Expert Systems With Applications 197 (2022) 116718

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Application of advanced tree search and proximal policy optimization on formula-E race strategy development

Xuze Liu *, Abbas Fotouhi, Daniel Auger

Advanced Vehicle Engineering Centre, School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, Bedfordshire MK43 0AL, UK

ARTICLE INFO

Keywords:
Energy management
Formula-E race strategy
Monte Carlo Tree search
Proximal policy optimization

ABSTRACT

Energy and thermal management is a crucial element in Formula-E race strategy development. Most published literature focuses on the optimal management strategy for a single lap and results in sub-optimal solutions to the larger multi-lap problem. In this study, two Monte Carlo Tree Search (MCTS) enhancement techniques are proposed for multi-lap Formula-E racing strategy development. It is shown that using the bivariate Gaussian distribution enhancement, race finishing time improves by at least 0.25% and its variance reduces by more than 26%. Compared to the published conventional MCTS technique used in multi-lap problems, this proposed technique is proved to bring a remarkable enhancement with no additional computational time cost. By further enhancing the MCTS using proximal policy optimization, the final product is capable of generating more than 0.5% quicker race time solutions and improving the consistency by over 90% which makes it a very suitable method particularly when enough training time is guaranteed

1. Introduction

In recent years, electric vehicles have become more and more popular with powertrain and battery technologies developing rapidly. Energy management has been one of the most popular topics on electric cars, both hybrid and full electric. Top-level motorsport series have continuously introduced stricter boundaries for energy consumptions to encourage more high-efficiency powertrain technology development. Specifically, in Formula One (F1), the total fuel usage during a full race is limited along with restricted usage of the electric energy in the hybrid system (FIA, 2021a). In World Endurance Championships (WEC), the concept is similar but refuelling during a race is allowed (FIA, 2021c) due to the much longer duration of the race. While race strategy development for these two series also needs to account for other factors such as tire management, in Formula–E (FE) championships, the race strategy is more concentrated on powertrain, specifically the energy and thermal management.

The technical regulation (FIA, 2021b) of FE states that for a complete race, the total amount of energy that can be delivered from the Rechargeable Energy Storage System (RESS) to the Motor Generator Unit (MGU) is limited to 52 kWh. The maximum power is limited to 200 kW in race mode settings. Beyond that, teams are given options of activating attack mode for a certain amount of time which gives an extra power of 50 kW during this power mode. In addition to the energy-wise restrictions by the regulation, thermal management is another major concern when races are held in hot climates such as those present in Marrakesh and Santiago. Teams have to avoid battery overheating which leads to de-rated power and potentially a Did Not Finish (DNF). Heat is generated both during propulsion (vital for speed) and regeneration (vital for energy efficiency and endurance). In general, the problem of FE race strategy development can be described as making decisions for each race such as how much energy to use for a lap (i.e. Energy per lap) and choice of power mode; the objective is the quickest race time, and the problem must be solved within present energy and thermal constraints.

The energy management problems in published research fall into two main categories: (1) real-time applications, which are the majority (Peng, He, & Xiong, 2017; Wang, Huang, Khajepour, & Song, 2016; Wieczorek & Lewandowski, 2017; Zhang & Xiong, 2015); and (2) trip-oriented off-line optimizations for a pre-specified route (Brayshaw & Harrison, 2005; Zhang & Vahidi, 2012; Du, Zhao, Wang, Zhang, & Xia, 2016; Gong, Li, & Peng, 2008). In the first category, various control strategies have been used to manage the power flow among multiple energy resources. These researches targeted to achieve the maximum overall powertrain efficiency at each timeframe or a certain optimization horizon given a specific power demand from the diver input. In the

* Corresponding author.
E-mail address: xuze.liu@cranfield.ac.uk (X. Liu).

https://doi.org/10.1016/j.eswa.2022.116718
Received 14 April 2021; Received in revised form 6 January 2022; Accepted 21 February 2022
Available online 25 February 2022
0957-4174/© 2022 Elsevier Ltd. All rights reserved.

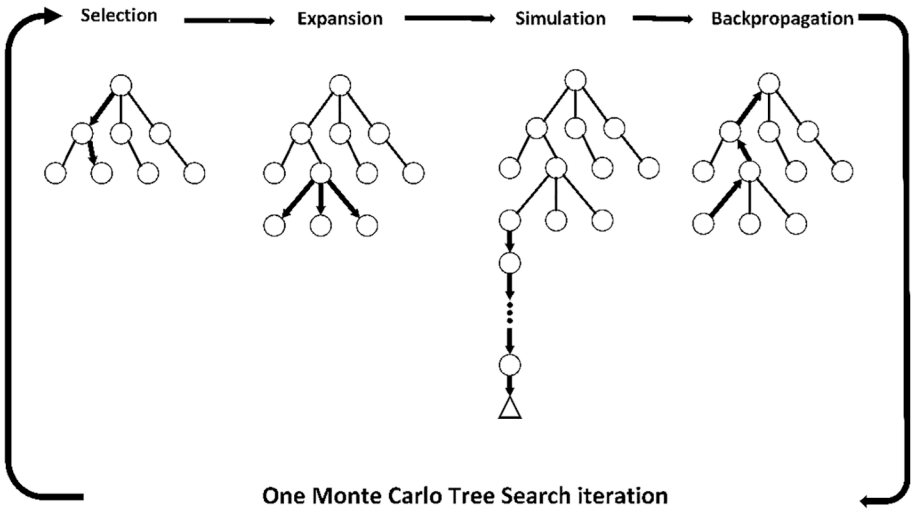


Fig. 1. Monte Carlo Tree Search iteration.

second category, the total energy consumption along the route is optimized based on the information of the route, vehicle and powertrain. In motorsport applications, the energy is usually treated as a constraint and the target is to achieve minimal lap time which is computed by lap time simulations (LTS). For LTSs, quasi-steady-state (QSS) technique (Brayshaw & Harrison, 2005; Heilmeier, Geisslinger, & Betz, 2019) is a widely used method which has the advantage of less computational time (typically taking seconds). However, the weakness of QSS methods is that computational efficiency comes at the expense of reduced complexity of the internal models, which may mean that the optimal solutions to the approximated problem vary significantly from the optimal solution to the true real-world problem. Also, the short optimization windows of QSS techniques make it very difficult to generate an optimal management problem solution for an entire lap (Heilmeier et al., 2019). To overcome this, the energy management problem is usually formulated as an optimal control problem (OCP), where the internal model can be detailed and more complex objectives can be applied. For example, Giacomo, et al. (Limebeer & Perantoni, 2015; Perantoni & Limebeer, 2015) used an optimal control technique to model a track in a 3D ribbon way with banking and elevation features and studied the effects of aero-suspension interaction. Vehicle parameter optimizations such as differential settings are studied in (Perantoni & Limebeer, 2014; Tremlett et al., 2015). In terms of management problems, Tremlett et al. (Tremlett & Limebeer, 2016) optimized the tire usage by including a thermal-dynamic model in the OCP. Limebeer et al. (Limebeer, Perantoni, & Rao, 2014) studied the energy management strategy for an F1 hybrid system.

Optimal control methods perform very reliably for solving minimum time manoeuvre problems with various constraints. However, compared to QSS methods, optimal control has a high time cost (dozens of minutes for a single lap calculation). To be useful to race engineers, a 'strategic tool' requires special features such as: (1) have prediction models that predict performances with acceptable accuracy; (2) be able to perform a long optimization horizon or deep depth (e.g. length of a race, multiple laps) of decision makings; and (3) be able to generate a solution quickly. Solving multi-lap management problems using optimal control methods would easily cost hours of computational time, not to mention for the full length of a race. The third point – speed – is crucial because motorsport is a highly dynamic environment and teams have to be able to make fast decisions in reaction to unexpected changes during a race (e.g. environmental or race conditions)

There are very few publications targeting such strategy problems. Most of these focus on building race simulation discussing how to discretize a race and what influence factors to include. Bekker (Bekker & Lotz, 2009) discretized a race into sectors of approximately 150 m in length. The time for each sector was calculated by penalties representing adding air resistance and fuel load to a baseline time. (Sulsters & Bekker, 2018) and (Choo, 2015) both discretized a race into laps. While Christopher mainly focused on studying historical data patterns, Claudia used more specific formulas to calculate lap times, adding tire age effect and random variability. Meanwhile in (Choo, 2015), overtaking bonuses and penalties were accounted for with fixed numbers and the probability of DNFs is introduced. Alexander, et al. (Heilmeier, Graf, & Lienkamp, 2018) introduced a more detailed discretization in which based on a lap level discretization, a pitstop is described into sub-sectors to capture overtake opportunities during pitstops. Additionally, Alexander introduced more factors such as driver interaction and car/driver abilities into lap time calculations. In terms of race strategy decision making, (Liu & Fotouhi, 2020) is the latest publication focusing on a relevant problem in which a race is discretized into laps and Monte Carlo Tree Search (MCTS) was proposed as the decision-making algorithm.

Race strategy development for FE competitions features several challenges to tackle: (1) a big action space comprising many choices for driving style and power mode; (2) the decision problem is a long-sequential in nature (due to the number of laps in a race); (3) Decisions made in the early stages have profound effects in later stages (e.g. decisions on energy consumption and battery temperature); (4) During a race, conditions may change; so one has to properly adapt to the changes.

MCTS has several fundamental concepts which make it a very suitable algorithm for such race strategic planning application: (1) It progressively builds a partial tree instead of a full game tree which is way beyond the realm of computation; (2) It allows tuning of the dilemma between exploration and exploitation; (3) It takes reward only from the terminal state (race end) which teams ultimately care about; (4) The quality of an action is approximated through random simulations; (5) The quality values can be used to adjust the searching policy toward more promising solutions.

As proposed in (Liu & Fotouhi, 2020), MCTS managed to tackle the race strategy development problem. And it has been proved that MCTS can generate a decent solution for both pre-race planning and in-race condition changing scenarios. However, it has also been pointed that the solution given by MCTS is sub-optimal which isn't favourable for motorsport where a one-second difference in solution will hugely affect the race outcome. The reason for such a result is that the MCTS used in (Liu & Fotouhi, 2020) is a very ordinary type without further tailoring for this application. In this study, the aim is to apply different modifications to the solution proposed in (Liu & Fotouhi, 2020) in order to

Table 1
Symbol definition.

Symbol	Description
S	A set of states where s_0 and s_T are the initial and terminal state respectively
A(s)	Action space under the state s
T(s,a, s')	Transition model which describes the transition from state s to s' when action a is taken
Q(s)	Reward function
$\pi_d(a\|s)$	Probability of action a being taken under state s

Table 2
State description

State variable	Description
N_{lap}	Remaining number of laps
E_r	Remaining usable energy
T_{amb}	Ambient temperature
N_{att}	Available number of Attack Mode(AM) activation
$N_{Rattlap}$	Current remaining number of AM laps
T_{bat}	Battery temperature

enhance its performance.

