IEEE Transactions on Intelligent Transportation Systems, Available online 07 May 2024
DOI: 10.1109/TITS.2024.3389155


# Formula-E multi-car race strategy development—A novel approach using reinforcement learning

Xuze Liu, Abbas Fotouhi, *Senior Member, IEEE*, Daniel Auger, *Senior Member, IEEE*

***Abstract*— Electric motorsport such as Formula E is becoming more and more popular in recent years. Race strategy in such races can be very complex involving resource management, e.g. energy and thermal management, but more importantly multi-car interactions which could be both collaborative and competitive. Reinforcement Learning has been implemented in the literature for such electric racing strategy development but only accounts for one single car. In this paper, we proposed a new architecture iRaXL to implement reinforcement learning for such complex strategy development featuring hybrid action space, multi-car interactions, and non-zero-sum gaming. The iRaXL proves to be able to develop different strategies for individual competitors and also team-based objectives. In a bigger scope, this framework can be used to solve more generic problems with hybrid features such as zero/non-zero-sum games, discretized/continuous action space, and competition/collaboration interactions.**

***Index Terms*— Motorsport, Race strategy, Reinforcement learning, Neural network.**

## I. INTRODUCTION

With the rapid growth in popularity of hybrid and electric vehicles over the recent years, higher-efficiency powertrain performance and better energy management strategy have always been among the top development priorities to satisfy the industrial demand for a higher driving range. In top-class motorsport series, more and more strict restrictions on energy/fuel consumption are introduced in the technical regulations to encourage higher-performance powertrain designs. Meanwhile, resource management is becoming a significantly important element in racing events. In the full-electric Formula-E (FE) world championship, two of the dominant elements in race strategy development are focused on energy management and battery thermal management which are proved by a large number of winnings and losses witnessed over the last seasons [1].

The majority of energy management-related research for road vehicles mainly focuses on developing advanced real-time controllers [2][3] to manage the energy flow of the vehicle. In contrast, the aim of energy management in motorsport is usually to extract the maximum performance out of a racecar within an energy/fuel constraint lap. Such resource management problems are usually referred to as lap time simulations (LTS) and are solved by using optimal control techniques. Limebeer et al. [4] studied the energy management strategy for an F1 hybrid system. For Formula-E races, Liu et al. [5] studied the energy management strategy under both energy and battery thermal constraints. Although optimal control techniques prove to be very reliable in solving such constrained problems, it is also a time-costly approach that takes minutes to hours to investigate a single lap and may take hours or even days for a typical race length (i.e. dozens of laps) depending on model fidelity. Therefore, it is more suitable for pre-race analysis but not favorable for making quick decisions during the race due to the highly dynamic and stochastic nature of motorsport.

For race-level strategy development, a race is usually discretized into laps [6] and lap performance is described using equations [7] that add different penalty terms to a baseline estimated time. Neural networks were first proposed in [8] as transition models that predict the car performance based on the state by the end of the previous lap and the strategic decision made upon it. Monte Carlo Tree Search (MCTS) was initially proposed in [8] to solve such multi-stage decision-making problems with sub-optimal solutions and then was improved in [16] by using reinforcement learning (RL) techniques. In the most recent published research, Liu et al. [9] adopted the deep deterministic policy gradient method to more realistically optimize the hybrid type of strategic actions (i.e. both discrete and continuous actions). This method efficiently generates much faster race strategy solutions compared to the MCTS methods which would potentially hide optimal solutions in the fully discrete action space.

Despite these methods proving to be powerful in strategy development, the literature still mostly focus on single-car race scenario. However, a real-life race usually comprises multiple teams and drivers. The interactions among the participants make the event highly dynamic. Moreover, different properties of the participants (e.g. vehicle efficiency, driver ability) could also affect a team's strategy choice. There is very little literature addressing multi-play race scenarios. Overtaking strategy was studied in [17] by formulating an optimal control problem for a single lap. On the race level, most literature focuses more on developing race simulators [10]. A limited number of recent research addressing decision-making on such race events are applying simple Monte Carlo methods for decision-making [11] or using supervised learning to learn from historical data [12]. The former method is usually implemented in real-time by using powerful computers. However, it could be very

All authors are with the Advanced Vehicle Engineering Centre, School of Aerospace, Transport and Manufacturing, Cranfield University, MK43 0AL, UK (e-mails: xuze.liu@cranfield.ac.uk; a.fotouhi@cranfield.ac.uk; d.j.auger@cranfield.ac.uk).

computationally expensive for a problem with very high dimensions and wouldn't be able to generate a timely solution due to insufficient Monte Carlo simulations. The utility of the latter supervised learning approach would be limited by the amount of historical data and lack of adjustability because motorsport is subject to constant development (e.g. cars, drivers, regulations) all the time. To overcome these weaknesses, in this study, we investigate using reinforcement learning as a new robust approach for this multi-car race strategy problem. Deep Reinforcement Learning (DRL) is a branch of machine learning algorithms that are designed to optimize decision-making and trajectories. RL-based methods have been proven of great utilities in a wide variety of applications such as robotics [18], games [19], resource management [20], etc. The DRL is theoretically very suitable for strategy development in race environments which mostly have position-based point-rewarding systems.

A motorsport race has several unique features. Firstly, it is usually a non-zero-sum game that involves more than two players. Traditionally in a gaming environment, DRL algorithms are used to solve zero-sum games that involve two competing parties [21] or to solve multiple-agent collaborative problems [22]. However, in FE, there are ten parties competing for awarded points. At the end of each race, the top ten drivers receive ranking-based championship points as shown in table C2 while non-top ten drivers receive nothing. Drivers are competing for higher points in each race in order to win the season championship title. Additionally, championships like FE are more than games of independent competing individuals. Teams, each normally with two cars, are also competing against each other for the team championship title (i.e. constructors' championship). Therefore, to maximize a team's benefit, a cooperative strategy might be needed between two cars from the same team under certain circumstances. Secondly, an FE race, like many from other categories, is a game with imperfect information. A team has total data access to its own cars while information of the other opponents is very limited. Opponents' battery status, which is a crucial element in the race strategy, is only occasionally observable through TV broadcasts. Good strategy development needs to make the best of this limited information. Last but not least, despite some participants that might have a slight advantage (e.g. vehicle efficiency, driver ability), considering human errors and interactions, there is hardly a winning-guaranteed strategy.

In this paper, we investigate the application of reinforcement learning techniques to the FE race strategy problem from two different perspectives: (1) Each driver competes independently and (2) A team simultaneously operates two cars to the team's benefit. The background is briefly introduced in this section. The proposed framework iRaXL would allow engineers to analyze each specific race independently of outmoded historical data. Meanwhile, strategy optimizations can be transferred from online to offline. The significant amount of time between race events breaks the online computational resource bottleneck and allows strategists to further investigate higher dimensional and more complex scenarios.

## II. Race environment

A typical FE race consists of a number of laps and all cars are initialized with the same amount of usable energy. Additionally, a battery temperature limit is shared among the cars which might be a significant constraint when a race is held in a warm climate. Some basic environmental information is shown in Table I.

