electronics

MDPI

Article

Beyond Trial and Error: Lane Keeping with Monte Carlo Tree Search-Driven Optimization of Reinforcement Learning

Bálint Kővári [1,2], Bálint Pelenczei [3], István Gellért Knáb [3] and Tamás Bécsi [1,*]

[1] Department of Control for Transportation and Vehicle Systems, Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics, H-1111 Budapest, Hungary; kovari.balint@kjk.bme.hu
[2] Asura Technologies Ltd., H-1122 Budapest, Hungary
[3] Systems and Control Laboratory, HUN-REN Institute for Computer Science and Control (SZTAKI), H-1111 Budapest, Hungary; pelenczei.balint@sztaki.hun-ren.hu (B.P.); knab.istvan.gellert@sztaki.hun-ren.hu (I.G.K.)
* Correspondence: becsi.tamas@kjk.bme.hu

Abstract: In recent years, Reinforcement Learning (RL) has excelled in the realm of autonomous vehicle control, which is distinguished by the absence of limitations, such as specific training data or the necessity for explicit mathematical model identification. Particularly in the context of lane keeping, a diverse set of rewarding strategies yields a spectrum of realizable policies. Nevertheless, the challenge lies in discerning the optimal behavior that maximizes performance. Traditional approaches entail exhaustive training through a trial-and-error strategy across conceivable reward functions, which is a process notorious for its time-consuming nature and substantial financial implications. Contrary to conventional methodologies, the Monte Carlo Tree Search (MCTS) enables the prediction of reward function quality through Monte Carlo simulations, thereby eliminating the need for exhaustive training on all available reward functions. The findings obtained from MCTS simulations can be effectively leveraged to selectively train only the most suitable RL models. This approach helps alleviate the resource-heavy nature of traditional RL processes through altering the training pipeline. This paper validates the theoretical framework concerning the unique property of the Monte Carlo Tree Search algorithm by emphasizing its generality through highlighting crossalgorithmic and crossenvironmental capabilities while also showcasing its potential to reduce training costs.

Keywords: autonomous vehicles; reinforcement learning; lane keeping assist systems; Monte Carlo methods; vehicle dynamics

Citation: Kővári, B.; Pelenczei, B.; Knáb, I.G.; Bécsi, T. Beyond Trial and Error: Lane Keeping with Monte Carlo Tree Search-Driven Optimization of Reinforcement Learning. *Electronics* **2024**, *13*, 2058. https://doi.org/10.3390/electronics13112058

Academic Editors: Arkaitz Zubiaga, Yanping Zhang, Wenlin Han and Jianjun Yang

Received: 25 March 2024
Revised: 3 May 2024
Accepted: 23 May 2024
Published: 25 May 2024

1. Introduction

Nowadays, the domain of vehicle control and its closely associated field of autonomous driving stands out as one of the most dynamically evolving sectors [1–3]. The genesis of such advancements can be attributed to various factors, including the perpetual risk to human safety in transportation scenarios [4]. The adoption of data-driven design not only presents the advantage of low latency decision-making but also frequently surpasses the efficiency of human actions and conventional solutions.

While the integration of black box models in engineering demands meticulous consideration, encompassing legal and safety perspectives [5], it is evident that delegating tasks, which are not directly safety-critical, such as lane keeping assistance or a lane departure warning system, to Machine Learning (ML)-based methodologies is imperative for realizing the vision of safer transportation.

Although the final performance of such systems justifies their utilization, it is crucial to consider the constraints associated with their design. Preceding their deployment, an offline learning phase, incurring substantial financial implications, becomes imperative. This process plays a vital role in the development of a model endowed with the ability to

make optimal decisions within predefined parameters or conditions. Illustrated in Figure 1, the training cost of state-of-the-art Machine Learning models exhibits a tendency to escalate monetarily over time. Hence, for industry stakeholders, a decrease in training duration would not only result in efficiency gains, but also manifest noteworthy enhancements in revenue maximization.