MCTS is an iterative tree searching algorithm where each iteration comprises four steps as follows (shown in Fig. 1):

1) Selection: The agent starts from the root node, moves to the child with the highest priority and progressively descend through the branches until reaching an expandable leaf node. The priority is defined by the tree policy. A node is expandable if it is neither a terminal state nor its action space is null.
2) Expansion: A node is expanded by adding child nodes under it based on its action space.
3) Simulation: A simulation, governed by the simulation policy, is performed from the newly added node to a terminal state and a reward is calculated.
4) Backpropagation: The simulation reward is backpropagated through the branches toward the root node and the information of the passed node is updated.

The enhancement techniques of MCTS proposed in the literature mainly focused on the four steps of MCTS iteration. For the selection step, the most popular selection algorithm is the Upper Confidence Bounds for Trees (UCT) proposed by (Kocsis, Szepesvári, & Willemson, 2006) stating that the agent follows the child with the highest UCT value given by:

$$UCT = \frac{Q(v')}{N(v')} + c\sqrt{\frac{2lnN(v)}{N(v')}} \tag{1}$$

where v denotes the root node and v' is the node where the agent locates. $Q(v')$ is the sum of simulation reward under node v' and N denotes the visit count of a node. c is a balancing parameter that balances the exploration and exploitation of the search. The majority of enhancement on the selection step is to ensure higher reliability of a node. This includes replacing the second term (Auer, Cesa-Bianchi, & Fischer, 2002; van den Broeck, Driessens, & Ramon, 2009) or adding a third term (CHASLOT, WINANDS, HERIK, UITERWIJK, & BOUZY, 2008; Schadd, Winands, van den Herik, Chaslot, & Uiterwijk, 2008) to the equation to assess the variance and reliability of the rewards.

For the expansion step, there is no specific enhancement method. One thing is the decision to use single child node expansion per step or to add multiple child nodes per step. This depends largely on the computational budget.

The simulation step is another area of improvement as stated in the literature. The default simulation policy is to randomly choose an action from the action space. This is a very general starting point that no domain-specific knowledge is required. However, the main issue that this policy will lead to is the aforementioned high level of uncertainty or variance in the simulation rewards which might compromise the selection step. And the randomness cannot guarantee a realistic approximation of the quality of action because some poor actions may be taken equally likely as the promising ones. There are two main categories of simulation enhancement. First, simulation can be governed or evaluated. Silver (Silver & Tesauro, 2009) proposed ruled based simulation which includes domain-specific knowledge in the simulation sequence. Rimmel (Rimmel & Teytaud, 2010) introduced a domain-independent contextual concept into the simulation which uses statistics from previous simulations to guide the future ones. The second category includes those whose simulation actions are evaluated. Winands, et al. (Winands & Björnsson, 2010) used the evaluation function to avoid bad moves in the simulation. Some literature combined learning features such as Temporal Difference Learning (TDL) (Silver, Sutton, & Müller, 2008) in the simulation step. Actions are taken based on their values (Q) and the Q-tables are updated during each backpropagation step (Finnsson & Björnsson, 2010; Finnsson, 2007).

The enhancement for the backpropagation step mainly aims to favour the selection step by updating additional information such as those in the aforementioned selection equations. Other modifications on backpropagation mainly focus on weighting different simulation results. Xie, et al. (Xie & Liu, 2009) allocate heavier weight to the later simulation results which are believed to be more accurate than earlier performed ones. Decaying reward is another method to weigh earlier wins to later wins by multiplying a factor $0 < \gamma < 1$ recursively while the agent backpropagates from leaf node to root node.

A big breakthrough in recent years is the success of AlphaGo-Zero (Silver et al., 2017) which used a variant of MCTS with an Actor-Critic (A2C)-like-frame to enhance self-learning performance. It introduced a prior probability value $P(v')$ to the selection equation $Q(v') + \frac{P(v')}{1+N(v')}$ and used the policy head of A2C to guide the simulation. One of the most popular A2C like algorithms recently is the Proximal policy optimization (PPO) [Proximal Policy Optimization Algorithms]. Previous policy gradient methods may easily suffer from their sensitivities to hyperparameters such as learning rate, update period, etc. hence poor convergence performance. The introduction of a clipped objective PPO policy update ensures the deviation from the previous policy is relatively small. This makes PPO a very stable and easy-to-implement method in finding an optimal policy.

Based on the previous weakness pointed out in (Liu & Fotouhi, 2020), this study aims to improve the quality of the FE race strategy solution by investigating different methods to enhance the tree search. The first section introduces the problem background and previous researches. The second section demonstrates the formulation of the strategy development problem. In the third section, different enhancement methods used in this study are presented. The results and discussion are demonstrated in section 4. Finally, the conclusion is presented in section 5.

2. Problem formulation

The decision making for race strategy is a sequential decision-making problem whose environment is fully observable. To formulate such a problem in this paper, some symbols are defined as stated in Table 1.

2.1. States and actions

A state s contains the information accessible during a race including variables that have a significant influence on decision making. Such variables which are contained in a state are stated in Table 2.

In this study, the Marrakesh ePrix track is used. For a 45-minute race, the total number of laps is 34 thus the N_{lap} is 34 at most. The N_{lap} will

Table 3
Action space

PM	EPL(kWh)	QM
1	1.2,1.3,…,1.9,2.0	0,1,2,3
2	1.2,1.3,…,2.0,2.1	0,1,2,3
3	1.2,1.3,…,2.1,2.2	0,1,2,3

later determine the depth of searching. The FE technical regulation states that the amount of energy that can be delivered to the MGUs by the RESS is limited to 52 kWh. Therefore E_r is 52kWh at most. The ambient temperature is another important element that affects the cooling of the battery. Overheating the battery (T_{amb} reaching above an upper limit) will lead to a DNF or other forms of unfavourable results hence must be avoided. Both E_r and T_{bat} will be accounted for in the reward function. N_{att} gives how many times a team can activate attack mode during a race. Each activation lasts a number of laps and $N_{Rattlap}$ gives how many attack mode laps remains during one activation. N_{att} and $N_{Rattlap}$ will mainly affect the size of the action space.

In a previous study presented in (Liu & Fotouhi, 2020), the action space has been defined by the combinations of four inputs, namely the drive power, regeneration power, lift and coasting (LaC) distance and LaC torque. They have their different impacts on energy consumption, lap time and battery temperature changes. In this study, these are replaced by three more representative parameters which are more closed to real-life applications (explained below). The reason for choosing these more macroscopic parameters is that on such race (multiple laps) strategy level, capturing what changes lap after lap is more important than understanding what happens during a single lap.

The first parameter is called Energy per Lap (EPL). With total usable energy being a limited resource during a race, this is a very directly energy-related parameter teams use to decide how much energy to use for a certain number of laps. The second parameter is defined to indicate which power mode (PM) to be used. For example, normal race mode (PM = 1) and attack mode (PM = 3) are the choices whose upper limits for driving power are 200 kW and 250 kW respectively. Additionally, considering that the attack mode is activated at a certain point in the middle of a lap instead of at the start/finish line, and the extra 50 kW of power are only allowed afterwards, a new power mode (PM = 2, attack mode activation lap) is introduced between those two modes to describe the performance in the activation lap. The third action parameter is called the Heat Generation Mode which later in this paper will be referred to as Q-mode (QM). More specifically, with the same amount of available energy, different settings of thermal boundaries may compromise the lap time but offer a more efficient way to manage the temperature rise compared to brutally decreasing the ELP. In this study, QM has four options from 0 to 3. QM of 0 indicates no temperature constraint is applied while the following choices from 1 to 3 each restricts the temperature rise to 95% of the former one (i.e. if QM0 has a battery T_{bat} rise of 4 ℃, then for QM1 T_{bat} will rise by 3.8 ℃). To summarize, the complete action space is made of 120 actions as shown in Table 3.

In this study of race strategy development, the action space varies as the race goes on. For example, if PM = 2 (attack mode activation lap) is chosen at a certain lap, the action space for the next lap will only contain actions under PM = 3. The action space following a PM = 1 lap will either contain actions for both PM = 1 and 2 or only for PM = 1 when there's no available attack mode left. Therefore, as the MCTS agent descends into different layers, the action space changes. However, it should be noted that the action space doesn't shrink gradually in the later phases like in board games or delivery problems where there will be fewer choices as the problem progresses. The vast searching space would make the problem difficult to solve which will be addressed later.

2.2. Transition model

The transition model here in this study should provide information on the effect of an action on the lap time, battery temperature and energy consumption during a single lap. In previous research (Liu & Fotouhi, 2020), neural networks were proposed to be trained as transition models providing decent accuracy. While a commercial simulation software was used to generate training data in (Liu & Fotouhi, 2020), in this study, the training datasets are collected by formulating each case into an OCP. The reason for doing so is to guarantee the optimality of lap time performance of a given input while commercial simulation software failed to do so with their empirical driver model. An OCP is formulated to minimize a Lagrange cost function of

$$J = \int_{t_0}^{t_f} l(t, x(t), u(t), p)dt \tag{2}$$

which is subject to the constraints of

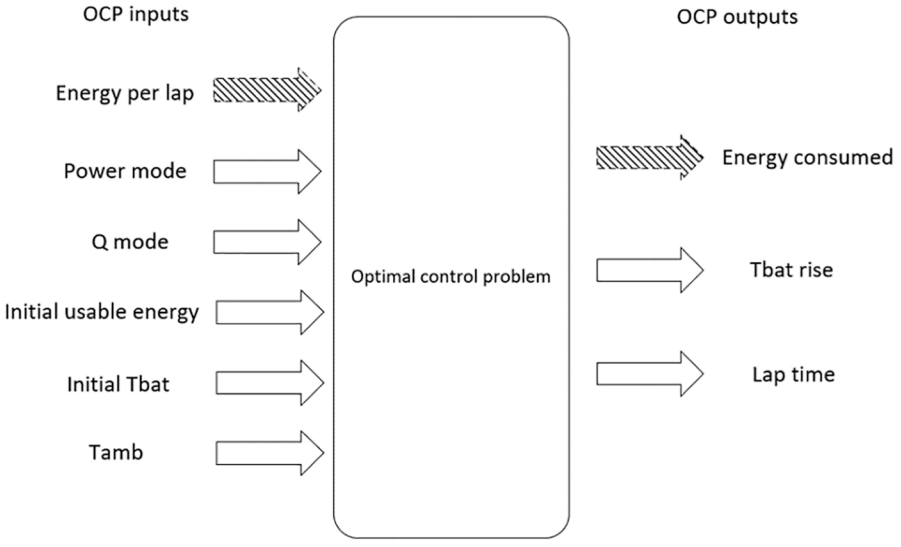


Fig. 2. Data collection process through the OCP.