TABLE I
BASIC ENVIRONMENT PARAMETERS

| Parameter | Value |
|---|---|
| Number of laps $N_{lap}$ | 12 |
| Total usable energy $E_{tot}$ | 20 kWh |
| Battery thermal limit $T_{limit}$ | 58 degC |
| Ambient temperature $T_{Amb}$ | 20-30 degC |
| Number of attack mode activations $N_{Att}$ | 1 |
| Number of laps per attack mode activation $N_{Attlap}$ | 2 |

The number of laps determines the end of a race and $E_{tot}$ is the upper limit of the energy consumed over the entire length of the race. $T_{limit}$ is the regulatory upper limit of the battery temperature exceeding which the performance will be significantly damaged. In this study, this is implemented as a constraint violating which the result be deemed as invalid. Ambien temperature is the environmental temperature at the place and time the race is held. It could significantly impact the cooling behavior of the battery and as a consequence, the margin to $T_{limit}$ and the strategic decision accordingly. The attack mode refers to a powertrain mode by activating which allows a car to have a higher power output from its powertrain. To activate such mode, a car would need to drive through a special zone, namely the activation zone, on the track which deviates from the normal race line. Therefore, a car driving in its normal mode, activation lap, and attack-activated lap should be considered as three different statuses [5]. Each team is given an identical maximum number of times to use such mode and it is mandatory for a team to use all of it.

After activation, such a higher power limit will sustain for a given amount of time and then return to the normal mode. In this study, due to a relatively short race, we set the $N_{Att}$ to one time and the sustain period to 2 laps per activation.

Similar to the race formulations in the literature, in this study, a race is discretized into lap steps. This section introduces the performance calculation within each environmental step along with other key elements of the environment used for RL training, including states, action space, and rewards.

### *A. Baseline performance calculation*

In this section, interactions among the race participants are not accounted for. The baseline performance metrics are calculated for an individual only.

Despite the race being discretized into laps which is more of a distance domain, time remains the key metric to determine the positions of cars in the environment. To gain a higher position in a race, as mentioned previously, a participant needs to make the best of their available energy to finish the race faster than

their opponents meanwhile maintaining the battery temperature within a boundary. Therefore, the performance calculation includes three key elements, namely lap time, energy consumption, and battery temperature rise.

Based on the race formulation in [9], to calculate the time a car performs in a lap, a baseline lap time is introduced as a function of three major strategic action components:

$$t_{lap,base} = f_{laptime}(a_{ELP}, a_{QM}, S_{PM}) \quad (1)$$

$a_{ELP}$ is referred to as "Energy per lap" which indicates how much energy a participant plans to use for a specific lap. $a_{QM}$ denotes the level of thermal management and $S_{PM}$ indicates which power mode the car is using. Details of the actions will be introduced later.

Given the baseline lap time, the energy consumption is also calculated based on a function of the three participant actions. The actual amount of energy consumption is calculated using equation:

$$E_{lap} = f_{energy}(a_{ELP}, a_{QM}, S_{PM}) \quad (2)$$

It should be noted that the base energy consumption is a function instead of directly inherited from $a_{ELP}$. This is because, in situations of high $a_{ELP}$ and $a_{QM}$ combinations, the planned energy is not fully consumed due to the stricter battery temperature management intention [8] (details previously discussed in [16]).

The battery temperature is relatively more complex as it can be affected by both internal factors (i.e. driver actions) and external factors such as ambient environment and battery characteristics. Therefore, the amount of battery temperature rise is calculated using a function:

$$T_{rise} = f_{temp}(E_{lap}, a_{QM}, S_{PM}, T_{Amb}, T_{Bat}, SOC) \quad (3)$$

where $T_{Bat}$ and $SOC$ are the battery temperature and state of charge at the beginning of a lap respectively. The three functions used in this study are three trained networks inherited from [16].

*B. State space*

In a typical race environment, the state space isn't fully observable. This is basically because a participant does not have access to its opponent's detailed information. In this study, the state information a participant gets from each environment step can be categorized into three groups.

The first group contains the state variables related to only a participant itself. These variables are shown in table II.

TABLE II
PARTICIPANT-SPECIFIC STATE VARIABLES

| Index | State variable | Definition |
|---|---|---|
| 1 | $T_{Amb}$ | Ambient temperature |
| 2 | $SOC$ | Battery state of charge |
| 3 | $T_{Bat}$ | Battery temperature |
| 4 | $t_{race}$ | Race time at the step |
| 5 | $S_{PM}$ | Current power mode |
| 6 | $S_{Att}$ | Remaining number of attack mode |
| 7 | $S_{Nattlap}$ | Remaining number of laps in current attack mode |

A participant has access to its own states such as time and battery information. $t_{race}$ is the total time accumulated lap time of each passed environment step starting from the beginning of a race. $S_{PM} \in [1,2,3]$ denotes the power mode the participant used in the last lap. $S_{Att}$ gives information on the number of available attack mode activations. And if the participant is currently using the attack mode. $S_{Nattlap}$ gives how many such higher power laps of this current activation remain.

In a multi-player race environment, a bigger group of observations is made of the information related to the other participants that can be constantly available through the official data stream, broadcasting, etc. These observations are potentially more valuable in tactics because they indicate the status of the opponents relative to which the participant is running in the race. Table III lists such state variables where $NP_{tot}$ is the total number of participants in a race.

TABLE III
FULLY OBSERVABLE STATES RELATED TO OTHER PARTICIPANTS

| Index | State variable | Definition |
|---|---|---|
| 8~8+($NP_{tot}$-1) | $DNF_{Pi}$ | DNF (did not finish) status $i$ |
| 9+($NP_{tot}$-1)~ 8+2*($NP_{tot}$-1) | $S_{Pi\_PM}$ | Participant $i$ current power mode |
| 9+2*($NP_{tot}$-1)~ 8+3*($NP_{tot}$-1) | $S_{Pi\_Att}$ | Participant $i$ remaining number of attack mode |
| 9+3*($NP_{tot}$-1)~ 8+4*($NP_{tot}$-1) | $S_{Pi\_Nattlap}$ | Participant $i$ remaining number of laps in current attack mode |
| 9+4*($NP_{tot}$-1)~ 8+5*($NP_{tot}$-1) | $dt_{Pi}$ | Time gap to participant $i$ |
| 9+5*($NP_{tot}$-1)~ 8+6*($NP_{tot}$-1) | $B\_TM_{Pi}$ | Boolean indicator if participant $i$ is a teammate |

Similar to Table II, the abilities and efficiencies of the participants are always accessible as they are participant-specific properties that normally do not change throughout the championships.

$DNF_{Pi}$ provides the information of whether or not the participant $i$ has already been out of the race due to either a battery issue or other incidents. Because the number of attack mode activations and the number of laps per activation are set by the regulations, and in real life, there are clear indicators of such states available to all viewers, state variables regarding the power mode and attack mode related status of other participants are always observable. $dt_{Pi}$ is the race time delta from the observer to the observee that also implies the position (i.e. ranking) of the observer on the track. The $B\_TM_{Pi}$ variable is $true$ if the observe (i.e. participant $i$) is the teammate of the observer otherwise it is $false$.

The final group of environment state variables are partially observable. This includes data that describes the detailed status of an opponent like those in the first group. Although such sensitive variables are not continuously observable, in the environment used in this study, additional state variables are introduced to describe such partially observable characteristics in order to guarantee the observation size is constant through the environment steps.