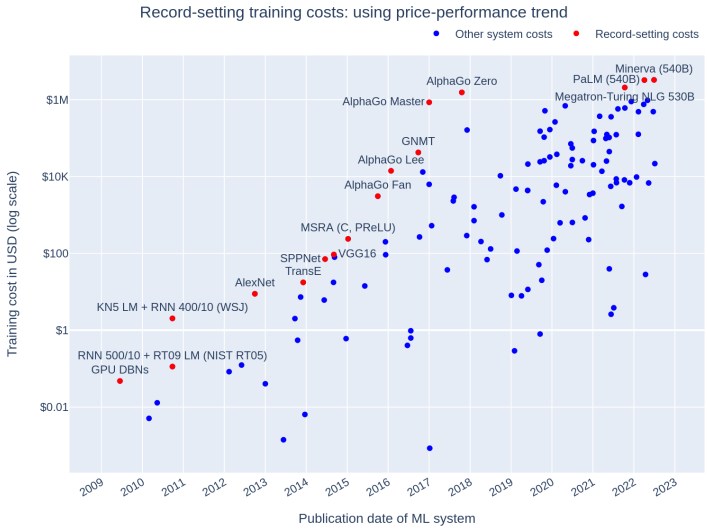

Figure 1. Training cost of Machine Learning systems expressed in *USD* on logarithmic scale [6].

The training process evolves into an iterative procedure due to the fine-tuning of diverse instructional hyperparameters. Monitoring the progress of individual models facilitates the determination of suitable values for these parameters. While this statement universally applies to Machine Learning, Reinforcement Learning introduces an additional challenge, namely the definition of a reward function that quantifies the success of a specific objective, thus indirectly leading the agent's behavior to the optimization goal. However, on the one hand, the achievable performance highly varies based on the selected rewarding criteria, as shown in Figure 2. On the other hand, articulating this function is inherently intricate; often, various physical attributes are interconnected heuristically, and the adaptability of trained agents to a specific environment becomes discernible through their behavioral manifestations.

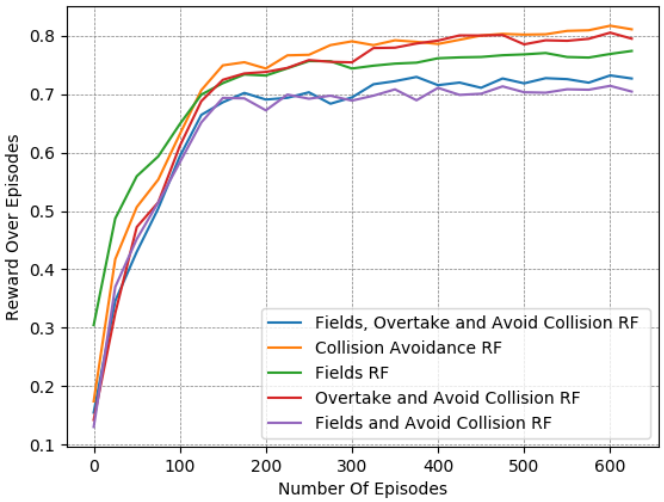

Figure 2. Training characteristics of different reward functions [7].

In order to attain the desired reduction in time and financial costs involved in model development, there arises a need to preliminarily measure the quality of the formulated

rewards. This ensures that only the most suitable concepts are utilized for training purposes. The formulation of this evaluation metric falls within the scope of the Monte Carlo Tree Search [8], which is an algorithm entirely devoid of explicit training that accomplishes the construction of an asymmetric tree through the execution of Monte Carlo-simulations. In real-time applications, sole reliance on the MCTS is evidently not advisable; nonetheless, it can offer a viable solution for the preliminary assessment of individual reward schemes.