Table 4
Input variables' range

Input	Range
Energy per lap(kWh)	1.2 ~ 2.2
Power mode	1,2,3
Q mode	0,1,2,3
Initial usable energy(kWh)	0 ~ 52
Initial battery temperature(℃)	20 ~ 60
Ambient temperature(℃)	15 ~ 40

$$\begin{cases} \dfrac{dx}{dt} - f(t, x(t), u(t)) = 0 \\ g(t, x(t), u(t)) = 0 \\ h(t, x(t), u(t)) \le 0 \\ g_b(x(t_0), x(t_f), u(t_0), u(t_f)) = 0 \end{cases} \tag{3}$$

In this problem, $x(t) \in R^n$ is the state vector made of vehicle dynamics information and u$(t) \in R^m$ is the control vector of steering and paddles. The system dynamics is described in the vector $f(t, x(t), u(t)) \in R^n$. Vector $g \in R^{n_g}$ and $g_b \in R^{n_{gb}}$ are the quality constraints and boundary constraints. The inequality constraints are defined in $h \in R^{n_h}$ where the aforementioned EPL and QM are included.

This approach has been thoroughly explained in (Liu, Fotouhi, & Auger, 2020) therefore won't be detailed in this paper. The data collection process through the OCP is shown in Fig. 2.

The initial usable energy represents the state of charge (SOC) of the battery which influences the thermal dynamics model inside the problem. It should be noted that the energy element appears on both input and output sides as the filled arrow showed in Fig. 2. The reason for doing so will be explained later. The input ranges for data collection are defined in Table 4. The input variables are picked randomly from their range. A total number of 129,000 datasets are collected to train the neural network transition model. An example of the collected data is shown in Fig. 3 including the effects of Q mode and EPL on the lap time. The blocked area in the figure explains why 'energy' has to be on both the input and output sides. That is because, in situations of high EPL and QM combinations, the requested energy isn't fully consumed due to the strict thermal boundaries. Therefore, there will be a nonlinear mapping between each side and the transition model have to be able to capture this feature.

As a result, three individual networks are trained to model the energy consumption, battery temperature rise and lap time separately. This procedure is inherited from (Liu & Fotouhi, 2020) thus is not demonstrated here. The structure of the networks and accuracies are shown in appendix A.

2.3. Tree structure and reward function

As previously introduced, a Monte Carlo tree is made of a root node, branches and layers of nodes. In this study, the root node is defined as the initial state of an event. Typically for a full race at Marrakesh, this means the root node state vector (defined in section 2.1) is assigned as:

$$\left[N_{lap}, E_r, T_{bat}, T_{amb}, N_{att}, N_{Rattlap}\right] = [34, 52, 20, 20, 2, 2] \tag{4}$$

The race is discretized into laps. Each time the searching agent descends into a deeper layer, it means that the race progresses into another lap. The neural network transition model acts when the branches connecting a parent node to its child nodes updates the state information of the child nodes. In the simulation step performed from a leaf node, the transition model is also used to simulate from the leaf node state to the terminal state that is when N_{lap} becomes zero. The simulation reward is backpropagated from the leaf node all the way to the root node in the backpropagation step. The reward is defined as:

$$Q(s_T) = \begin{cases} 3000 - t_T, if\, T_{bat} < 60 and E_r > 0 \\ 0, else \end{cases} \tag{5}$$

which means that a successful finish with faster time t_T , gets a higher reward and a DNF (due to either overheated battery or over-consumed energy) gets a reward of zero. In this way, the MCTS is expected to generate a successful fast race finishing strategy solution

3. Enhancement methods OF MCTS

In this section, different methods of MCTS enhancement are demonstrated. This includes applying UCT enhancement techniques from the literature, a new simulation and expansion method using bivariate Gaussian distribution (BGD) proposed by the authors and a reinforcement learning method for further enhancing both expansion and simulation steps.

3.1. UCT modification

The most commonly used UCT algorithm is proposed by Kocsis and Szepesvári (Kocsis et al., 2006).

$$UCT_AveR = \frac{Q(v')}{N(v')} + c\sqrt{\frac{2lnN(v)}{N(v')}} \tag{6}$$

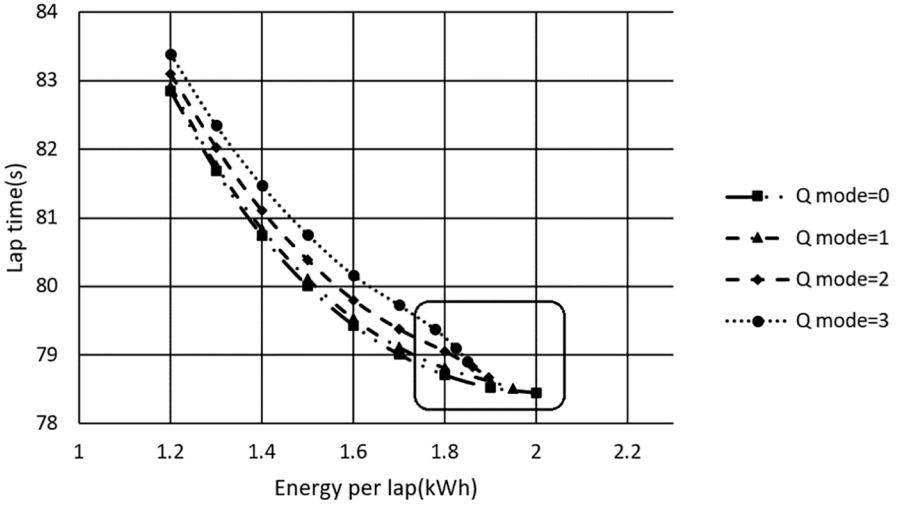


Fig. 3. Lap time of different EPL and Q modes (Power mode = 1).

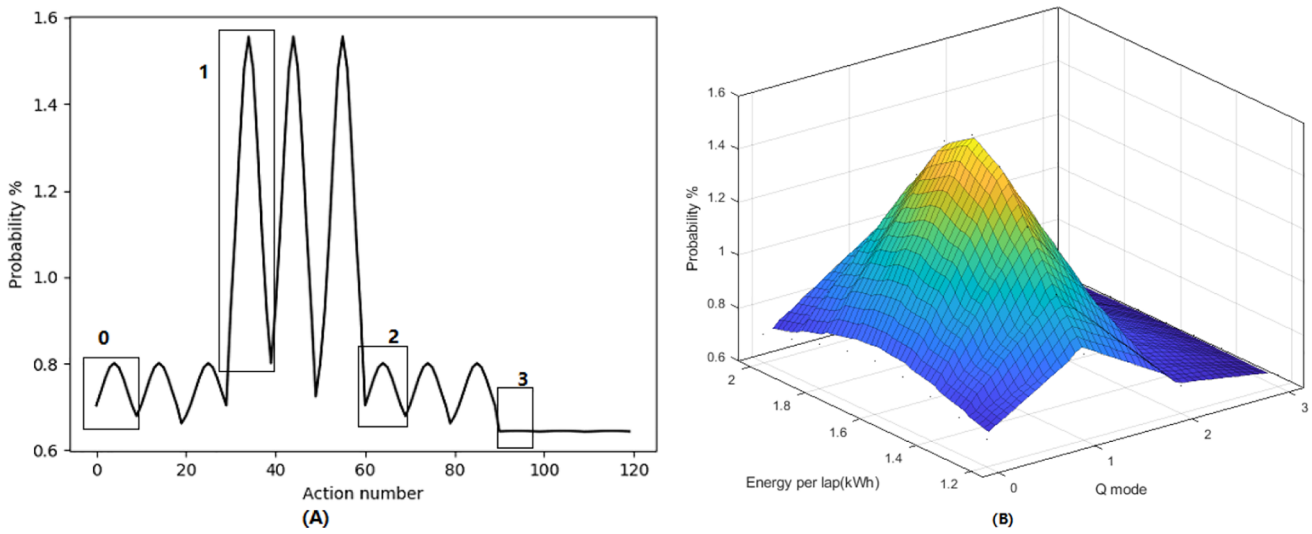


Fig. 4. Example of an action probability distribution A) Probability distribution over action space B) Enlarged distribution of Power mode 1 actions.

Table 5
Abbreviations of methods

No.	Abbreviation	Selection	Simulation	Expansion(Max number)
1	UCT_AR	Eq. (6)	Random	Full(-)
2	UCT_MR	Eq. (7)	Random	Full(-)
3	UCB1_T	Eq. (9)	Random	Full(-)
4	UCT_SP	Eq. (10)	Random	Full(-)
5	UCT_AR_Gau	Eq. (6)	Guided by BGD	Full(-)
6	UCT_MR_Gau	Eq. (7)	Guided by BGD	Full(-)
7	UCB1_T_Gau	Eq. (9)	Guided by BGD	Full(-)
8	UCT_SP_Gau	Eq. (10)	Guided by BGD	Full(-)
9	Gau_Gau	Eq. (7)	Guided by BGD	Guided by BGD(10)
10	PPO	Eq. (7)	Guided by PPO actor	Guided by PPO actor (10)

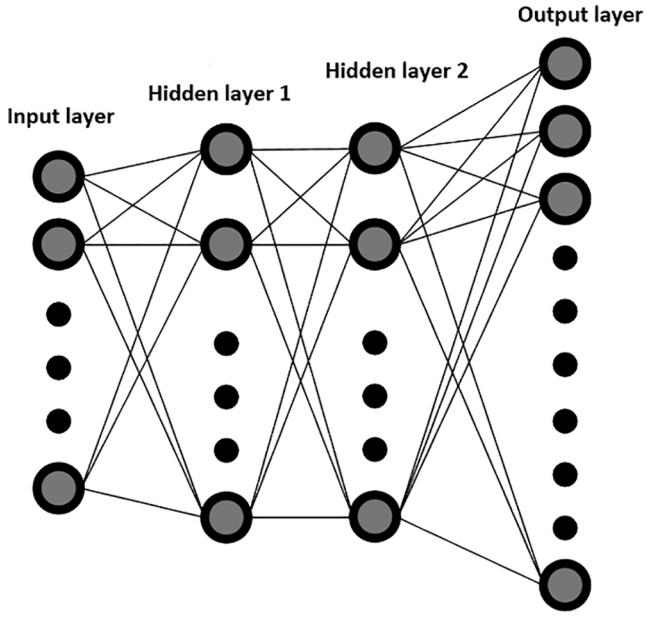


Fig. 5. The policy network layout.

The second term of the formula is called the exploration term. The first term is called the exploitation term which is the average reward gained at node $v^{'}$. In dual player games or zero-sum game theory, this

Table 6
Initial state

State parameter	Value
N_{lap}	34
E_r (kWh)	52
T_{bat} (℃)	20
T_{amb} (℃)	25
N_{att}	2
$N_{Rattlap}$	2

usually means the UCT value leads the agent toward higher winning odds. However, in this race strategy problem, the aim is to have the fastest finishing time instead of the odds of faster times. This average reward exploitation term may diverge the agent from the optimal solution because the fastest time can easily be compromised by its weak siblings. So, to tailor the selection algorithm in this study, the first term is replaced with the highest reward ever propagated through the node $v^{'}$. That is:

$$UCT_MaxR = Qmax(v') + c\sqrt{\frac{2lnN(v)}{N(v^{'})}} \tag{7}$$

In MCTS, as its name suggests, the quality of an action is approximated through randomly guided simulation steps. The randomness introduces noises into the approximation. Simulations starting from upper layers are longer than those from the lower layers hence the variance is more severe. Most previous researches have been focused on accounting for the reward variance in the selection criterion.