TABLE IV
PARTIALLY OBSERVABLE STATES RELATED TO OTHER PARTICIPANTS

| Index | State variable | Definition |
|---|---|---|
| 9+6*($NP_{tot}$-1)~ 8+7*($NP_{tot}$-1) | $SOC_{Pi}$ | Participant $i$ battery state of charge |
| 9+7*($NP_{tot}$-1)~ 8+8*($NP_{tot}$-1) | $SOC\_sGap_{Pi}$ | Number of passed steps since Participant $i$ battery SOC was last observable |
| 9+8*($NP_{tot}$-1)~ 8+9*($NP_{tot}$-1) | $TBat_{Pi}$ | Participant $i$ battery temperature |
| 9+9*($NP_{tot}$-1)~ 8+10*($NP_{tot}$-1) | $TBat\_sGap_{Pi}$ | Number of passed steps since Participant $i$ battery temperature was last observable |

To elaborate, the battery SOC and temperature information of other participants are not available to the observer under normal circumstances. They are observable only under two scenarios. The first one is when a specific participant is the teammate of the observer. In this case, the SOC and temperature are always available to the observer and the corresponding $sGap$s, therefore, are always zero. The second scenario is similar to what might happen in a real race. Although battery status is not directly accessible, such information is occasionally leaked by either TV broadcast or team radio. In this study, a probability $\epsilon_{leak}$ is introduced with which these sensitive states become available to the observer. The corresponding $sGap$s will be set to zeros at the leaking step and will increase step by step until another leak occurs.

Overall, in this study, a step observation returned from the environment comprises a total number of 10*$NP_{tot}$-2 state variables.

### C. Action space

According to [9], the action space for a single-car race is hybrid but relatively simple. It contains only one discrete action which decides whether to activate the attack mode or not and two continuous actions that are related to energy and thermal management as previously mentioned. However, when a race involves multiple cars, the action space becomes much larger because tactic actions are introduced. In this study, the hybrid actions are categorized into three groups.

The first group is a discrete action group that will be referred to as the tactics. Based on the aforementioned attack mode activation action, four extra tactic options are introduced. The first tactic option is "Chase" by choosing which the participant aims to shorten the distance from a car in the front. Similarly, a second tactic option of "Overtake" indicates the intention to overtake another participant. The third tactic is referred to as "Maintain" whose purpose is to keep the time gap from a front car by the end of a lap the same as it was at the beginning of the lap. One of the potential benefits of such a tactic is that energy consumption can be reduced by taking advantage of slipstreaming behind an upstream car. The final tactic option is "Free" which essentially means the participant is driving regardless of the other participants on the track. Overall, combining the four tactics with the attack mode activation option, this group has a categorical size of 8 as listed in table V.

TABLE V
TACTIC ACTION GROUP

| Index | Tactic description |
|---|---|
| 1 | Active attack mode and Chase |
| 2 | Active attack mode and Overtake |
| 3 | Active attack mode and Maintain |
| 4 | Active attack mode and Free |
| 5 | Do not active attack mode and Chase |
| 6 | Do not active attack mode and Overtake |
| 7 | Do not active attack mode and Maintain |
| 8 | Do not active attack mode and Free |

Given the tactic options, another essential group of discrete actions is required in a multi-car race environment to pinpoint which specific participant is the observer applying the tactic. Therefore, this second group simply contains $NP_{tot} - 1$ categorical options.

The third action group contains the continuous actions in the hybrid space. Table VI shows the action definitions and their ranges.

TABLE VI
CONTINUOUS ACTION GROUP

| Index | Symbol | Definition | Range |
|---|---|---|---|
| 1 | $a_{ELP}$ | Maximum energy a participant plan to use | 1.2~2.2 (kWh) |
| 2 | $a_{QM}$ | Level of thermal management | 0~3 |
| 3 | $a_{scalar}$ | Tactic scalar | -3~3 (s) |

The first continuous action $a_{ELP}$ represents how much energy a participant is willing to consume which is a crucial action for energy management. The range is 1.2~2.2 kWh is picked based on the Marrakech track used in this study. The action $a_{QM}$, as previously mentioned, determines the level of thermal management in a lap. $a_{QM}$ equal to 0 indicates that the battery temperature does not need to be taken care of, while $a_{QM}$ >0 means restricting target temperature rise by the magnitude of $0.95^{a_{QM}}$. For example, if $a_{QM}$=0 has a $T_{Bat}$ rise of 4 ℃, then for $a_{QM}$=1, $T_{Bat}$ will be expected to rise by 3.8 ℃. A tactic scalar $a_{scalar}$ is introduced to the continuous action group as a measurement to quantify the tactics. When using the Chase tactic, $a_{scalar}$ indicates what time gap by the end of the lap the observer wants to achieve behind its tactic participant. On the contrary, if Overtake tactic option is chosen, $a_{scalar}$ indicates how much the observer wants to be ahead of another participant after executing an overtaking manoeuvre. When Maintain or Free tactic is used, the scalar $a_{scalar}$ will not be effective in the environment step calculations.

### D. Performance calculation in multicar scenarios

Based on section II.A, in this section, we demonstrate how the performance calculation is modified to account for the interactions among the participants in a race.

**1) Minimum time gap**

In real life, cars are not mass points. There's a minimum gap between two cars by crossing which indicate two cars are colliding with each other. Therefore, a minimum time gap $t_{gap\_min}$ is introduced. In this study, at the end of each environment step, a recursive race time regularization is performed. If a car is detected to be followed too closely (i.e. $dt_{Pi} < t_{gap\_min}$) by a car at its back, a small time $t_\varepsilon$ will be added to the lap time $t_{lap,base}$ of the latter car to enforce the $dt_{Pi}$ to be $t_{gap\_min}$ at minimum. This mimics a real-life scenario where a driver aims to deliver a lap time based on the intended strategy which theoretically would end up in a position very close to the car in the front but got blocked. Therefore, instead of driving in a colliding manner, the car in the back needs to back off because the one in the front has a prior track position.

**2) Overtaking penalty**

To gain a higher position in a race, overtaking is inevitable. However, the feasibility depends on multiple factors. As pointed out in [17], overtaking feasibility varies based on conditions between two cars such as the initial gap at the beginning of a lap, power mode being used and intended energy consumption. The main assumption in the study was that the overtake had to be clean which means the car in the front takes no defensive manoeuvres and the car in the back needs to avoid driving aggressively into an exclusive zone of its opponent. In real life, although sometimes a clean overtake is not possible, a participant can still drive aggressively to battle with its opponent in order to make an overtake attempt. However, such manoeuvres usually bring an extra cost to energy consumption and inevitably with higher battery temperature rise. In this study, three judgements are introduced regarding overtaking actions based on the previous study. When an observer tries to overtake its teammate, no extra cost will be added to its step assuming its teammate is giving up track position collaboratively. When an observer is targeting its opponent, an extra amount of energy $E_\varepsilon$ will be added to the observer's lap energy if the initial gap between the two cars exceeds a threshold $f_{t_{\text{threshold}}}(a_{ELP_{ob}}, S_{PM_{ob}}, a_{ELP_{op}}, S_{PM_{op}})$ which is a function of intended energy consumption and power mode of both the observer its opponent. If added, the total energy consumption of the observer will become:

$$E_{lap} = f_{energy}(a_{ELP}, a_{QM}, S_{PM}) + E_\varepsilon \quad (4)$$

and its battery temperature rise will be calculated accordingly using equation (3). On contrary, if the initial time gap is less than the threshold, then the overtaking will be taken as a clean one hence no extra energy cost will be added.

