indicates, for each lap, if a pit-stop is made or not;

Other individual information, that are not included in the genotype but are useful for the computations, are:

- `TireWear`: contains, at each lap, the tire wear % of each tire;
- `TireAge`: contains, at each lap, the tire age of the mounted tires;
- `LapTime`: at each lap a predicted lap time is generated and stored in this array;
- `NumPitStop`: the total number of pit-stops;
- `Weather`: quantity of rain on the track at each lap, stored as presented in Section 2.3;
- `Valid`: indicates if the strategy violates the constraints;
- `TotalTime`: the total race time obtained by summing all lap times. Notice that `TotalTime` is expressed in ms.

The `TotalTime` is used as the fitness value for each individual, to be minimized. This value is obtained as a sum of multiple time losses w.r.t. the best lap-time, as proposed in [10]:

$$TotalTime = \sum_{lap=1}^{laps} (t_{best} + t_{compound} + t_{tire\_wear} + t_{fuel} + t_{pit-stop} + t_{weather}) \quad (1)$$

where:

- t_{best} is the best time obtained in the free practices, removing all the possible time losses. It is expected to be the time of the qualifying, while it is difficult to achieve it on race session due to the multiple time losses;
- $t_{compound}$ is the time loss between the different tire compounds;
- t_{tire_wear} is the time loss due to the tire wear;
- t_{fuel} is the time loss due to the fuel weight. This value increases linearly, usually it is around 0.03 s per kg of fuel [10];
- $t_{pit-stop}$ is the average time loss due to a pit-stop;
- $t_{weather}$ is the time loss due to weather.

Note that the GA takes the `TotalTime` value in ms and converts it into hh:mm:ss.ms for better readability.

3.2 Algorithm flow

The proposed GA works as follows:

(1) Build the initial population randomly. The constraints are then checked, in order to verify if each individual is valid or not.
(2) Check if there are individuals that are equal, and in that case keep only one of them in the population.
(3) Sort the population with a custom penalty function and perform selection.
(4) From the selected individuals, arrange them in random pairs and apply crossover with probability $p_{crossover}$.
(5) Apply to each offspring mutation with probability $p_{mutation}$.
(6) Add the resulting individuals to the new population.
(7) Add other individuals, built randomly, to the population, until the population has the right number of individuals.
(8) Repeat from step (2) until the maximum no. of generations.

3.3 Selection and replacement

The selection process is based on the fitness value with an application of a penalty in some specific cases. It is important to notice that a specific strategy could be optimal, but it could also break only a constraint by a little, e.g., for the fuel or the compound change. Even if the strategy is not valid, with some minor changes, it could be turned into a very good solution. To make this happen, the selection makes a first check about the validity of each solution and then it applies a penalty starting from the `TotalTime` of each non-valid strategy. The penalty is built in a dynamic way. At each generation, the algorithm computes the following steps:

(1) Compute Δ. For each i-th individual in the population, Δ_1 is the absolute value of the difference between the `TotalTime` and the best time found so far:

$$\Delta_i = |TotalTime_i - BestTime|. \tag{2}$$

(2) Compute α. The latter is a multiplicative value, used to calculate the penalties, and it is a value related to the number of generation. The later the generation, the bigger α:

$$\alpha = e^{1+\frac{N_{step}}{N_{iteration}}}. \tag{3}$$

(3) Compute the penalties. For each i-th individual, the algorithm computes the penalties by normalizing Δ_i and multiplying it by the α factor calculated in the previous step:

$$penalty_i = \frac{\Delta}{\max_{i \in pop}(\Delta_i)} \times \alpha. \tag{4}$$

(4) At this point, if the strategy is valid no penalty is applied; otherwise, we check the constraints:
   (a) Constraint on the compound change: if this is not verified, we multiply the penalty value by the same α factor mentioned before. This is done because this constraint is verified or not, i.e., there is no intermediate possibility.
   (b) Constraint on the fuel load: the constraint imposes that the fuel load must be bigger than or equal to zero. Otherwise, an exponential penalty is applied as follows:

$$e^{|last\_lap\_fuel\_load|}. \tag{5}$$

Finally, the population is sorted based on the penalty, and selection is applied. Only a part of the population is selected for the next generation. The old population is replaced through elitism: n best individuals are maintained from the previous generation, to prevent loss of the best individuals by effect of mutations or sub-optimal fitness evaluations. In order to maintain keep the size of the population (pop) fixed, it is necessary to find the right n for the proposed algorithm, namely: $n = \lfloor pop/7 \rfloor$. This value results from the following reasoning. In order to compute the right n, we need to take into account how many offspring are created through crossover and mutation at each generation. Given the fact that there is only one type of crossover operation, at most n new children are created at each generation. For mutation, there will be at maximum 5 new children for each parent, 4 from the 4 types of mutations, and one created by the mutations applied together (see Section 3.5). Then, it results that:

$$n + offspring\_crossover + offspring\_mutation = pop$$
$$n + n + 5 \times n = pop \rightarrow n = \lfloor pop/7 \rfloor.$$

In this way, the number of individuals in the new population will always be less than or equal to pop. If the number is less than pop, the new population is filled with random children until it reaches pop. Then, the new population replaces the old one.

3.4 Crossover

The implemented crossover works only on the fuel. If the crossover is applied on two strategies, they exchange their initial fuel, creating two new individuals for the next population, as shown in Fig. 2. Their fitness values are computed again with the new initial fuel load. Note that the crossover has been applied only on the fuel and not on the other two variables, i.e., the tires and pit-stops, because in preliminary experiments we noted that the application of crossover on those variables was causing problems with the feasibility of the resulting strategies (e.g., resulting in a number of pit-stops not compatible with the number of compounds).

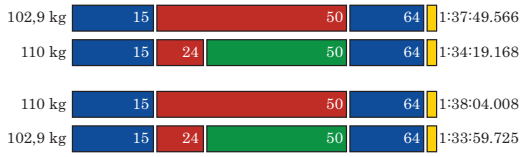

Figure 2: Example of crossover operation.

3.5 Mutation

The mutations implemented are of four types and they are all applied with the same probability $p_{mutation}$. One of them regards the tires, two regard the pit-stops, and the last one regards the fuel load. The first mutation works by taking all the different compounds used in a strategy, randomly picking one of them, and changing it with another compound until another pit-stop is made or the race ends. The two types of mutations on pit-stops have opposite roles: the first one has the goal to remove the unnecessary pit-stops in the strategy, while the second one tries to add a pit-stop in those cases where a certain tire is too worn out. They work as follows:

- Remove a pit-stop: the algorithm takes a random number i from 1 to the total number of pit-stops in that specific strategy, then the i^{th} pit-stop is removed and the tire compound we would have used until the next pit-stop or the end of the race is replaced with the tire of the previous laps.
- Add a pit-stop: the algorithm takes a random number j in the range between 1 and the total number of laps of the race, and it adds in the j^{th} lap a pit-stop with new tires.

The fourth mutation works on the initial fuel load. If this is applied, the initial value of the fuel is changed with a random value uniformly sampled from $[Fuel\_Load_{initial} - 10; Fuel\_Load_{initial} + 10]$ where $Fuel\_Load_{initial}$ is the initial value of the fuel load.

4 EXPERIMENTAL SETUP

In this section, we explain the experimental setup and the data collection. Code, data, and results can be found on GitHub[1].

The experimental campaign is divided in two parts: experiments on simulated data, and experiments with real data provided by Pirelli. We describe these two sets of experiments below.

4.1 Simulated data experiments

In the first part of our experiments, the F1 2021 simulation game has been used to gather simulated telemetry data. The GA has been tested on the circuits of Bahrain, Montreal, Monza, Portimao, Spielberg, and Zandvoort. We should highlight the fact that, while the data acquired from simulation are simplified, they are still reasonably plausible. Moreover, the driver collecting the simulated data was an amateur driver, thus we cannot rule out the possibility that the results of the algorithm on the simulations have been affected by the driver abilities. Yet, as we will see below, these results provide interesting insights.