Auer et al. (Auer et al., 2002) proposed a variant called the UCB1-Tuned method by replacing the second term in equation (6) with:

$$\sqrt{\frac{2lnN(v)}{N(v^{'})}min\left\{\frac{1}{4}, B(N(v^{'}))\right\}}$$

where $B(N(v^{'}))$ is given by:

$$B(N(v^{'})) = \frac{1}{N(v^{'})}\sum_{\tau=1}^{N(v^{'})} Q(v')^2 - \overline{X_{v^{'}}}^2 + c\sqrt{\frac{2lnN(v)}{N(v^{'})}} \tag{8}$$

This term gives the upper confidence bound a value that is the sample variance plus the exploration term of $\sqrt{\frac{2lnN(v)}{N(v^{'})}}$. In this study, we also replace the exploitation term with the max reward value:

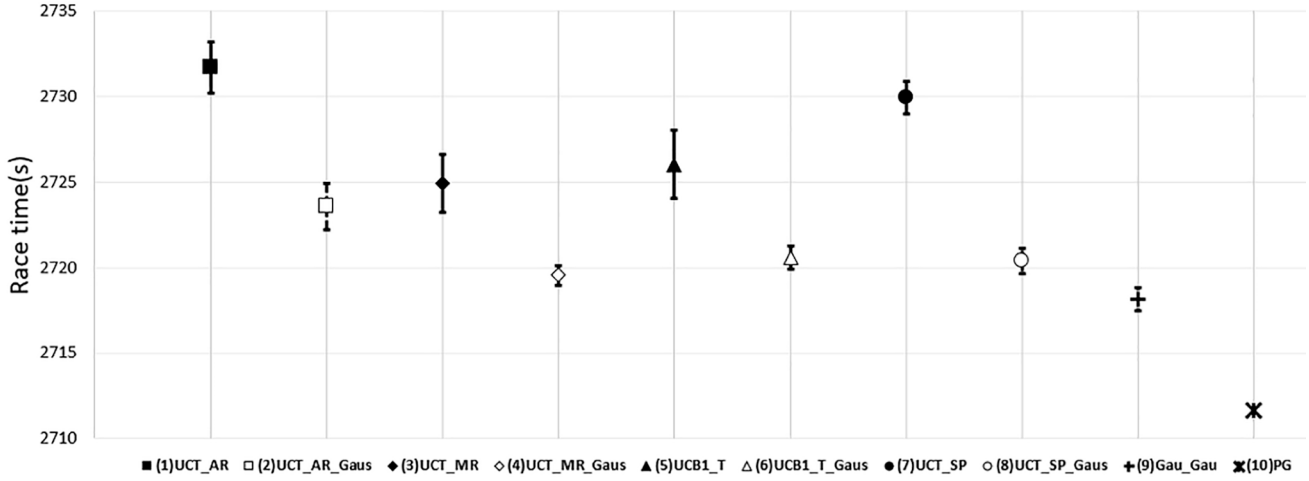


Fig. 6. Average race time results and variance of different methods.

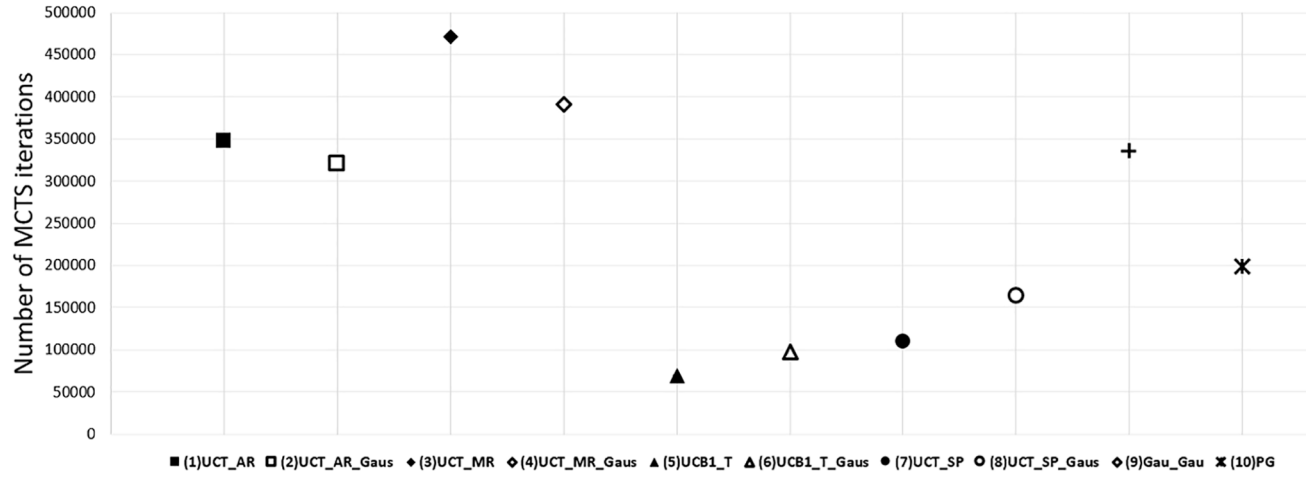


Fig. 7. Number of MTCS iterations at termination.

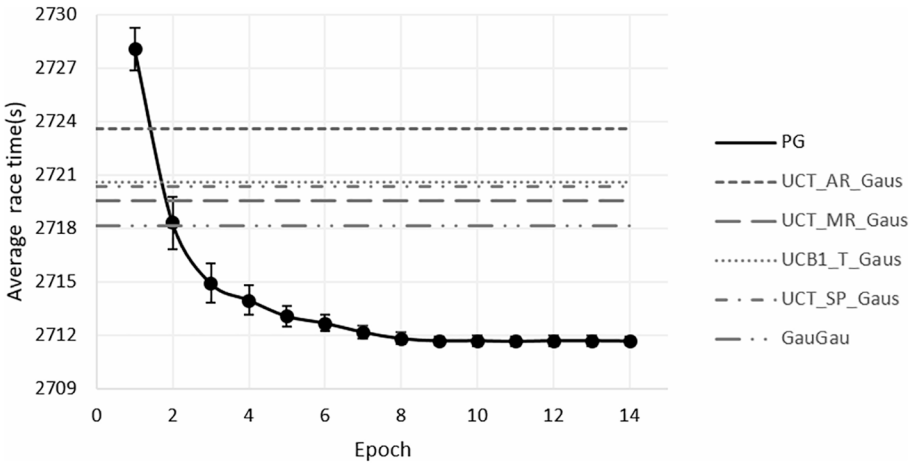


Fig. 8. Performance improvement through PPO progress.

Table 7
Performance improvement

Metrics	Base method	BGD simulation enhancement	Gau_Gau	PGRL
Race finishing time	UCT_AR	−2.964 ‰	−4.953 ‰	−7.337 ‰
	UCT_MR	−1.973 ‰	−2.482 ‰	−4.872 ‰
	UCB1_T	−1.997 ‰	−2.889 ‰	−5.279 ‰
	UCT_SP	−3.500 ‰	−4.311 ‰	−6.697 ‰
Variance	UCT_AR	−9.27 %	−53.4 %	−95.3 %
	UCT_MR	−65.6 %	−58.4 %	−95.8 %
	UCB1_T	−65.2 %	−64.7 %	−96.5 %
	UCT_SP	−19.6 %	−26.1 %	−92.6 %

$$UCB1_Tuned = Qmax(v') + c\sqrt{\frac{2lnN(v)}{N(v')}min\left\{\frac{1}{4}, \frac{1}{N(v')}\sum_{\tau=1}^{N(v')} Q(v')^2 - \overline{X_{v'}}^2 + \sqrt{\frac{2lnN(v)}{N(v')}}\right\}} \tag{9}$$

Instead of replacing the second term, literature has also proposed adding a third term to the original UCT formula. Schadd et al. (Schadd et al., 2008) introduced a third term written as

$$\sqrt{\sigma(v')^2 + \frac{D}{N(v')}}$$

where D is a constant and $\sigma(v')^2$ is the variance of the simulation results. This term describes the reward uncertainty level of a given node and is designed for a single player application. This gives the selection formula of:

$$UCT_SPMaxR = Qmax(v') + c\sqrt{\frac{2\ln N(v)}{N(v')}} + \sqrt{\sigma(v')^2 + \frac{D}{N(v')}} \tag{10}$$

The aforementioned methods are tested and discussed in Section 4.

3.2. Bivariate Gaussian distribution enhancement

Conventional full expansions and pure random simulations might introduce a large number of irrational actions and diverge the searching occasionally into relatively useless searching spaces. In terms of action space in the race strategy development problem, the size of the action space does not significantly reduce as the race progresses because the EPLs and QMs are independent of the states. This results in a large searching space of approximately 40^{34} to 80^{34} starting from the early stages of a race. While MCTS method has an advantage in solving such planning problems, given such a vast searching space, the computational time remains an issue because in real life strategic decision has to be made in seconds. To overcome this limitation, a fundamental solution is to reduce the size of the problem action space.

In terms of reward variance, the aforementioned UCT modifications mainly account for the variance of the rewards which originally comes from random simulation steps. Therefore, the simulation step also needs to be enhanced. There are two main reasons for doing so. First, the calculation of variance in the UCT will become a computational burden as searching progresses. Considering the number of MCTS iteration will easily reach above several hundreds of thousands, the variance calculation will slow down the searching performance while the searching tries to converge in the ending phase. Second, it would be irrational to have random actions all over the action space. From the real-life point of view, big changes in action sequence rarely happen unless something unexpected happens such as system failure or any form of race suspension. Mathematically, it is also not favourable for performance. While battery thermal dynamics is strongly nonlinear and difficult to conclude, the lap time sensitivity can provide a simpler insight into the issue. From the result presented in Fig. 3, it can be seen that the lap time as a function to ELP is concave. For multiple lap races with limited energy, to minimize the sum of lap times, choices of ELP are preferred to concentrate on the middle range instead of being split up on the two ends.