It should be noted that this overtaking penalty also applies when a participant is on a Free tactic and able to overtake another participant due to the relatively slower pace of the latter one. The only scenario where the penalty will not be considered is when a participant overtakes a DNF car ahead of it.

**3) Slipstreaming effect**

Slipstreaming (or drafting) in motorsport refers to a technique of a downstream car driving close to an upstream car. In high-speed sectors of a track, a large amount of propulsion power is usually spent to overcome the aerodynamic drag. In those high-speed conditions, an upstream car would create a low-pressure region behind it. By driving in that region, the downstream car benefits from a reduction in drag [13] hence a significant reduction in energy consumption. In energy-crucial races, this technique becomes useful as it allows participants to save energy for future attacks. To account for such an effect, an energy bonus $E_{draft}(dt_{Pi})$ is introduced. Empirically, more energy can be saved when the downstream car maintains a closer gap to the upstream car. On the contrary, this effect will disappear if the gap is too big because the size of the low-pressure region is limited. Overall, if a participant chooses the Maintain tactic and the gap is close enough, a small amount of energy (i.e. $E_{draft}(dt_{Pi})$) will be subtracted as the following equation.

$$E_{lap} = f_{energy}(a_{ELP}, a_{QM}, S_{PM}) - E_{draft}(dt_{Pi}) \quad (5)$$

After calculating the three correction factors, at the end of each lap, a regulating process will be executed to force any intervals that are smaller than 0.1 seconds to 0.1 seconds by moving the following car backward. This effectively slows down the lap time of the following car however no energy will be returned to it. Such process is to mimic real-life environment where two cars cannot be infinitely close to each other.

Overall, these corrections are introduced to make the multi-car environment more realistic than in most literature where interactions among participants are neglected.

*E. Reward formulation*

In this study, the reward returned from an environmental step comprises several components as shown in the equation

$$\begin{aligned} R_{pi} = p_{action} + p_{DNF}(TBat_{Pi}, SOC_{Pi}, n_{lapremaining}) \\ + r_{rk}(Ranking_{Pi}, DNF_{pi}) \\ + r_{sm}(dt_{front}, DNF_{pi}) \end{aligned} \quad (6)$$

The first term is a behaviour modification penalty. A small amount of negative reward will be given to a participant when the intended action is infeasible. Such infeasible actions include but are not limited to (1) attempting to Overtake another participant far in the front; (2) maintaining distance with a car behind; (3) activating attack mode when no available activations are left. The aim of this penalty is to modify the participant's behaviour to avoid generating garbage actions. The second term is to penalize a participant for getting a DNF during the race due to either a flat or overheated battery. It is calculated by:

$$\begin{aligned} p_{DNF}(TBat_{Pi}, SOC_{Pi}, n_{lapremaining}) \\ = c1 * \min(0, SOC_{Pi}) + c2 \\ * \min(0, T_{limit} - TBat_{Pi}) + c3 \\ * n_{lapremaining} \end{aligned} \quad (7)$$

Where c1, c2 and c3 are three coefficients. Essentially, this penalty becomes active when $SOC_{Pi}$ drops below zero (i.e. flat battery) or battery temperature exceeds its regulatory limit $T_{limit}$. Furthermore, the third term penalizes more when a participant gets the DNF when there are still a number of laps remaining to finish in a race.

The third and fourth terms in equation (6) are the termination

reward. In the environment, a state is considered terminal when either of the two conditions is met: (1) all laps are finished; (2) all participants get DNF. The terminal reward $r_{rk}(Ranking_{Pi}, DNF_{pi})$ is a ranking-based reward. The higher position at which a participant successfully finishes the race, the higher the positive reward It will be given. Based on the ranking-based reward, a fourth term $r_{sm}(dt_{front}, DNF_{pi})$ is added as a smoothing term which is calculated by:

$$r_{sm}(dt_{front}, DNF_{pi}) = \max\left(\left(c4 - \log\left(c5 * dt_{front} + c6\right)\right), 0\right) * \left(r_{rk}(Ranking_{Pi} - 1) - r_{rk}(Ranking_{Pi})\right) \quad (8)$$

where $dt_{front}$ is the terminal time gap between the participant and the one in front of it, c4, c5 and c6 are three coefficients. Essentially, this means a participant is given a higher reward when it finishes more closely to the one ranking just in front of it, but not as much as the one in front gets. There are two main reasons this term is introduced. First, it has been pointed out in [9] that a non-smooth reward such as the ranking-based term is not favourable because a reinforcement learning network will struggle to approximate sharp changes in rewards. Second, the introduction of this term incents the participant to be closer to its opponent which could potentially lead to an overtake or other beneficial opportunities such as slipstreaming.

In general, the penalty terms are returned mostly in intermediate environment steps aiming to modify a participant's behaviour while the reward terms are returned in the terminal step which incents the participant to compete for higher positions.

## III. Reinforcement Learning Architecture

In this study, we develop a novel architecture called iRaXL based on Proximal Policy Optimization(PPO) technique to tackle the multi-car race strategy problem with hybrid action space. This section demonstrates the components in the architecture, modifications to the PPO algorithm and the neural network layouts for the PPO critic and actors. The PPO algorithm is chosen due to two main reasons. First, it is an on-policy that in general converges faster than off-policy algorithms [14][15]. Compared to the policy update process, the data collection in this study is a much more time-consuming process which would potentially cause a data shortage in a replay buffer which is normally used in off-policy approaches. With an off-policy agent constantly sampling from an impoverished replay buffer, the learning process could easily be unstable and make hyperparameter tuning tricky. The on-policy PPO allow the policy to be updated based on an adequate amount of the latest policy experiences. Furthermore, with the clipping feature in PPO, the policy update and convergence can be decently stabilized and requires less hyperparameter tuning than other on-policy algorithms. Second, PPO is arguably one of the most popular model-free RL algorithms that have been extensively developed. In reality, a race could be extremely complicated because the opponents' strategies in the environment are highly changeable as it evolves. This makes model-learning for a race environment very infeasible. Therefore, a model-free RL algorithm would suit such complex scenarios better.

### A. Network layouts

The PPO algorithm generally requires two networks, namely an actor network and a critic network. The actor is used for experience collection (i.e. interaction with the environment) and the critic is more used in the network parameter update process. In this study, due to the complexity of the hybrid action space (hybrid action type and action dependencies), the actor network is modified with three sub-networks which evaluates sequentially. The general layout is demonstrated in Fig. **1**.