The race weekend considered in our experimentation is arranged as follows: (1) Free Practices (FP1, FP2, FP3); (2) Race. Specifically, telemetry data are collected on the free practices and are then used to compute the race strategy. Qualifying sessions are not considered, since the aim of this paper is not to search for the best tire and fuel load to achieve the best time for qualifications.

4.1.1 Data collection. Data are collected through a script that captures all the telemetry data available from the simulator. Data are then stored and elaborated through the GA (in particular, the parameters needed for computing all the time losses that are necessary for computing the fitness function shown in Eq. (1)). In particular, data are fit into a linear function to obtain the coefficients for the time losses (e.g., the different values of tire compound, tire wear, weather conditions, and fuel load).

4.1.2 Weather conditions. In order to evaluate the proposed GA, we consider two different scenarios, namely a simple dry condition, and a complex mixed weather condition, defined as follows:

- *Dry condition*: This is the simplest case, where only the dry compounds are involved.
- *Mixed condition*: In this case both dry and wet conditions happen. The considered weather condition assumes that for the first 15 laps the track is wet, then it becomes dry for about 20/35 laps, depending on the circuit, then wet again for 15 laps until the last 4/5 laps, which are in dry condition. This is a very unrealistic situation, but the goal is to stress the GA and see how it handles unexpected conditions.

4.2 Real data experiments

For these experiments, we got in contact with race engineers from Pirelli, who provided the relevant parameters for modeling the tire wear of some of the 2021 circuits: Bahrain, Barcelona, Monza, Portimao, Silverstone, Spielberg, and Zandvoort. The parameters provided by Pirelli are: $delta$, $degradation$, and the maximum number of laps for each tire given a certain circuit. The $delta$ is a value that expresses the time difference w.r.t. the Soft compound. The $degradation$ is a value that express the time loss at each lap due to degradation. In other words: considering a linear degradation of the tire, where the number of laps is the x-axis and the time loss is on the y-axis, the $delta$ is the intercept while the $degradation$ is the slope of such linear degradation. Note that we cannot provide these data, because they are not publicly available.

Since the initial fuel load was kept secret, in this case an exact computation for the fuel usage is not possible. A feasible computation is $110/total\_laps$ (kg/lap), and the time loss is fixed at 0.3 s/kg. Moreover, in this case the weather forecast is not taken in consideration since Pirelli proposals are for dry races only. Therefore, for the real data experiments the fitness function is adapted as follows:

$$TotalTime = \sum_{lap=1}^{laps} (delta + degradation \times lap + t_{fuel} + t_{pit-stop}) \tag{6}$$

In this case the GA returns, for each run, the fastest 1-pit strategy and the fastest 2-pit strategy. The equal strategies (in terms of number of pit-stops) are combined together in order to provide a pit-window as Pirelli does in the real practice. Note that, given the hypothesis explained in the first part of this Section, Eq. (1) and (6) are still comparable. The two formulations are different only because of the different nature of synthetic and real data.

Proof. In Eq. (1) the following terms are modified as follows:

- t_{best} is not provided for real data and actually is not needed for the goal of the algorithm, so it can be deleted.
- $t_{compound}$ corresponds to the $delta$ parameter provided by Pirelli.
- t_{tire_wear} is computed as $degration \times lap$.
- $t_{weather}$ can be deleted, given the fact that the weather forecast is not taken in consideration by Pirelli.

For these reasons, Eq. (1) and Eq. (6) are equivalent formulations of the problem. □

[1] https://github.com/bonom/Evolutionary-F1-Race-Strategy

Note that the race times are very different between simulated and real data experiments. This is because Eq. (6) does not have the parameter t_{best} that Eq. (1) instead has. As said this parameter is not available in real data experiments, thus it has been removed from the computations, leading to lower race times.