The aim of this research is to compare various reward strategies by utilizing trained neural networks and the Monte Carlo Tree Search independently, thereby demonstrating that the judicious fusion of algorithms can lead to the minimization of surplus Reinforcement Learning training runs, thus resulting in a reduction in terms of training costs. Throughout the evaluation phase, an exhaustive analysis has been carried out to examine how distinct agents respond to modifications applied under identical state and action representations to eventually determine the most suitable strategy with regard to adaptability.

2. Related Work

In the latter part of the twentieth century, the widespread adoption of computing technology spurred the integration of numerical methods and Machine Learning techniques into a wide set of engineering application domains, such as medical analysis [9], fraud detection [10], robotics [11] and aerial vehicle localization [12]. This integration aimed to tackle tasks requiring human-like intelligence capable of making decisions based on learned experiences. As computational power advanced and demand grew, Artificial Intelligence (AI) emerged as a pivotal force capable of real-time interventions in vehicular motion.

This marks a significant milestone in AI's application, thus enriching capabilities across various sectors, most notably in automotive and transportation systems. The development of AlphaGo by DeepMind [13], which famously defeated the world champion in the game of Go in 2016, stands as a landmark achievement. It showcased the potential of Deep Reinforcement Learning methodologies in executing control tasks with a precision surpassing human capabilities, particularly when augmented by Monte Carlo Tree Search algorithms.

Beyond applications in board games like Go and chess, the Monte Carlo Tree Search algorithm has demonstrated remarkable success across various domains. For instance, in trajectory planning for robotics [14], a decentralized variant of the MCTS was introduced, thus allowing multiple robots to generate a joint distribution over trajectory plans within a joint action space and periodically updating it with each robot's decision trees.

In the realm of drone control environments, area scanning tasks were explored by [15] using both Reinforcement Learning and Monte Carlo Tree Search methods. In a grid world environment, the objective was to comprehensively survey a designated geographic area while minimizing the time required.

Addressing the multimode resource-constrained multiproject scheduling problem, ref. [16] proposed a solution to optimize large-scale real-world computational tasks. Their approach aimed to minimize the total time requirements of all projects, thus considering potential resource sharing.

In [17], the experimentation involved both plain and hybrid versions of the MCTS, with the latter integrating a classic Genetic Algorithm. This hybrid approach was applied to a structural engineering design problem, which was specifically optimizing load-bearing elements within a reference reinforced concrete structure while adhering to constraints and dynamic requirements.

Furthermore, ref. [18] conducted a survey covering the Vehicle Routing Problem (VRP) domain, including issues such as simultaneous planning and resource allocation for land vehicle operations in logistics, UAV delivery, and various VRP subfields like Green VRP, City VRP, and periodic VRP. The solutions employed hyperheuristics, Monte Carlo simulations, and other established methodologies within the field.

These diverse applications underscore the versatility of the Monte Carlo Tree Search beyond gaming scenarios. However, despite the breadth of research, integration with

Deep Reinforcement Learning methods remains an area warranting further exploration. For additional insights into alternative applications of the MCTS, refer to [19].

The utilization of the MCTS is not initially associated here with training the DQN. Its relevance was highlighted in a prior investigation focusing on urban traffic scenarios [20], thus laying the foundation for the current research objective. Here, the aim was to mitigate emissions and fuel consumption rates, thus prompting the exploration of integrating these two methodologies. Throughout the research, an examination was conducted on varying reward structures within the context of a single intersection that can be seen in Figure 3. Subsequent to the instructional phase, a hierarchy was formulated between rewards, thus paralleling the categorization ascertained through the implementation of the Monte Carlo Tree Search.

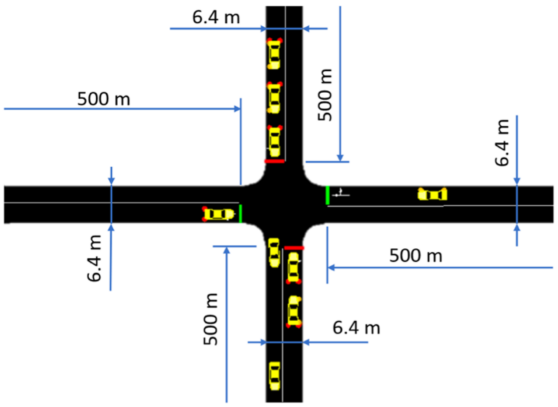

Figure 3. Geometric design of the junction in the previous research [20].