Therefore, the authors propose the use of bivariate Gaussian distribution (BGD) to guide the expansion step and simulation action sequence. The probability of action (EPL_C, QM_c) being taken at parent state s is:

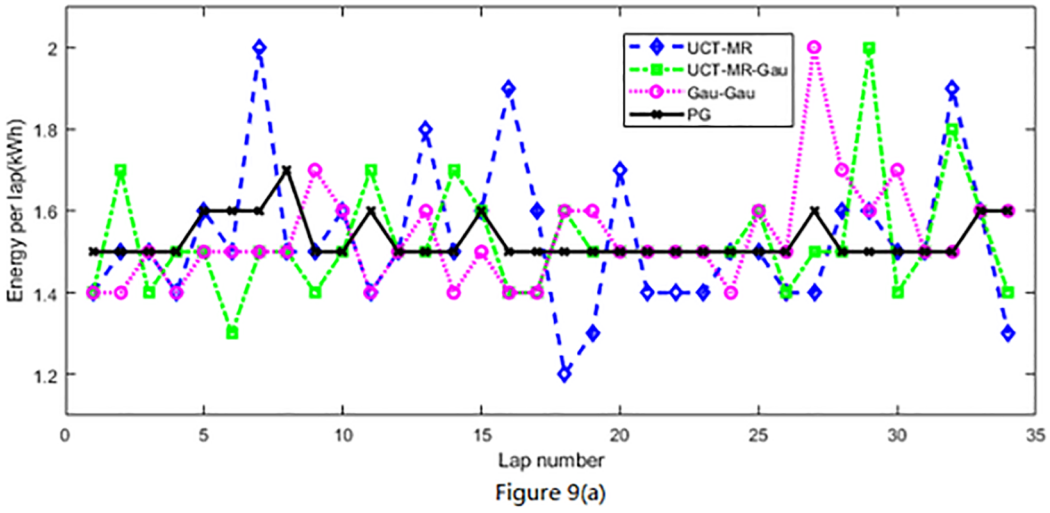
$$P((EPL_C, QM_c)|s) = (2\pi\sigma_1\sigma_2)^{-1}\exp\left(-\frac{1}{1-\rho^2}\left(\frac{(EPL_C - EPL_P)^2}{\sigma_1^2} - \frac{2\rho(EPL_C - EPL_P)(QM_c - QM_P)}{\sigma_1\sigma_2} + \frac{(QM_c - QM_P)^2}{\sigma_2^2}\right)\right) \tag{11}$$

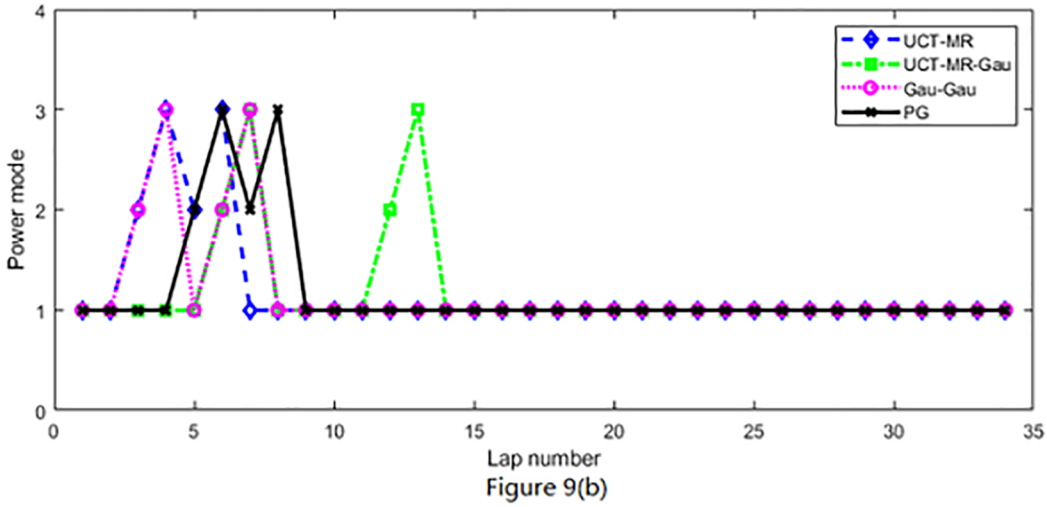
where σ_1 , σ_2 and ρ are shaping factors of bivariate Gaussian distribution. The subscripts of C and P indicate the variable is from child state and parent state respectively. This equation assigns every action in the action space with probability values according to which action was previously picked. Fig. 4 shows a visualized example of this probability distribution with parent action of $EPL_P = 1.6$ and $QM_P = 1$. Fig. 4b shows the probability distribution on the Power mode 1(200 kW) actions which are shown by blocks numbered 0,1,2,3 in Fig. 4a indicating the Q mode. To clarify Fig. 4a, in the 120 total actions, 0–29 are for Q mode 0; 30–59 are for Q mode 1; 60–89 are under Q mode 2 and 90–119 are for Q mode 3. The tripling curves in each Q mode demonstrate actions for different power modes which are 1,2,3 from left to right. The distribution among different power modes is identical because the bivariate Gaussian distribution does not influence when to activate the attack mode.

This simulation enhancement will be integrated with the selection methods (introduced in the previous section) and the results will be discussed later in Section 4. Also, the BGD technique will be used to guide the expansion step where only a predefined number of new nodes will be added instead of a full expansion leading to 40–80 child nodes.

3.3. Proximal policy Optimization(PPO)

As previously discussed, it would be beneficial if the expansion and simulation steps can be only focused on more promising actions by neglecting the irrational ones. However, it is difficult to determine





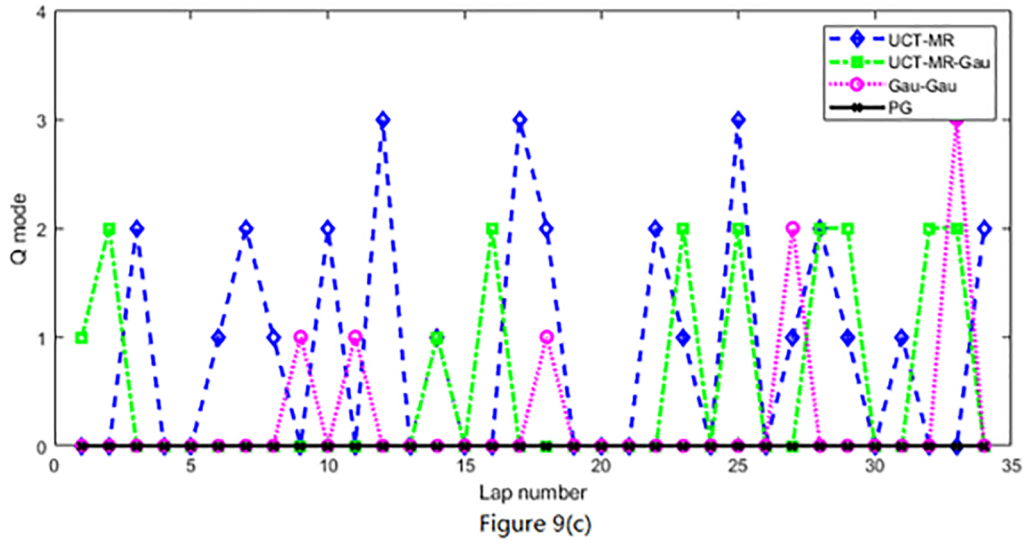


Fig. 9. Pre-race planning solutions in Case 1: (a) Energy per lap (EPL); (b) Power mode; (c) Q mode.

which actions are more promising in various scenarios (i.e. energy limited, thermal limited or both). Although the BGD technique helps to shrink the size of the solution space, there is a chance that BGD hides some potentially promising actions too. So, in this study, PPO is implemented as an on-policy model-free learning algorithm to develop policy used in MCTS expansion and simulation steps.

Table 8
Terminal states using different methods in Case 1

Method	Remaining energy (kWh)	Battery temperature (℃)	Race finishing time(s)
UCT_MR	0.1	55.18	2724.6
UCT_MR_Gaus	0.3	55.08	2720.7
Gau_Gau	0	57.19	2716.2
PG	0	57.5	2711.6

3.3.1. Actor and critic network layout

PPO method requires two networks, namely a critic network for evaluating a state and an actor network mapping a state into a probability distribution over the action space. In this study, both networks are implemented with fully connected layers with the same input size (i.e. state vector of $[N_{lap}, E_r, T_{bat}, T_{amb}, N_{att}, N_{Rattlap}]$). The critic network has one output neuron and the actor has an output size of 120. Details of actor and critic networks and hyperparameters used in this study are listed in appendix B.

3.3.2. PPO algorithm with MCT

PPO algorithm aims to maximize the objective of:

$$L^{CLIP}(\theta) = \widehat{E}_t[\min(r_t(\theta)\widehat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t)] \quad (12)$$

where $r_t(\theta)$ denotes the probability ratio $\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}\right)$, ϵ is a hyperparameter of clipping ratio. Clipping the probability ratio is an effective method to stabilize the policy updating process by removing the incentive for moving $r_t(\theta)$ outside of the interval $[1-\epsilon, 1+\epsilon]$. This ensures the new policy does not deviate too much from the old policy which creates instability. $\widehat{A}_t$ is an estimator of the advantage function at timestep t. For a length-T trajectory, $\widehat{A}_t$ can be calculated by:

$$\widehat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \quad (13)$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, γ is the discount factor and λ 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 $L_t^{VF} = (V_\theta(s_t) - V_t^{targ})^2$.

There are two main reasons why PPO is used in this study. First, it is an on-policy which in general converges faster than off-policy algorithms(Labao, Martija, & Naval, 2021; Lapan, 2018). In this study, the experiences used for training are collected from tree searches. Compared to the RL training process, the data collection is a much more time-consuming process which would potentially cause data shortage in a replay buffer. 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 being influenced by more unpredictable factors such as track conditions (wet or dusty), incidents during a race and more importantly multiple opponents. This makes model-learning for a race environment very infeasible. Therefore, a model-free RL algorithm would build a solid foundation if more complex scenarios are to be investigated.

Theoretically, PPO is able to learn the policy on itself. However, in this study, we choose to use PPO together with MCTS. This leads to two major advantages: (1) Although PPO as an on-policy method, in general, converges faster than off-policy ones, it still cost quite an amount of time in exploration especially at the very beginning when the PPO networks are less intelligent and trying to find a feasible solution from a poor policy. On contrary, MCTS can generate a feasible solution even though with a poor policy. Using MCTS can guarantee the data generation quality at each policy update iteration therefore the convergence can be further accelerated; (2) Finding the precise PPO terminating time is a very difficult task because is very hard to determine if the PPO reaches the optima or is still on its way and over-running the PPO will very likely to cause a 'catastrophic forgetting' scenario. In this case, MCTS provides additional robustness to the process. Due to its own exploration–exploitation feature, MCTS is still capable of finding the optimal solution given a sub-optimal policy. This means the PPO process does not have to be terminated precisely at the perfect time. MCTS can compensate for the sub-optimality thus the robustness and optimality of the entire process is guaranteed.

3.3.3. Algorithms implementation

In general, for each PPO iteration, a number of solutions will be generated by MCTS. The generated sequence will be used for policy updates of the PPO. After each update, MCTS will use the new policy in its expansion and simulation steps and generate new solution sequences. The complete PPO process and MCTS algorithm used in this study can be described in the following pseudocode.