4.3 GA parameters

The GA is executed for 30 runs for each circuit and, in the case of simulated data, also for each type of weather condition. The hyperparameters used for the GA are summarized in Table 1. These parameters have been set empirically.

Table 1: Hyperparameters used for the GA.

pop	No. generations	$p_{mutation}$	$p_{crossover}$
250	1000	0.9	0.6

4.4 Baseline algorithm: reduced brute-force

Since for the problem at hand there is no specific state-of-the-art algorithm to compare with, we implemented a (reduced) brute-force method in order to have a baseline for the results of the proposed GA. The search space of the brute-force algorithm can be seen as a tree: starting from the root, at each lap there are different children (one for each possible tire choice), and for each children there are two other possible children for the choice of making a pit-stop or not. All the possible paths starting from the root and ending in the leafs are the possible strategies for a race. The best strategy (path of the tree) is the one with the lowest total time computed by Eq. (1). The space of all the possible solutions would be however too big to explore: in fact, the complexity of a full brute-force would be $O(10^n)$, where n is the number of laps. To ensure a feasible computation, the number of pit-stops is therefore limited to be below than or equal to 2. Moreover, the brute-force is forced to choose a specific tire given the weather condition, to decrease the complexity even more. While these assumptions make the computation feasible, they prevent finding some possible strategies with more than 2 pit-stops that could be optimal. Yet, this form of reduced brute-force can provide baseline results that are sufficient for our purposes. The choice for the baseline fell on the brute-force approach rather than on MC sampling, as mentioned in Section 1, for two reasons. The MC method outputs a certain strategy, which is connected with a certain probability of winning the race, while brute-force deterministically outputs the best strategy. Moreover, brute-force was a better and more convenient choice than the MC algorithm, because the last one typically needs more computational power for computing accurate results, and it is especially useful with a higher number of parameters to optimize. The proposed algorithm can work with any number of variable, without change anything in the actual flow (Section 3.2).

5 RESULTS

We present now the results obtained on the simulated and real data by comparing the proposed GA with the brute-force on the various circuits and weather conditions mentioned above.

5.1 Simulated data experiments

5.1.1 Dry condition. In Table 2, the results obtained for the circuits under consideration are reported. The table shows: the brute-force final race time, the best final race time achieved by the GA, the mean of all the final race times achieved by the GA across the different runs, the standard deviation of the same final race times, and the difference between the GA best and the brute-force one.

Table 2: Simulated data experiments - Sunny weather condition results. "Diff." indicates the difference between the GA best and the brute-force results. Results are displayed as hh:mm:ss.ms.

Circuit	Brute-force	GA Best	GA Mean	GA Std	Diff.
Bahrain	1:24:03.264	1:24:03.261	1:25:40.219	2:19.233	- 0.003
Montreal	1:25:21.596	1:25:17.117	1:25:54.326	1:43.793	- 4.479
Monza	1:14:29.778	1:14:29.838	1:16:23.050	2:28.598	+ 0.060
Portimao	1:28:17.329	1:28:17.686	1:29:14.057	2:11.722	+ 0.357
Spielberg	1:19:49.182	1:19:49.402	1:20:20.591	1:31.701	+ 0.220
Zandvoort	1:28:26.830	1:28:26.646	1:28:48.877	1:16.399	- 0.184

We can see that the GA manages well this type of condition. It misses the best results found by brute-force by only a few ms, while in some cases it even beats the brute-force strategy. In Bahrain, the best solution found by brute-force is slightly better than the one provided by the GA because of some approximations errors. In Montreal, the GA is capable of beating the brute-force due to the pit-stop limitations on the brute-force: in fact, the fastest strategy is a 3-pit, while the brute-force is limited at two. In Monza, the GA also performs well: it finds a good strategy w.r.t. the brute-force, with a difference of one lap on two tire compounds. On Portimao the GA struggles more but is able to obtain a very good strategy: the brute-force strategy performs more laps with the Medium compound, while the GA strategy performs more laps with the Soft compound. In Spielberg, the GA has a good performance: as in Monza, it only misses by two laps the best performance. Finally, in Zandvoort, the GA outperforms the brute-force because it finds that a 3-pit strategy is better than a 2-pit one.