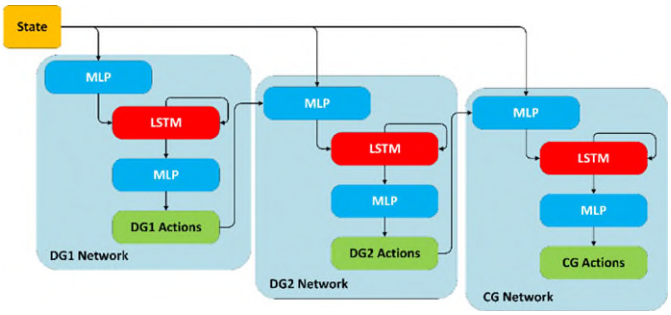



**Fig. 1** PPO actor network layout

The DG1 network is the top sub-network that is responsible for the discrete tactic actions. Using observations from the environment, it generates corresponding categorical outputs as mentioned in table V. Due to the dependency of the second action group on the tactic actions as mentioned in section II.C, the input of DG2 network comprises the observation from the environment and the action output from DG1 network. The output of DG2 is the second group of discrete actions which determines to which participant it applies the tactic as introduced previously. Similarly, the CG network combines all the actions from DG1 and DG2 networks with the observation from the environment and then generates the final continuous action group. In comparison, the critic network is relatively simpler as it only uses the observations to compute advantage estimates.

Each network has its own Long Short-Term Memory (LSTM) kernel. The LSTM kernel is integrated to capture more hidden information in the observations. Technically, a race strategy development is a partially observable Markov decision process. In real-life races, an observation sequence potentially reveals the strategy a participant opts for. Therefore, using an LSTM kernel has a natural advantage over a flat network that only makes use of a single observation. The input dimensions of these three kernels are different. The DG1 kernel input is the MLP processed sequence of states. For DG2 network, the DG1 action output is firstly repeated to the same sequence size of the states and then concatenated with the state sequence. The concatenated sequence is into the DG2 input MLP whose output becomes the input to the DG2 LSTM kernel. Similarly, the CG network takes the concatenation of DG2 action and state sequence as the network input.

The modified PPO update for these networks is demonstrated in the next section.

*B. PPO implementation*

In general, PPO algorithm aims to maximize the objective of:

$$L^{CLIP}(\theta) = c_1 \hat{E}_t\big[\min\big(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\big] + c_2 ETP[\pi_\theta](s_t) \quad (1)$$

where $r_t(\theta)$ denotes the probability ratio $\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}\right)$, $\epsilon$ is a hyperparameter of clipping ratio, $ETP[\pi_\theta](s_t)$ is an entropy term to encourage actor exploration, $c_1$ and $c_2$ are two weighting factors. When the probability ratio between new policy and old policy, ie. $r_t(\theta)$, is potentially outside the interval of $[1-\epsilon, 1+\epsilon]$, the ratio is clipped to such bounds during the objective calculation followed by the gradient calculations. The purpose such clipping is to ensure the new policy does not deviate too much from the old policy which is likely to destabilize the training. $\hat{A}_t$ is an estimator of the advantage function at timestep $t$. For a length-T trajectory, $\hat{A}_t$ can be calculated by:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \quad (2)$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, $\gamma$ is the discount factor and $\lambda$ is the generalized advantage estimation (GAE) factor. For the PPO critic network, the aim is to minimize the prediction error which is a simple squared-error loss $L_t^{VF} = (V_{\theta^{VF}}(s_t) - V_t^{targ})^2$.

PPO policy update algorithm suits well to both continuous action space and discrete categorical action space. In this study, the PPO algorithm is used for parameter updates for all the sub-networks in the actor and the critic network. The update process can be described in the following pseudocode.

**Algorithm** PPO networks update

Initialize PPO actor and critic network parameters $\theta^{\mu\_DG1}$, $\theta^{\mu\_DG2}$, $\theta^{\mu\_CG}$, $\theta^{VF}$
For PPO iteration=1, M do
Repeat episode with policy $\pi_{\theta^{\mu\_DG1}old}$, $\pi_{\theta^{\mu\_DG2}old}$, $\pi_{\theta^{\mu\_CG}old}$
Store sequences $[S_{set}, (a_{DG1}, a_{DG2}, a_{CG}), S'_{set}, \mathrm{R}]$
Until a total number of $T$ sequences are collected
Compute
Compute advantage estimates $\hat{A}_1, \cdots, \hat{A}_T$ for the sequences
for epoch = 1, K do
Sample a random minibatch of N transitions $(S_{set}, (a_{DG1}, a_{DG2}, a_{CG}), S'_{set}, \mathrm{R})$
Update critic network parameters $\theta^{VF}$ using $L_t^{VF}$
end for
for epoch = 1, K do
Sample a random minibatch of N transitions $(S_{set}, a_{DG1}, S'_{set}, \mathrm{R})$
Update actor sub-network DG1 parameters $\theta^{\mu\_DG1}$ using $c_1 L^{CLIP}(\theta^{\mu\_DG1}) + c_2 ETP[\pi_{\theta^{\mu\_DG1}}](s_t)$
end for
for epoch = 1, K do
Sample a random minibatch of N transitions $(S_{set}, (a_{DG1}, a_{DG2}), S'_{set}, \mathrm{R})$
Update actor sub-network DG2 parameters $\theta^{\mu\_DG2}$ using $c_1 L^{CLIP}(\theta^{\mu\_DG2}) + c_2 ETP[\pi_{\theta^{\mu\_DG2}}](s_t, a_{DG1})$
end for
for epoch = 1, K do
Sample a random minibatch of N transitions $(S_{set}, (a_{DG1}, a_{DG2}, a_{CG}), S'_{set}, \mathrm{R})$
Update actor sub-network CG parameters $\theta^{\mu\_CG}$ using $c_1 L^{CLIP}(\theta^{\mu\_CG}) + c_2 ETP[\pi_{\theta^{\mu\_CG}}](s_t, a_{DG1}, a_{DG2})$
end for
end for

*C. The iRaXL architecture*

The iRaXL architecture has three major components, a worker cluster, a main learner and a policy pool. The architecture is illustrated in Fig. **2**.

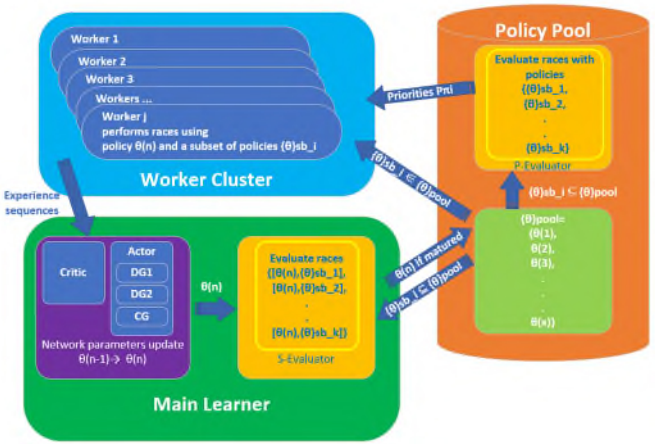



**Fig. 2.** iRaXL architecture

All parameter updates for the critic and actor networks in the PPO algorithm happen within the main learner. It receives experience sequences from the worker cluster and uses them to update the networks. Additionally, the main learner has an internal sub-evaluator (referred to as S-Evaluator). After each parameter update, the new network parameters are sent to the workers to perform another on-policy experience-collecting process. Meanwhile, the S-Evaluator begins to evaluate the updated policy by using it to play against other saved policies in the policy pool. Once the S-Evaluator assesses the updated policy to be matured, the matured policy will be added to the policy pool. The criterion for deciding if a policy is matured also accounts for the performance of a total number $N_{pre}$ of its predecessors in their evaluations. This is to ensure the policy has stable and consistent performance rather than a peculiar that appears to be strong. The latest policy will be assessed as matured only if it satisfies all the following criteria: (1) The latest policy and its predecessors must not have DNF results from all their evaluations; (2) The average team finishing position must be higher than a pre-set threshold $Ranking_{Ave\,Threshold}$; (3) The iteration number of the latest policy must be greater than the last matured policy iteration number by a value of $N_{pre}$. In this study, the third point is added as a non-result-oriented criterion. The PPO algorithm is

designed to guarantee policies do not diverge too much from one iteration to another, therefore, improving overall convergence. Because the S-Evaluator in this framework does not evaluate the policies in a full-factorial manner and the adjacent policies can be very similar due to the nature of PPO. It is very likely that redundant matured policies are collected into the pool. Therefore, this third criterion is added to guarantee each collected policy is distinguished from its predecessor.