Algorithm 1 PPO

Initialize PPO actor and critic network parameter θ^μ , θ^Q

For PPO iteration = 1, M **do**
Initialize replay buffer R
Repeat episode
  Initialize $s_0 = [N_{lap}, E_r, T_{bat}, T_{amb}, N_{att}, N_{Rattlap}]$
  Store result sequence $Q(A,S), A, S \leftarrow MCTS(s_0, \theta^\mu)$ in R
  Compute advantage estimates $\widehat{A}_1, \cdots, \widehat{A}_T$ for the sequence
Until episode number reached
for epoch = 1, K **do**
Sample a random minibatch of N transitions (S, A, $Q(A,S)$) from R
Update actor and critic network using $L^{CLIP}(\theta^\mu)$ and L_t^{VF}
End for
End for

Algorithm 2 Monte Carlo tree search

Function MCTS(s_0, θ^μ)
  Create root node v_0 with state s_0
    while within computational budget do
      $v_l \leftarrow$ TreePolicy (v_0, θ^μ)
      $\Delta \leftarrow$ Simulation($s(v_l), \theta^\mu$)
    Backup (v_l , Δ)
  return highest-reward sequence $Q(A,S), A, S$
Function TreePolicy(v, θ^μ)
  while v is not terminal **do**
    if$v$is not expanded **then**
      $v \leftarrow$ Expansion(v,θ^μ)
    Else
      $v \leftarrow$ Bestchild(v) based on UCT-MaxR(v)
  Return v
Function Expansion(v, θ^μ)
  For i = 1,Max Expansion Number **do**
  choose a $\in$ untried actions from A($s(v)$) based on policy $\pi_{\theta^\mu}(a|s)$
  add a new child v' to v
    with s(v') = Transition(s(v) ,a)
end for
    $v \leftarrow$ first child $v'(1)$
return v
Function Simulation (s, θ^μ)
while s is not terminal **do**
    choose a $\in$ A($s(v)$) based on policy $\pi_{\theta^\mu}(a|s)$
    $s \leftarrow$ Transition(s ,a)
  return reward for terminal state s

3.3.4. Result collection

In this section, the methods used to collect the result is clarified. The stopping criteria of the MCTS algorithm is usually defined according to a computational budget (Browne et al., 2012) such as computation time,

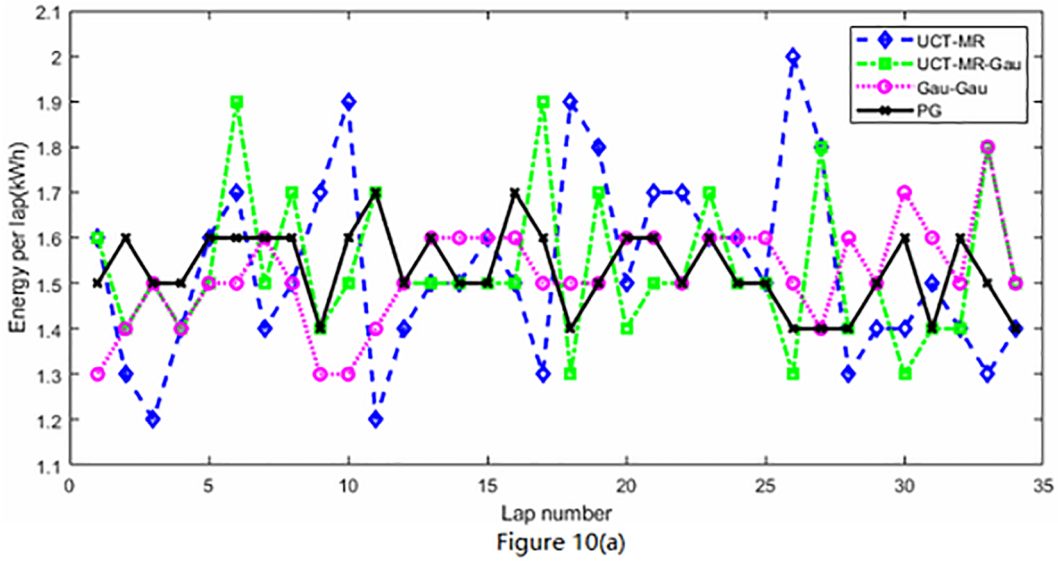


Figure 10(a)

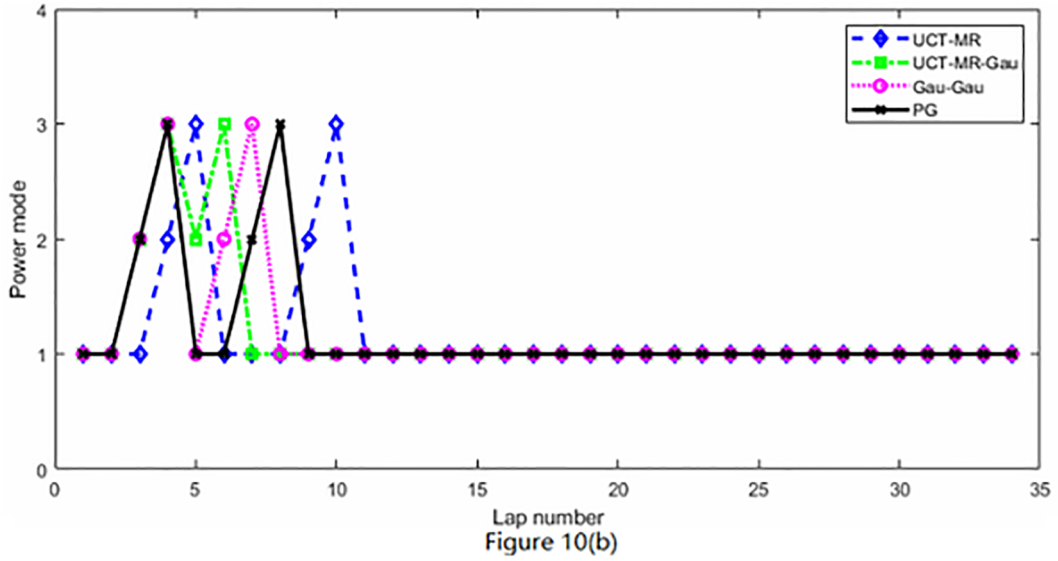


Figure 10(b)

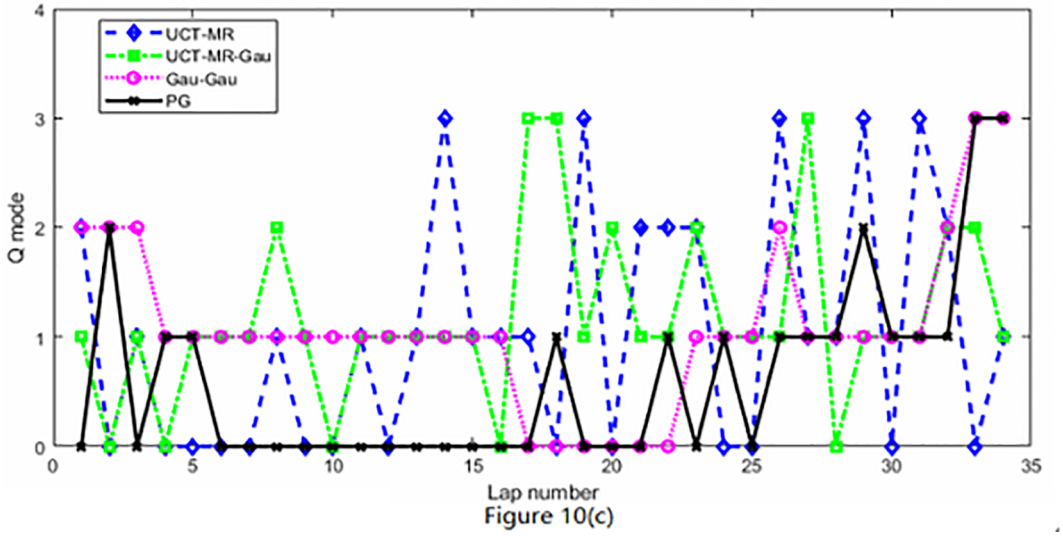


Figure 10(c)

Fig. 10. Pre-race planning solutions in Case 2: (a) Energy per lap (EPL); (b) Power mode; (c) Q mode.

Table 9
Terminal states using different methods in Case 2

Method	Remaining energy (kWh)	Battery temperature (℃)	Race finishing time(s)
UCT_MR	0.05	57.82	2726.8
UCT_MR_Gaus	0.1	57.79	2723.4
Gau_Gau	0.3	57.86	2722.2
PG	0	57.98	2716.8

memory and/or the number of iterations. Without defining stopping criteria, the iteration can go on forever, keeping growing the tree. Because decision making time is crucial in strategy development especially when trying to adapt to changes during a race, in this study, the stopping criteria is set to be 30 s of computation time.

Björnsson and Finnsson (Bjornsson & Finnsson, 2009) pointed out that an optimal solution might be hidden due to its weak siblings. This is one of the biggest concerns in applications of such a race strategy development where the fastest race time is desirable instead of an average faster time. Moreover, given a vast searching space and a stopping criteria of time, it is very likely that the search is terminated prematurely while the agent is tending to explore and grow the tree and lands on a poor branch. Therefore, in this study, every simulation reward is tracked and the best one is returned when the search is terminated.

To avoid confusion in the result analysis section, the abbreviations used for each method are presented in Table 5.

Overall, the methods used in this study can be categorized into three groups. The first group (methods 1–4 in Table 5 comprises different formulas (eq. (6), 7, 9, 10) used in the selection process of MCTS while the simulation and expansion processes remain the same as an ordinary MCTS. The modifications in this group essentially aim to improve the MCTS search quality by accounting for more information on the back-propagated rewards. Based on the first group, methods with a suffix of

Table 10
Scenario definition.

	Originally planned	Scenario 1	Scenario 2
Number of remaining laps	10	10	10
Available attacks	0	0	0
Remaining energy(kWh)	14.7	16	14
Battery temperature(℃)	44.82	43	45.2
Ambient temperature(℃)	30	30	32