5.1.2 Mixed condition. In Table 3, the results for each circuit in consideration are reported. The table shows the best final race time achieved by the GA, the mean of all the final race times achieved by the GA across the different runs, and the standard deviation of the same final race times. In this case brute-force is not considered as baseline because it is not computationally feasible.

We can see that the GA managed well also the mixed weather condition. In fact, the algorithm was able to return a strategy in a short amount of time. Moreover, for every circuit the proposed strategy is feasible.

5.2 Real data experiments

Tables 4-10 summarize the results obtained on the real data experiments on each circuit. Accordingly, Figures 3-16 visualize the best strategies proposed by the GA and the ones proposed by Pirelli for each circuit before the race day. As for the GA results, we present the best strategies found across the 30 runs available for each circuit. The Pirelli strategies are publicly available, respectively for

Table 3: Simulated data experiments - Mixed weather condition results. Results are displayed as hh:mm:ss.ms.

Circuit	GA Best	GA Mean	GA Std
Bahrain	1:28:35.913	1:28:58.271	19.633
Montreal	1:30:48.090	1:31:46.961	59.287
Monza	1:19:00.718	1:19:44.536	31.977
Portimao	1:33:57.904	1:34:34.585	23.166
Spielberg	1:24:36.648	1:25:48.128	57.578
Zandvoort	1:33:02.185	1:34:00.552	41.004

Bahrain [4], Barcelona [5], Monza [6], Portimao [7], Silverstone [11], Spielberg [8], and Zandvoort [9]. In the figures, rows are ordered from the fastest to the slowest strategy. Red indicates the Soft compound; yellow indicates the Medium compound; white indicates the Hard compound. The green box indicates the pit-window.

It can be seen that on the real data experiments the GA performed very well: in all cases, it equals or outperforms the proposal of Pirelli. In the following, we compare in detail the results of the GA and the strategies proposed by Pirelli for each circuit.

Bahrain. In the circuit of Bahrain the GA finds a well-performing strategy. Indeed, it determines that the fastest strategy is a 2-pit strategy with the Hard compound involved. It outperforms the strategy proposed by Pirelli. However, it is worth noticing that it is not obvious to have two new Hard compounds available on the race day. Indeed, usually teams have a new Hard compound and a used one. The strategies are shown in Fig. 3 and Fig. 4.

Table 4: Bahrain results.

Order	Output	Strategy	Diff. [s.ms]
1	GA	SHH	-
2	Pirelli	SHM	+3.090
3	Pirelli	MHM	+3.220
4	GA	MH	+3.610
5	Pirelli	SHS	+3.810
6	Pirelli	SMM	+8.010

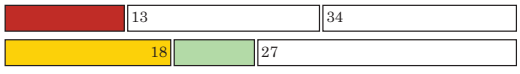

Figure 3: The GA strategies for Bahrain.

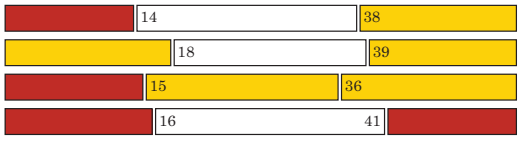

Figure 4: Pirelli strategies for Bahrain.

Barcelona. In the circuit of Barcelona, the GA was able to reproduce the strategies proposed by Pirelli and, in two cases out of three, it was even able to propose an improvement. The algorithm got the compounds right and basically found the best strategies: the fastest one is a 1-pit strategy with Medium and Hard, while the second fastest one is a 2-pit strategy with Soft and Medium compounds. In these two cases the GA proposed a little improvement w.r.t. the strategies proposed by Pirelli, which differs only for some laps in the pit-window. The strategies are shown in Fig. 5 and Fig. 6.