3. Contribution

While there is unanimous acknowledgment that the reward function, serving as an abstraction of the optimization objective, stands as a main element in Reinforcement Learning—as it is the sole source of feedback guiding the agent's behavioral adaptation throughout task resolution—a prevalent trend exists where researchers continually introduce novel metrics in pursuit of enhanced performance. However, the challenge of selecting the optimal function to minimize training costs remains insufficiently addressed, thus often relying on a trial-and-error approach.

Therefore, our contribution unfolds in three main aspects: Firstly, building upon our previous research [20], we empirically substantiate and validate our hypothesis within the domain of lane keeping using a kinematic bicycle model. Specifically, we demonstrate that the Monte Carlo Tree Search serves as a viable algorithm for evaluating the effectiveness of reward functions in Reinforcement Learning, without the need for training. This approach optimizes RL training processes in such a manner that, following the execution of Monte Carlo simulations, explicit training is exclusively requisite for the optimal reward function. In our previous study, a traffic signal control problem has been investigated; therefore, we demonstrate adaptability across environments of our method. Additionally, we showcase that our methodology also exhibits versatility across algorithms as, contrary to prior research utilizing a policy gradient method, this paper includes comparisons with a Q Learning-based algorithm. Secondly, we have trained Deep Q Network agents for this problem formulation employing five distinct rewarding strategies commonly found in the literature. Eventually, we have utilized the Monte Carlo Tree Search for the aforementioned task under identical conditions with the same rewarding strategies.

4. Environment

Numerous methodologies exist for the physical representation of vehicles, thus offering diverse models grounded in geometric, kinematic, and dynamic constraints [21]. When considering their application, careful consideration must be given to the velocity of the

vehicle's motion. At lower speeds, a kinematic description provides a sufficiently accurate approximation [22]. Nevertheless, for the evaluation of highway conditions or higher velocities, it becomes imperative to integrate and compute forces and torques influencing the vehicle's dynamics. In such scenarios, the adoption of dynamic vehicle models becomes indispensable for achieving a more precise representation and analysis.

Throughout both the training and evaluation phases, delivering steering interventions at a consistent velocity may impact the yaw motion of the vehicle. Given that both phases were conducted at low speeds, a kinematic bicycle model [23] has been employed, which is schematically shown in Figure 4 and mathematically described with the following equations:

$$\dot{x} = v\cos(\theta + \beta)$$
$$\dot{y} = v\sin(\theta + \beta)$$
$$\dot{\psi} = \frac{v\cos(\beta)}{l_r}\sin(\beta)$$
$$\dot{v} = a$$
$$\beta = tan^{-1}(\frac{l_r}{l_f + l_r} * tan(\delta_f))$$

where:

x: Vehicle's x coordinate
y: Vehicle's y coordinate
θ: Vehicle's heading angle
β: Side slip angle
ψ: Yaw angle
v: Vehicle's velocity
δ_f: Steering angle
a: Acceleration
l_f, l_r: Distance from the center of gravity to the front and rear axles

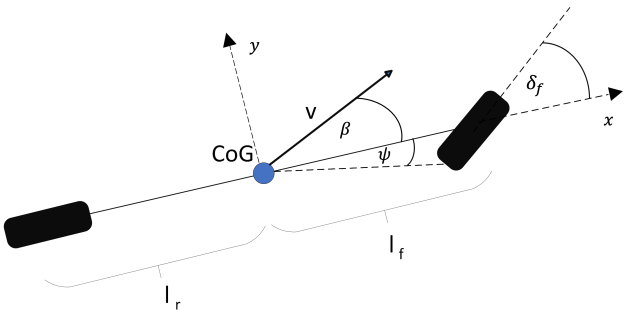

Figure 4. Kinematic bicycle model.