The policy pool is a shared storage between the worker cluster and the main learner. Additionally, another major responsibility of the pool is to evaluate all the matured policies and provide priority information to its users (i.e. worker cluster and main learner). The policy evaluator (referred to as P-Evaluator) is constantly playing episodes of races by using randomly picked policies from the pool with random race start positions. After a number of evaluations, the P-Evaluator calculates the priorities $P$ of every policy $\pi_i$ in the pool based on the evaluation results using the equation below.

$$P_{\pi_i} = e^{w_{ranking}\frac{\sum Ranking_i}{n_i} - w_{DNF}\frac{n_{DNF_i}}{n_i}} \quad (3)$$

where $n_i$ is the total number of races that policy $\pi_i$ has participated, $n_{DNF_i}$ is the number of times $\pi_i$ has failed to complete a race, $w_{ranking}$ and $w_{DNF}$ are two weighting factors. Basically, a policy will be assigned a higher priority when it has less DNF rate and higher finishing rankings.

The worker cluster has access to the policies in the pool with their priorities and receives the latest policy directly from the main learner. As illustrated in Fig. **2**, when workers perform races, the latest policy will be assigned to two random race participants (i.e. a team) in each race. For the other participants, a worker will choose policies from the pool with a probability proportional to its corresponding priority. After a race terminates, the $(s, a, r, s')$ i.e. observation-action-reward-new observation sequences will be sent to the main learner and used for policy updates.

Overall, the iRaXL architecture guarantees a stable learning process for such multi-player zero-sum environments. The PPO algorithm provides a foundation for stable learning with clip ratios. And the LSTM kernels enhance information capturing capabilities. Additionally, the pool provides a mixture of gradually matured policies all with a positive priority. This allows the workers to collect experience against smarter and smarter policies. More importantly, it ensures that the relatively less intelligent policies are still involved to avoid "forgetting" behaviour during the learning process. The outcome of this architecture is demonstrated in the next section.

## IV. Result and Discussion

In this study, we trained the policy in two different scenarios. In the first one, each participant competes individually whereas in the second one participants are paired into teams. After approximately 4 days of computing time of training, we collected a total of 29 matured policies in the Single scenario and 28 matured policies in the Team scenario. It should be noted the shown results are collected from the pool evaluations of these policies at the end instead of the training process.

### *A. Individual scenario result*

Fig. **1** shows four of the major metrics in the environment. It can be observed in Fig. **1**(c) and (d) that in the early learning phase (first 5 versions), the agent is mainly learning to use as many resources as possible. This is understandable because more energy consumption and less thermal management empirically bring faster lap time hence overall faster race time. This can be reflected in Fig. **1**(b) where the average race finishing time was reduced. From version 25 the race time starts to converge to a fast level. It should be noted that despite the learning agent is no longer able to find significant faster race time solutions due to the fundamental resource limit, new policies are still being developed, evaluated as matured and put into the pool. The higher average finishing position shows the later policies have relatively stronger performances within the pool. To a certain degree, this phenomenon reveals the underneath learning process where the learning agent's focus has transferred from utilizing the resource to exploring different tactics. This can be jointly proved by the actions taken by these collected policies shown in Fig. **2**.

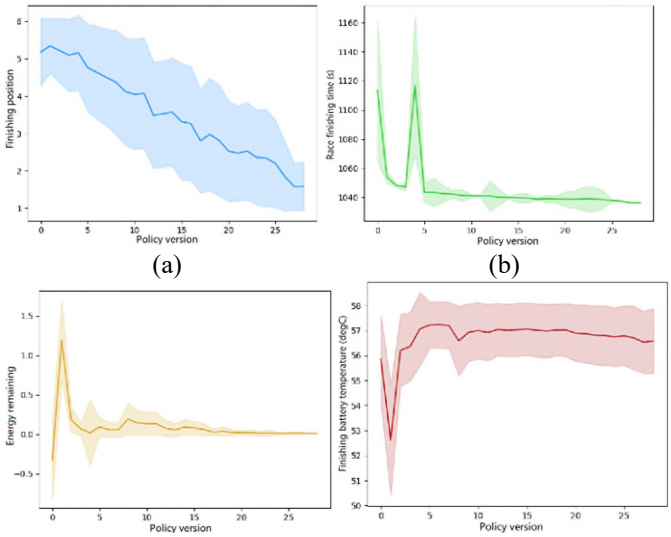
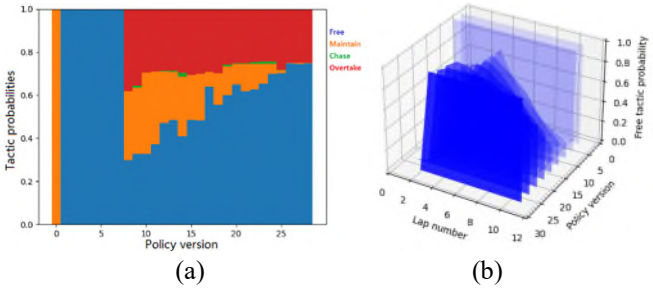
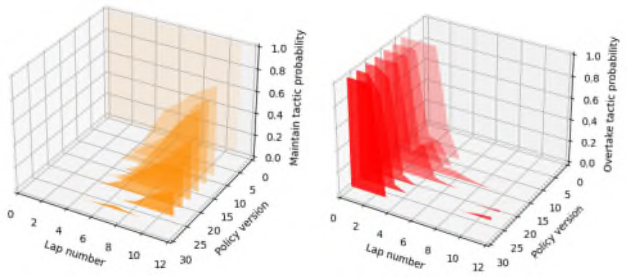



**Fig. 1.** (a) Average finishing position; (b) Average Race time; (c) Average remaining energy at the end of races; (d) Average battery temperature at the end of races. The shades in the figures represent the minimum and maximum metric values performed by the policy in the evaluations.