Table 5: Barcelona results.

Order	Output	Strategy	Diff. [s.ms]
1	GA	MH	-
2	GA	SSM	+0.120
3	Pirelli	SMS	+0.140
4	Pirelli	SH	+1.790
5	GA	SH	+2.450

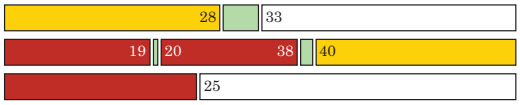

Figure 5: The GA strategies for Barcelona.

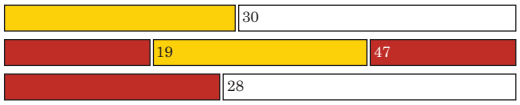

Figure 6: Pirelli strategies for Barcelona.

Monza. In the circuit of Monza the performances are very similar, since the fastest strategy is the 1-pit strategy with the usage of the Soft compound. The GA and Pirelli propose the same fastest strategy, while the second fastest one is predicted to be faster if the Soft compound is kept for some laps more than Pirelli suggested. The strategies are shown in Fig. 7 and Fig. 8.

Table 6: Monza results.

Order	Output	Strategy	Diff. [s.ms]
1	GA	SM	-
1	Pirelli	MS	+0.000
3	GA	SH	+3.460
4	Pirelli	SH	+4.810

Portimao. In the circuit of Portimao the GA performed very good: the proposals of the GA are almost identical to the Pirelli proposals. Indeed, the GA always proposed a pit-window that is shifted by just a few laps. Specifically, the algorithm proposes to use a little longer the first tire compound. The strategies are shown in Fig. 9 and Fig. 10.

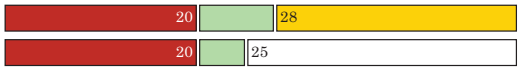

Figure 7: The GA strategies for Monza.

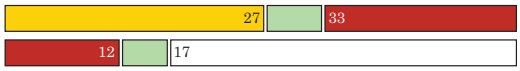

Figure 8: Pirelli strategies for Monza.

Table 7: Portimao results.

Order	Output	Strategy	Diff. [s.ms]
1	GA	SM	-
1	Pirelli	SM	+0.000
3	GA	SH	+2.840
4	Pirelli	MH	+2.890

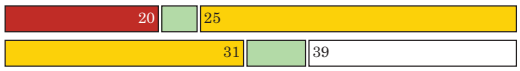

Figure 9: The GA strategies for Portimao.

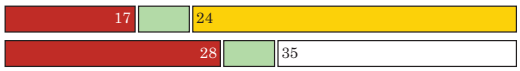

Figure 10: Pirelli strategies for Portimao.

Silverstone. In the circuit of Silverstone the GA finds the fastest strategy as the 1-pit strategy. There are also two proposals for the 2-pit strategy since Pirelli proposed one possible strategy with two pits. In particular, the strategies are almost identical between the GA and Pirelli proposals. The GA proposed also some intermediate strategies that can be worth to consider. However, the third strategy proposed by the GA, the Soft-Medium one, reaches the maximum wear life possible. While this strategy is in principle fine from a numerical perspective, in the practice it can yield a high risk of puncture. The strategies are shown in Fig. 11 and Fig. 12.

Table 8: Silverstone results.

Order	Output	Strategy	Diff. [s.ms]
1	GA	MH	-
1	Pirelli	MH	+0.000
3	GA	SH	+0.210
3	Pirelli	SH	+0.210
5	GA	SM	+1.260
6	GA	SSM	+12.080
7	GA	SMM	+12.110
7	Pirelli	SMM	+12.110

Spielberg. In the circuit of Spielberg the GA found the same strategies proposed by Pirelli. The only differences are, again, the laps of the pit-window. In the case of Pirelli, they proposed the pit-stops later than the GA, which leads to a little loss in terms of final race time. The strategies are shown in Fig. 13 and Fig. 14.