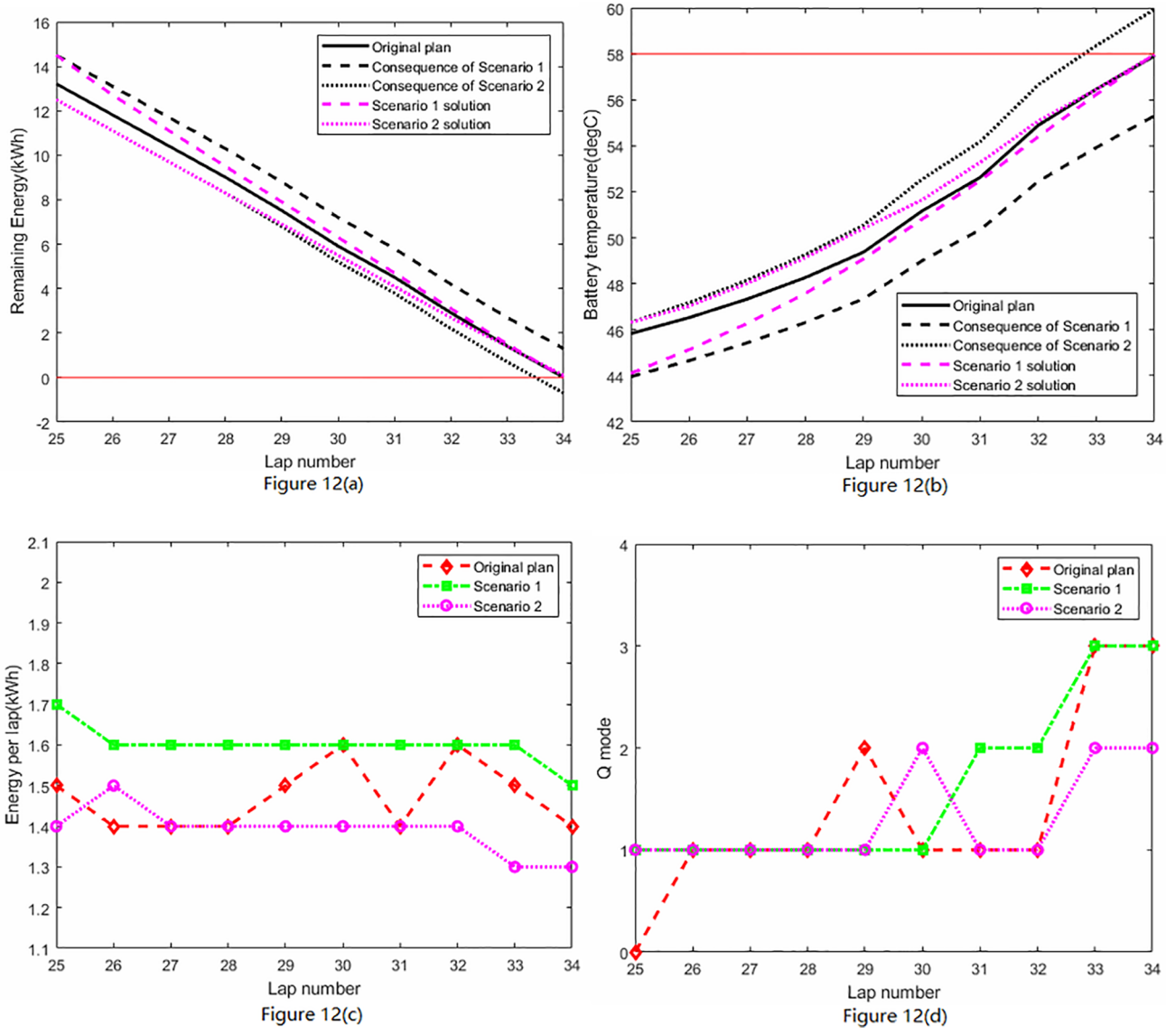


Fig. 11. In-race scenario solutions: (a) Energy data, (b) Battery temperature, (c) EPL, (d) Q mode.

'_Gau' (methods 5–8) form the second group where the BGD technique is introduced in the simulation process. The concept of this modification is to rationalize (i.e. reduce the degree of randomness) the simulation process therefore the quality of the tree search solution can be better approximated by the collected rewards. The third group contains two methods (methods 9–10) whose simulation and expansion processes are both modified. Both methods aim to rationalize the simulation and shrink the expansion down on potentially more promising actions. While the Gau_Gau method uses knowledge-based distribution (i.e. BGD) for both simulation and expansion processes, the PPO-enhanced method uses distribution information from the actor network trained based on collected tree search solutions. Because the distribution in the latter method is not empirically restrained, it enhances exploration capability thus potentially become able to find better solutions (Fig. 5).

4. Results and discussion

4.1. Performance

As previously pointed out in (Liu & Fotouhi, 2020), the conventional MCTS method tends to generate occasionally sub-optimal solutions with a big variance. In the race strategy development, the tool is expected to generate higher reward solutions with decent stability. In this section, different methods are compared in terms of their result quality. The battery temperature limit is set to 58℃ (Jowett, 2018) and the initial condition used in this section is as presented in Table 6.

For comparison, each method has been deployed repetitively to collect 200 results of a full race (i.e. 34 laps). A comparison of average race time and its variance using different methods is shown in Fig. 6. The effect of BGD guiding simulation steps can be clearly seen. With BGD enhancement on the simulations, all four UCT variations generate significantly faster race times and much less variance (method 2, 4, 6, 8 compared to 1, 3, 5, 7). Among the first eight methods (full expansion), the UCT_AR method gives the worst average race time solutions which conforms to the previous discussion that strong branches are very likely to be hidden by weak siblings. In contrast, UCT_MR_Gaus has generated the fastest time among the full expansion methods. However, none of these matched the performance of Gau_Gau which generated the best race times and variance among the non-PPO methods. It should be noted that UCB1_T(_Gaus) and UCT_SP(_Gaus) methods failed to generate better solutions than UCT_MR(_Gaus) methods while theoretically they should have, because they further account for variance in the UCT formula. The reason underneath can be explained by looking at other results shown in Fig. 7 that includes the number of iterations completed by the time that searches were terminated.

While the searching space is difficult to be visualized, the number of MCTS iterations provides a good alternative insight into how much searching has been completed. It can be seen in Fig. 7 that the UCB1_T (_Gaus) and UCT_SP(_Gaus) methods completed much fewer iterations than the UCT_AR(_Gaus) and UCT_MR(_Gaus) methods. This is mainly because of the added computational complexity of the variance calculations. As a result, they were terminated less matured and could not yet find better solutions. This also explains why authors choose to use equation (7) for the Gau_Gau and PPO methods (Table 5). Although with BGD technique is applied to both the expansion and simulation steps, the Gau_Gau method still performs weaker than the PPO method as can be clearly seen in Fig. 6.

It should be noted that the PPO result shown in Figs. 6 and 7 were collected using the final product after the reinforcement learning process (14 epochs). The full PPO progress is illustrated in Fig. 8. It can be seen that at the beginning of the reinforcement learning, the PPO method performed much weaker than the BGD-enhanced full-expansion methods. The random initialization of PPO actor network parameters with a partial expansion makes it very likely to neglect strong play branches. However, as the actor parameters update every iteration after another, the performance significantly improved. The average race time became faster and the variance decreased. As shown in Fig. 8, it takes only one epoch to surpass the full-expansion methods becoming very close to the Gau_Gau method. Then just after another iteration, its performance surpassed the Gau_Gau method. The improvement starts to flatten after 9 iterations and the variance becomes tiny which suggests a very stable performance. The 14 iterations of training in this study took about 4 h.

In general, the Gau_Gau method appeared to be a promising one with decent performance. Although it is not as good as the final PPO product, Gau_Gau's average result and variance are clearly better than the other full expansion or random methods (method 1–8). Additionally, the Gau_Gau method does not require training time as the PPO method. This makes it a preferable substitute when there is not enough time for the PPO learning phase. The improvement of BGD and PPO methods is summarized in Table 7.

Based on the performance result, in the next section, only UCT_MR, UCT_MR_Gau, Gau_Gau and PPO methods are used for demonstration.

4.2. Race strategy solutions

This section presents the details of race strategy solutions generated by the MCTS methods. The results are collected for two different applications. The first application is pre-race planning which tests the deep searching capability of the algorithms. Based on the Marrakech track used in this study, this calls for a full 34-layer search. Two cases with different ambient temperatures are presented.

5. Pre-race planning

5.1. Case 1: Ambient temperature of 25℃

In this case, the initial states are set the same as in section 4.1 with the ambient temperature of 25℃ which would not cause too much cooling issue. The solutions generated by the selected methods are presented in Fig. 9 and Table 8.

In terms of power mode selections, all four methods activated attack mode in the first half of the race. UCT_MR and PPO methods suggested using the available two attacks one immediately after another while the other two methods chose to have normal race mode (Power mode 1) laps between the attacks. Fig. 9a shows that the majority of EPL selections suggested by the four methods lied around the 1.5kWh options. The PPO method used only between 1.5 and 1.7 options among which the 1.6 and 1.7 options were used only on the attack mode laps and a few other laps. In contrast, variance in the sequence can be clearly observed in the other three methods which as previously discussed in section 3.2, is not favourable. Q mode choices are mainly used to manage the battery temperature rise. But Q mode of 1, 2 and 3 can definitely compromise the lap time performance according to the OCP results (section 3.2). The PPO method managed to find a path where no heat-saving actions have to be taken while the others all have in their solutions. The UCT_MR method had the most non-zero actions and Gau_Gau had the least among non-PPO methods.

Table 8 reveals the terminal states generated by these methods. Gau_Gau and PPO method utilized the full available energy whereas UCT_MR and UCT_MR_Gaus failed to do so which means there is still a rather big space for better search results. Regarding the battery temperature constraint, the over conservative actions left a big margin between the terminal battery temperature and the constraint of 58℃. In terms of race finishing time, the PPO method remains the best conforming to the previous performance analysis.

6. Case 2: Ambient temperature of 30℃

For this case, the ambient temperature is raised to 30℃ which is a common condition in Marrakech. This would potentially result in stricter heat-saving actions being taken. The solutions from the four

methods are shown in Fig. 10 and Table 9.

Similar to the first case, the EPL actions still fall into the 1.4–1.7 range. This is understandable that to make full use of the same amount of energy, the EPL range would not change too much. However, it can be observed in Case 2 that the actions sequence becomes more inconsistent. The power mode solutions have similar patterns with Case 1 showing the attack modes tend to be activated in the early stage of the race. The biggest difference between the two cases can be seen in the Q mode solutions. While the PPO method found a path of Q mode = 0 in Case 1, in Case 2 with higher ambient temperature, the PPO method starts to suggest higher Q modes to be taken to manage the battery temperature. The Q mode = 1 actions are most frequently taken with a few Q mode = 2,3 actions being taken at the finishing stage.

Another point to mention when comparing results of Case 1 and 2 is the longer race finishing time when the ambient temperature increases. As stated in Table 9, the best performance (i.e. for PPO method) in Case 2 is 2716.8 s for race finishing time whereas this record was 2711.6 s in Case 1 which means around 5 s slower finish due to 5℃ higher ambient temperature.

From the terminal states information, the PPO method has pushed the resource usage to the absolute boundaries (Table 9) and generated the fastest race finishing time. Others have left a margin on both energy and temperature, hence a large searching to be further explored.

One crucial element of race-level planning is to decide when to activate the attack mode. The regulation states that it cannot be activated in the first two laps of a race. It can be seen from the result that the PPO method suggests that the attack mode should be activated as early as it is allowed. This can also be told from the policy network itself which is another advantage of using the policy network. Without re-running the cases, the policy network can tell which actions are more preferred by simply inputting a state of interest. Fig. 11 shows the policy network outputs of two different states: state 1 from the early stage and state 2 from the later stage of a race both with available attacks. The blocks in the figure denote the attack mode activation actions (Power mode = 2). The full description of the action space is shown in Appendix C.

6.0.1. *In-race scenario*

In real life Formula-E races, various incidents may happen which make the race progress deviated from what was originally planned. This would lead to either looser or tighter constraints. Two scenarios are defined in Table 10 assuming to emerge at the 10th to the last lap. Because the PPO method has been proved much more powerful than the rest, the results are collected using only the PPO method to see what actions need to be taken to adapt to these scenarios.

Two scenarios are considered here: (1) Scenario 1 represents a slightly loose change which might happen in real life due to previously over-conservative driving giving more available energy and lower battery temperature, and (2) In contrast, in the second scenario the resources are over-consumed which is very likely to happen due to over-aggressive driving. To make it more critical, the ambient temperature rises by 2℃.