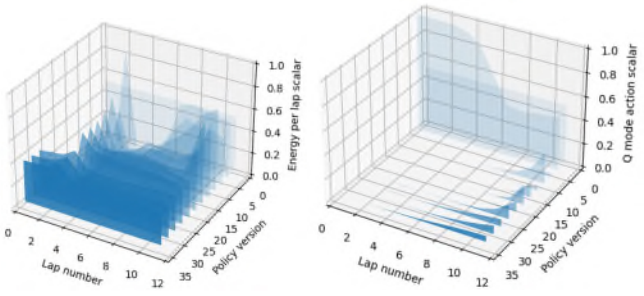





(c) (d)

**Fig. 2.** Probabilities of each tactic being used (a) Overall summary; (b) Free tactic; (c) Maintain; (d) Overtake

The 'Free' tactic option dominates the early policies. This is of less interest because the early versions are relative 'garbage'. From the 8th version, the policy started to have variety in its tactic actions. 'Free', 'Maintain' and 'Overtake' are the major chosen options with 'Chase' occupying a small percentage. As the training progresses, the 'Maintain' portion becomes smaller. After approximately the 26th version, the 'Overtake' and 'Free' are the major actions. Fig. **2**(b,c,d) demonstrates the statistics of where these tactics are adopted. It can be observed that the majority of 'Free' tactics are taken in the middle phase of races. The 'Maintain' tactics firstly emerged in the finishing phase and later became less adopted in later versions of policies. On contrary, the 'Overtake' tactics are mostly observed at the beginning of races. Similarly, the evolvement of the continuous actions is shown in Fig. **3**.



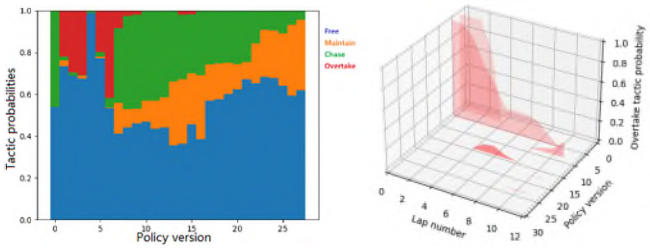

(a) (b)

**Fig. 3.** Average continuous action scalars (a) Energy per lap; (b) Thermal management Q mode.

It can be seen from the result that with the policy evolving, the energy strategy becomes more evenly distributed. The later policies suggest more energy should be spent in the beginning phase of a race and more thermal management (Q mode action) need to be taken care of in the finishing phase which agrees with the findings in the previous study [9]. To a certain extent, this explains why the 'Overtake' tactic is more favourable in the beginning. That is because empirically, overtaking is an expensive action in terms of energy consumption and battery thermal management. In a temperature-limited race, overtaking the final laps will increase the battery temperature more than doing it in the beginning laps. Therefore, the policy evolves in a direction where overtaking is preferable in the early phase of races.

*B. Team scenario result*

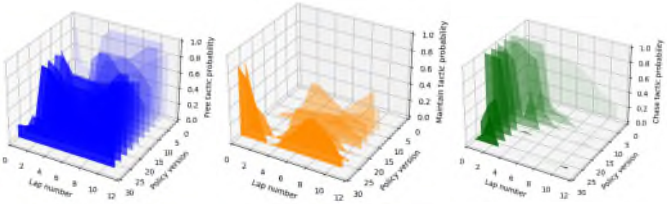
Because the general trends of the four metrics developed in the Team scenario are very similar to those in the Individual scenario therefore they are not discussed here. However, the observed tactics are very different when participants are paired into teams as shown in Fig. **4**.




(a) (b)

**Fig. 4.** Probabilities of each tactic being used (a) Overall summary; (b) Overtake

A very significant difference lies within the Overtake tactic which is very popular in the Individual scenario. In the Team scenario, the Overtake tactic was first developed and adopted mostly in the beginning phase of a race similar to that in the Individual scenario. However, it soon got abandoned and replaced by a mixture of Maintain and Chase tactics which are interestingly not preferable in the Individual scenario.



(a) (b) (c)

**Fig. 5.** Probabilities of each tactic being used (a) Free; (b) Maintain; (c) Chase

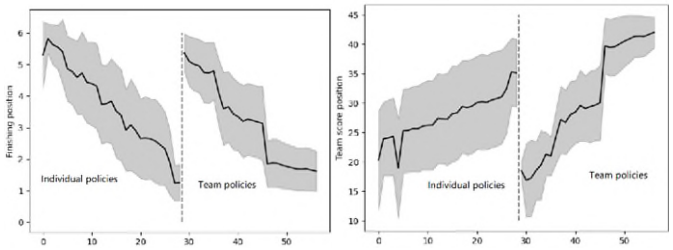
In general, the Free tactic is less dominant averaging 60% of the tactic actions compared to around 70% in the Individual scenario. The Maintain occupies the second largest portion. Although the Maintain tactic was also observed in the Individual scenario, the timing of such tactic has changed significantly from mostly in the second half of a race to a separated two chunks during a race comparing Fig. **2**(c) and Fig. **5**(b).

It is also worth noticing in the latest matured policies, that the portion of Chase reduces and is mostly taken over by Maintain during the beginning phase of a race. Although these two tactics have some overlapped functionalities, the result shows that the Maintain tactic is probably more favourable when the opponents become stronger. The energy and thermal strategies share the same patterns as those in the Individual scenario, therefore, are not further discussed here. But it should be noted that the non-Free tactics dominating the beginning phase of a race agree with the Individual scenario. This also supports the conclusion that if any special tactic is needed, it is preverbally to be done in the early laps of a race where the energy and thermal penalties are potentially the smallest.

### C. Team versus Individual result

The previous sections showed that the tactics developed from different scenarios could be very different. In this section, we mix all the policies from both scenarios and evaluate them in team-based races. It should be noted in this case, that although Individual policies are paired into teams, they still observe each other as opponents.

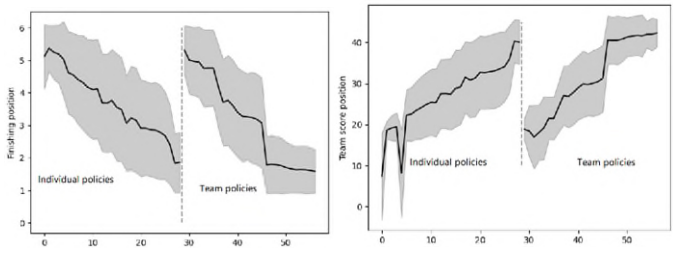
While little difference was observed in the tactics distributions, a more interesting result is shown from the race outcomes. Fig. **6** shows the information of positions and awarded points by the end of the evaluated races. The shades in the figures represent the minimum and maximum metric values performed by the policy in the evaluations.




(a) (b)

**Fig. 6.** Average race result (a) Finishing positions; (b) Collected team scores

In terms of finishing positions, the latest Individual policy has a very strong performance averaging close to first place in the races it participated in. On the Team policy side, during the four days of training, this scenario has developed many more versions of policies that deliver similar performance but none of them performs as strong as the latest Individual policy. However, in terms of collected team scores, the Team policies have a much more dominant performance despite those slight weaker individual contributions. This is understandable because when Individual policies are paired into teams, a stronger policy is very likely to be dragged by a weaker teammate. To eliminate this effect, we completed further evaluations in which we explicitly forced team-up individual policies to be the same version. The result is shown in Fig. **7**.




(a) (b)

**Fig. 7.** Average race result (a) Finishing positions; (b) Collected team scores

While the finishing information shares very similar patterns to the previous result, the relative dominance has changed. It can be observed by comparing Fig. **6**(b) and Fig. **7**(b) that when a strong Individual policy is given an equally strong teammate, the collected team score improves significantly yet is still not as good as Team policies. A dramatic flip also happened in the finishing positions. Previously the latest Individual policy has stronger performance than overall Team policies. However, when its teammate becomes equally strong, the average finishing position dropped. The reason is believed to be that a strong teammate was introduced along with an extra inevitable internal competition between the teammates. As a result, in this case, the Team policies not only have an advantage in the team score but also outperformed the Individual policies in terms of finishing positions.