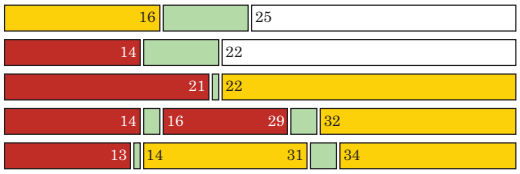

Figure 11: The GA strategies for Silverstone.

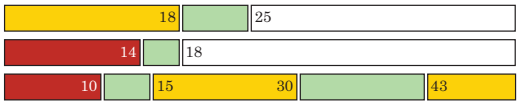

Figure 12: Pirelli strategies for Silverstone.

Table 9: Spielberg results.

Order	Output	Strategy	Diff. [s.ms]
1	GA	MH	-
2	Pirelli	MH	+0.020
3	GA	MHH	+1.760
4	Pirelli	MHH	+2.420
5	GA	SH	+3.870

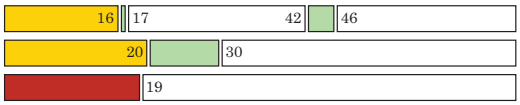

Figure 13: The GA strategies for Spielberg.

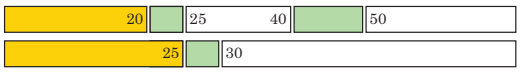

Figure 14: Pirelli strategies for Spielberg.

Zandvoort. In the circuit of Zandvoort the strategies proposed by the GA and Pirelli are very similar. In the case of a 1-pit strategy the compounds involved are the same and only the pit-window is a little different. For the 2-pit strategy the GA and Pirelli proposed a similar strategy, with Pirelli suggesting an earlier pit at each pit-stop. The fourth strategy proposed by the GA is included but, as for the case of Silverstone, while this strategy is numerically possible, realistically it can be very difficult to keep those tires for such an amount of laps. The strategies are shown in Fig. 15 and Fig. 16.

Table 10: Zandvoort results.

Order	Output	Strategy	Diff. [s.ms]
1	GA	SH	-
1	Pirelli	SH	+0.000
3	GA	MH	+2.300
3	Pirelli	MH	+2.300
5	GA	SSM	+3.860
5	Pirelli	SSM	+3.860
7	GA	SM	+4.950

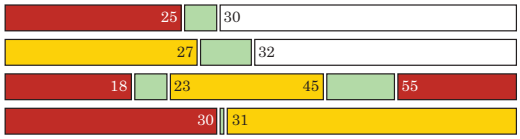

Figure 15: The GA strategies for Zandvoort.

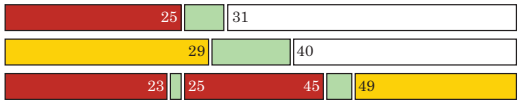

Figure 16: Pirelli strategies for Zandvoort.

5.3 Effect of the penalty

To evaluate the custom penalty proposed, a simple test was made in the case of the simulated data: the GA was run 10 times with the proposed dynamic penalty and 10 times with a ranking method based only on the fitness without any type of penalty. In both cases the selection has been applied on 1/7 of the population. As shown in Fig. 17, for the four circuits considered in this test, the mean of the final times with the dynamic penalty was better than the mean of the final times of the GA without dynamic penalty. Therefore we can conclude that the penalty is a crucial factor that effectively leads to better results.

Figure 17: Results with and without dynamic penalty.

5.4 Complexity

One of the key points of the proposed algorithm is the speed to compute a solution. Indeed, on a standard Linux PC (without multithreading) a single run of the GA makes between 1.5 and 2.5 iterations for each second in the case of the simulated data and it always reaches the minimum in less than 1 min. In the case of the real data, the GA reaches the minimum in about 5 s and it makes between 9 to 12 iterations per second.