The solution results of both scenarios are shown in Fig. 11. The black dash and dot lines in Fig. 11a and 11b show the consequences of no action changes in these two scenarios. If sticking to the original plan, there would be plenty of resources left unused in scenario 1 and the battery will go flat and overheated in scenario 2. To adapt to scenario 1, the solution suggests raising the EPL level to around 1.6 kWh to fully use the available energy and Q mode 1 needs to be chosen initially and then gradually raise to 2 and 3 in the last four laps to manage the battery temperature rise. For scenario 2, EPL has to be lowered to save energy. Because lower EPL generates less heat, only Q modes of 1 and 2 need to be used. As a result, it can be seen from Fig. 11a and 11b that the new solutions made full use of the remaining energy and managed the battery temperature below the limit.

7. Conclusions

In this study, the Formula-E race strategy was formulated into a multi-layer decision making problem and its solutions were proposed using enhanced MCTS-based algorithms. Bivariate Gaussian Distribution (BGD) was proposed as an enhancement technique to the MCTS and the result was compared against other enhancement techniques in the literature. According to the results, BGD proves to be a strong enhancement method on the simulation steps replacing the random action sequence. It improved the result variance and consequently the consistency of the MCTS. The variance was reduced by 9%-65% and the average race finishing time has been reduced by 2‰-3.5‰. When BGD is further used on the expansion step to reduce the size of the problem, it helped to reduce the variance by at least 26% and the average race finishing time was reduced by 2.5‰-5‰. The double BGD enhancement (Gau_Gau) method was also investigated that demonstrated much higher improvement than the other conventional enhancement techniques in this race strategy development application.

Proximal policy optimization proved to be another powerful method to enhance the MCTS performance in this study. The final product after the self-learning process significantly reduced the result variance by over 95% and the average race time by 4.9‰-7.3‰. The PPO method outperformed the other techniques in all studied cases. It was also concluded that the PPO method has the best performance but would require aforehand training time. So, it would be the most suitable choice when given enough preparation time. On the other hand, the Gau_Gau method is suggested as a strong alternative when the time is not guaranteed. Both methods can play a strong element in race strategy development in their suitable applications.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eswa.2022.116718.

References

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning, 47*(2/3), 235–256. https://doi.org/10.1023/A:1013689704352

Bekker, J., & Lotz, W. (2009). Planning Formula One race strategies using discrete-event simulation. *Journal of the Operational Research Society, 60*(7), 952–961. https://doi.org/10.1057/palgrave.jors.2602626

Bjornsson, Y., & Finnsson, H. (2009). CadiaPlayer: A Simulation-Based General Game Player. *IEEE Transactions on Computational Intelligence and AI in Games, 1*(1), 4–15. https://doi.org/10.1109/TCIAIG.2009.2018702

Brayshaw, D. L., & Harrison, M. F. (2005). A quasi steady state approach to race car lap simulation in order to understand the effects of racing line and centre of gravity location. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 219*(6), 725–739. https://doi.org/10.1243/095440705X11211

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., … Colton, S. (2012). A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games, 4*(1), 1–43. https://doi.org/10.1109/TCIAIG.2012.2186810

Chaslot, G. M. J.-B., Winands, M. H. M., Herik, H. J. van den, Uiterwijk, J. W. H. M., & Bouzy, B. (2008). PROGRESSIVE STRATEGIES FOR MONTE-CARLO TREE SEARCH. *New Mathematics and Natural Computation*, *04*(03), 343–357. 10.1142/S1793005708001094.

Zhang, C., & Vahidi, A. (2012). Route Preview in Energy Management of Plug-in Hybrid Vehicles. *IEEE Transactions on Control Systems Technology, 20*(2), 546–553. https://doi.org/10.1109/TCST.2011.2115242

Choo, C. L. W. (2015). *Real-time decision making in motorsports: analytics for improving professional car race strategy*.

Du, Y., Zhao, Y., Wang, Q., Zhang, Y., & Xia, H. (2016). Trip-oriented stochastic optimal energy management strategy for plug-in hybrid electric bus. *Energy, 115*, 1259–1271. https://doi.org/10.1016/j.energy.2016.09.056

FIA. (2021a). F1 Regulations.
FIA. (2021b). FE Regulations.
FIA. (2021c). WEC Regulations.
Finnsson, H., & Björnsson, Y. (2010). Learning simulation control in general game-playing agents. *Twenty-Fourth AAAI Conference on Artificial Intelligence*.
Finnsson, Hilmar. (2007). *CADIA-Player: A General Game Playing Agent*.
Heilmeier, A., Geisslinger, M., & Betz, J. (2019). A Quasi-Steady-State Lap Time Simulation for Electrified Race Cars. *2019 Fourteenth International Conference on Ecological Vehicles and Renewable Energies (EVER)*, 1–10. IEEE. 10.1109/EVER.2019.8813646.
Heilmeier, A., Graf, M., & Lienkamp, M. (2018). A Race Simulation for Strategy Decisions in Circuit Motorsports. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2986–2993. IEEE. 10.1109/ITSC.2018.8570012.
Jowett, R. (2018). Battery Thermal Management in Formula E.
Kocsis, L., Szepesvári, C., & Willemson, J. (2006). *Improved Monte-Carlo Search*.
Labao, A. B., Martija, M. A. M., & Naval, P. C. (2021). A3C-GS: Adaptive Moment Gradient Sharing With Locks for Asynchronous Actor-Critic Agents. *IEEE Transactions on Neural Networks and Learning Systems, 32*(3), 1162–1176. https://doi.org/10.1109/TNNLS.2020.2980743
Lapan, M. (2018). 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.
Limebeer, D. J. N., & Perantoni, G. (2015). Optimal Control of a Formula One Car on a Three-Dimensional Track—Part 2: Optimal Control. *Journal of Dynamic Systems, Measurement, and Control, 137*(5). https://doi.org/10.1115/1.4029466
Limebeer, D. J. N., Perantoni, G., & Rao, A. V. (2014). Optimal control of Formula One car energy recovery systems. *International Journal of Control*, 1–16. https://doi.org/10.1080/00207179.2014.900705
Liu, X., & Fotouhi, A. (2020). Formula-E race strategy development using artificial neural networks and Monte Carlo tree search. *Neural Computing and Applications, 32*(18), 15191–15207. https://doi.org/10.1007/s00521-020-04871-1
Liu, X., Fotouhi, A., & Auger, D. J. (2020). Optimal energy management for formula-E cars with regulatory limits and thermal constraints. *Applied Energy, 279*, 115805. https://doi.org/10.1016/j.apenergy.2020.115805
Peng, J., He, H., & Xiong, R. (2017). Rule based energy management strategy for a series–parallel plug-in hybrid electric bus optimized by dynamic programming. *Applied Energy, 185*, 1633–1643. https://doi.org/10.1016/j.apenergy.2015.12.031
Perantoni, G., & Limebeer, D. J. N. (2014). Optimal control for a Formula One car with variable parameters. *Vehicle System Dynamics, 52*(5), 653–678. https://doi.org/10.1080/00423114.2014.889315
Perantoni, G., & Limebeer, D. J. N. (2015). Optimal Control of a Formula One Car on a Three-Dimensional Track—Part 1: Track Modeling and Identification. *Journal of Dynamic Systems, Measurement, and Control, 137*(5). https://doi.org/10.1115/1.4028253
Gong, Q., Li, Y., & Peng, Z.-R. (2008). Trip-Based Optimal Power Management of Plug-in Hybrid Electric Vehicles. *IEEE Transactions on Vehicular Technology, 57*(6), 3393–3401. https://doi.org/10.1109/TVT.2008.921622
Rimmel, A., & Teytaud, F. (2010). *Multiple Overlapping Tiles for Contextual Monte Carlo Tree Search*. Doi: 10.1007/978-3-642-12239-2_21.
Schadd, M. P. D., Winands, M. H. M., van den Herik, H. J., Chaslot, G. M. J.-B., & Uiterwijk, J. W. H. M. (2008). *Single-Player Monte-Carlo Tree Search*. 10.1007/978-3-540-87608-3_1.
Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., … Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature, 550*(7676), 354–359. https://doi.org/10.1038/nature24270
Silver, D., Sutton, R. S., & Müller, M. (2008). Sample-based learning and search with permanent and transient memories. *Proceedings of the 25th International Conference on Machine Learning - ICML '08*, 968–975. New York, New York, USA: ACM Press. 10.1145/1390156.139027.
Silver, D., & Tesauro, G. (2009). *Monte-Carlo simulation balancing, ICML '09*, 1–8. https://doi.org/10.1145/1553374.1553495
Sulsters, C., & Bekker, R. (2018). *Simulating Formula One Race Strategies*.
Tremlett, A. J., & Limebeer, D. J. N. (2016). Optimal tyre usage for a Formula One car. *Vehicle System Dynamics, 54*(10), 1448–1473. https://doi.org/10.1080/00423114.2016.1213861
Tremlett, A. J., Massaro, M., Purdy, D. J., Velenis, E., Assadian, F., Moore, A. P., & Halley, M. (2015). Optimal control of motorsport differentials. *Vehicle System Dynamics, 53*(12), 1772–1794. https://doi.org/10.1080/00423114.2015.1093150
van den Broeck, G., Driessens, K., & Ramon, J. (2009). *Monte-Carlo Tree Search in Poker Using Expected Reward Distributions.*. https://doi.org/10.1007/978-3-642-05224-8_28
Wang, H., Huang, Y., Khajepour, A., & Song, Q. (2016). Model predictive control-based energy management strategy for a series hybrid electric tracked vehicle. *Applied Energy, 182*, 105–114. https://doi.org/10.1016/j.apenergy.2016.08.085
Wieczorek, M., & Lewandowski, M. (2017). A mathematical representation of an energy management strategy for hybrid energy storage system in electric vehicle and real time optimization using a genetic algorithm. *Applied Energy, 192*, 222–233. https://doi.org/10.1016/j.apenergy.2017.02.022
Winands, M. H. M., & Björnsson, Y. (2010). *Evaluation Function Based Monte-Carlo LOA.*. https://doi.org/10.1007/978-3-642-12993-3_4
Xie, F., & Liu, Z. (2009). *Backpropagation Modification in Monte-Carlo Game Tree Search. 2009 Third International Symposium on Intelligent Information Technology Application* (pp. 125–128). IEEE.
Zhang, S., & Xiong, R. (2015). Adaptive energy management of a plug-in hybrid electric vehicle based on driving pattern recognition and dynamic programming. *Applied Energy, 155*, 68–78. https://doi.org/10.1016/j.apenergy.2015.06.003

https://dspace.org/

Application of advanced tree search and proximal policy optimization on formula-E race strategy development

Liu, Xuze; Fotouhi, Abbas; Auger, Daniel J.

2022-02-25
Elsevier
https://doi.org/10.1016/j.eswa.2022.116718
Liu X, Fotouhi A, Auger D. (2022) Application of advanced tree search and proximal policy optimization on formula-E race strategy development, Expert Systems with Applications, Volume 197, July 2022, Article number 116718

Downloaded from a DSpace repository. https://dspace.org/