This change in the result reflects a typical dilemma in real life. When a team has two almost equally strong drivers, there's a balance between prioritizing the benefit of a team or a specific car. This will lead to different strategies being chosen and different race results. This is not within the scope of this study and therefore is not discussed.

## V. Conclusion

In this paper, reinforcement learning is implemented in race strategy development. We proposed a new architecture iRaXL to accommodate unique features of a race such as hybrid action space, non-zero-sum game, multiple players, including both collaboration and competition, etc.

The policies have been trained for two different objectives, namely individual and team rewards. It has been observed that different tactics were developed for these two different scenarios. From the mixed evaluations, Team policies always outperformed the Individual policies in terms of team benefit. The Individual ones, although can be strong, have their limitations. First, the Individual ones can only produce strong (although not as strong as Team ones) team performance when two cars on the same team are similarly competitive. Second, when one seeks higher individual finishing positions, a strong Individual must not be paired with similar policies. Otherwise, the competition between the teammates will compromise individual outcomes.

The iRaXL architecture proves to be able to develop race strategies. However, it should be noted that there is no universal strategy that could serve both individual and team benefits. Therefore iRaXL should be implemented with a clear vision of the target outcome.

## Acknowledgment

The research data from the project will be available at https://doi.org/10.17862/cranfield.rd.23642844; it is subject to an embargo due to the terms of a confidentiality agreement.

## References

[1] "FIA Formula E." Energy Management 101: The Importance of Energy in Formula E | FIA Formula E, 18 Sept. 2020, www.fiaformulae.com/en/news/2020/march/formula-e-energy-management.

[2] Wieczorek, Maciej, and Mirosław Lewandowski. "A mathematical representation of an energy management strategy for hybrid energy storage sssystem in electric vehicle and real time optimization using a genetic algorithm." Applied energy 192 (2017): 222-233.

[3] Montazeri, Morteza, Abbas Fotouhi, and Akbar Naderpour. "Driving segment simulation for determination of the most effective driving features for HEV intelligent control." Vehicle system dynamics 50.2 (2012): 229-246.

[4] Limebeer, David JN, Giacomo Perantoni, and Anil V. Rao. "Optimal control of formula one car energy recovery systems." International Journal of Control 87.10 (2014): 2065-2080.
[5] Liu, Xuze, Abbas Fotouhi, and Daniel J. Auger. "Optimal energy management for formula-E cars with regulatory limits and thermal constraints." Applied Energy 279 (2020): 115805.
[6] Bekker, James, and W. Lotz. "Planning Formula One race strategies using discrete-event simulation." Journal of the Operational Research Society 60.7 (2009): 952-961.
[7] Heilmeier, Alexander, Michael Graf, and Markus Lienkamp. "A Race Simulation for Strategy Decisions in Circuit Motorsports." 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018.
[8] Liu, Xuze, and Abbas Fotouhi. "Formula-E race strategy development using artificial neural networks and Monte Carlo tree search." Neural Computing and Applications (2020): 1-17.
[9] Liu, Xuze, Abbas Fotouhi, and Daniel J. Auger. "Formula-E race strategy development using distributed policy gradient reinforcement learning." Knowledge-Based Systems 216 (2021): 106781.
[10] Choo, Christopher Ledesma Weisen. Real-time decision making in motorsports: analytics for improving professional car race strategy. Diss. Massachusetts Institute of Technology, 2015.
[11] Heilmeier, Alexander, et al. " Application of Monte Carlo Methods to Consider Probabilistic Effects in a Race Simulation for Circuit Motorsport." Applied Sciences 10.12 (2020): 4229.
[12] Heilmeier, Alexander, et al. "Virtual Strategy Engineer: Using Artificial Neural Networks for Making Race Strategy Decisions in Circuit Motorsport." Applied Sciences 10.21 (2020): 7805.
[13] Newbon, Joshua, David Sims-Williams, and Robert Dominy. "Aerodynamic analysis of Grand Prix cars operating in wake flows." SAE International journal of passenger cars. Mechanical systems. 10.1 (2017): 318-329.
[14] Lapan, Maxim. Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more. Packt Publishing Ltd, 2018.
[15] Labao, Alfonso B., Mygel Andrei M. Martija, and Prospero C. Naval. "A3C-GS: Adaptive Moment Gradient Sharing With Locks for Asynchronous Actor-Critic Agents." IEEE Transactions on Neural Networks and Learning Systems (2020).
[16] Liu, Xuze, Abbas Fotouhi, and Daniel Auger. "Application of advanced tree search and proximal policy optimization on formula-E race strategy development." Expert Systems with Applications 197 (2022): 116718.
[17] Liu, Xuze, Abbas Fotouhi, and Daniel J. Auger. "Energy-optimal overtaking manoeuvres of Formula-E cars." Vehicle System Dynamics (2022): 1-28.
[18] Kober, Jens, J. Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey." The International Journal of Robotics Research 32.11 (2013): 1238-1274.Aa
[19] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).
[20] Mao, Hongzi, et al. "Resource management with deep reinforcement learning." Proceedings of the 15th ACM workshop on hot topics in networks. 2016.
[21] Silver, David, et al. "Mastering the game of go without human knowledge." nature 550.7676 (2017): 354-359.
[22] Feng, Jun, et al. "Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning." Proceedings of the 2018 World Wide Web Conference. 2018.

**Xuze Liu** received his BSc and MEng degrees from Beijing Institute of Technology (BIT), Beijing, China, in 2015 and 2017 respectively and received his MSc from Cranfield University, UK in 2018. He completed this Ph.D. degree within Advanced Vehicle Engineering Centre, School of Aerospace, Transport and Manufacturing, Cranfield University in 2022.

**Dr Abbas Fotouhi** is Senior Lecturer in Advanced Vehicle Engineering Centre at Cranfield University. He has more than fifteen years research experience in dynamical systems modelling, simulation, optimization, and control. He has also extensive practical and algorithmic experience of applying Artificial Intelligence and Machine Learning techniques in engineering problems. Before joining Cranfield, he was with the Centre for Artificial Intelligence and Robotics (CAIRO) at University Technology Malaysia. His current research is focused on vehicular systems including hybrid and electric vehicle powertrain systems, energy storage technologies and autonomous cars.

**Professor Daniel Auger** is Professor in Electrification, Automation and Control in Cranfield University's Advanced Vehicle Engineering Centre. He is an expert in control systems, vehicle electrification and autonomy. He leads major work streams on self-driving cars, and he has pioneered advanced battery management algorithms for lightweight lithium-sulfur batteries. He has also introduced new state-of-the-art MSc teaching in applied control engineering and launched a new course in connected and autonomous vehicle engineering.

# Formula-E multi-car race strategy development—a novel approach using reinforcement learning

Liu, Xuze

2024-08-01



Liu X, Fotouhi A, Auger D. (2024) Formula-E multi-car race strategy development—a novel approach using reinforcement learning. IEEE Transactions on Intelligent Transportation Systems. Volume 25, Issue 8, August 2024, pp. 9524-9534

https://doi.org/10.1109/TITS.2024.3389155