In principle, the low computational time of the GA is also an advantage for running the algorithm in real time during a race. Even though the strategies are already fixed before, it could be useful to run the proposed algorithm during the race with live parameters. This can be the case, for example, of an unpredictable weather conditions where, at the beginning, the weather forecast is sunny, while during the race it starts raining. Moreover, the tire could have a different wear due to different track temperatures, wind, or other parameters that were wrongly estimated before the race.

6 CONCLUSIONS

In all the experiments conducted in this study the proposed GA was always able to find a very good strategy. In simulated data, it performed very well, with only a tiny margin of error, while with the real data it even found some faster strategies than Pirelli. In the other cases, the proposals were similar.

It should be noted that the proposed GA was designed to search for the *fastest* strategy under specific conditions, which may not necessary be the *best* strategy overall. What we mean is that, differently from the strategies proposed by Pirelli, which are somehow generic, the strategies proposed by our GA can be adapted to the driver performances and preferences as well as any other specificity about the car, the tires, the circuit, etc. Having more parameters to work with, the algorithm could return even more specific strategies. Moreover, it could be possible to extend the GA to handle also other types of mutation and crossover. Another improvement would be to introduce parallelization (the current implementation is single-population and single-threaded). Not only this would lower the compute time, but it would also allow migration schemes, to better explore the search space. Another possibility would be to consider more compact genotype representations.

To conclude, the proposed GA is a very good example of a real application of Evolutionary Computation in the world of car racing. Even if the GA has been tested on F1 data, it can be obviously applied to a variety of racing competitions, possibly even in real-time settings, thanks to the fact that the algorithm is very fast at computing an optimal strategy even with limited resources.

ACKNOWLEDGMENTS

We would like to thank Simone Berra and Fernando Osuna from Pirelli for providing the real data and giving us feedback on the numerical results.

REFERENCES

[1] Formula1 Dictionary. 2023. Race Strategy. https://www.formula1-dictionary.net/strategy_race.html
[2] Agoston E Eiben and James E Smith. 2015. *Introduction to evolutionary computing*. Springer.
[3] Ashref Maiza. 2020. Reinforcement Learning for Formula 1 Race Strategy. https://towardsdatascience.com/reinforcement-learning-for-formula-1-race-strategy-7f29c966472a
[4] Pirelli Motorsport. 2021. F1 Bahrain Possible Race Strategies. https://twitter.com/pirellisport/status/1376110079867559939
[5] Pirelli Motorsport. 2021. F1 Barcelona Possible Race Strategies. https://twitter.com/pirellisport/status/1391317096265654280
[6] Pirelli Motorsport. 2021. F1 Monza Possible Race Strategies. https://twitter.com/pirellisport/status/1436979044839473155
[7] Pirelli Motorsport. 2021. F1 Portimao Possible Race Strategies. https://twitter.com/pirellisport/status/1388780325292097536
[8] Pirelli Motorsport. 2021. F1 Spielberg Possible Race Strategies. https://twitter.com/pirellisport/status/1411618162244820999
[9] Pirelli Motorsport. 2021. F1 Zandvoort Possible Race Strategies. https://twitter.com/pirellisport/status/1434442437758435331
[10] The Royal Academy of Engineering. 2010. Formula One Race Strategy. *The Royal Academy of Engineering* 1 (2010), 7 pages.
[11] Stefano Ollanu. 2021. GP Gran Bretagna: passo gara, strategie gomme, meteo e TV. https://www.formulapassion.it/motorsport/formula-1/gp-gran-bretagna-2021-passo-gara-strategie-gomme-meteo-e-tv Redazione FormulaPassion.
[12] Diego Piccinotti. 2020. *Open Loop Planning for Formula 1 Race Strategy identification*. Master's thesis. Politecnico di Torino.
[13] Krzysztof Wloch and Peter J. Bentley. 2004. Optimising the Performance of a Formula One Car Using a Genetic Algorithm. In *Parallel Problem Solving from Nature - PPSN VIII*. Springer Berlin Heidelberg, Berlin, Heidelberg, 702–711.