Once the vehicle model is established, the subsequent critical consideration involves track generation, where ensuring adequacy stands as a fundamental concern for successful training. By employing suitably randomized tracks, as shown in Figure 5, the issue of overfitting can be mitigated, thus facilitating the design of universally applicable trajectories. These trajectories are not solely tailored to enable the vehicle to navigate specific segments but also aim to ensure broader applicability. The generation of these tracks occurs along different seeds and is represented in the simulator characterized by the vehicle's center of mass. Additionally, the simulator specifies the maximum deviation; failure to maintain lane keeping within this limit results in track failure for that specific step.

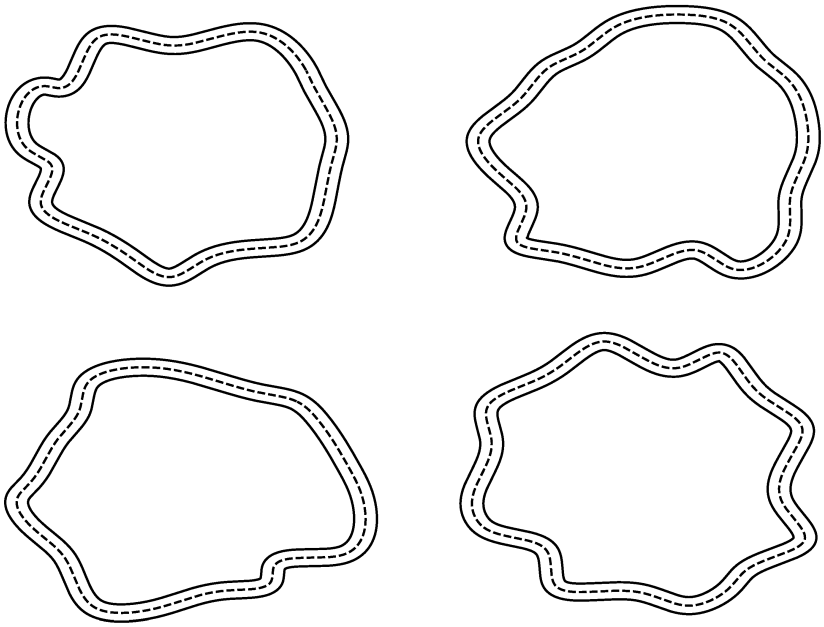

Figure 5. Randomly generated tracks [24].

Since the objective is to conduct a Reinforcement Learning task, the environment must feature an appropriate interface between the simulator and the agents. In this context, the widely adopted gym structure [25] is utilized due to its simplicity, standardization, and efficiency in training, thus necessitating only three fundamental functions for interobject communication.

4.1. State Representation

One of the three crucial abstractions concerning the agent's decision-making paradigm is the state representation, as it serves as a sole information source, based on which the agent is able to understand the inner dynamics of the given task. In the realm of lane keeping, this representation entails the current spatial orientation of the vehicle within the lane, which is expressed through both the distance from lane centerline d and the relative yaw angle ψ. Concurrently, the state representation is augmented with a desired number of lookahead sensory data (set to eight elements in our experiments) in the form of relative yaw angles. By computationally discerning relative yaw angles towards equidistant points along the forthcoming trajectory, this information yields indispensable orientation modifications, ensuring harmony with the desired trajectory. As a result, the vehicle's state manifests as a 10-element vector, thus serving as the neural network's input layer, as shown in Equation (1):

$$state_i = \left[|d_i| \ \psi \ \psi_1^* \ \psi_2^* \ \psi_3^* \ \psi_4^* \ \psi_5^* \ \psi_6^* \ \psi_7^* \ \psi_8^*\right]^T \tag{1}$$

4.2. Action Space

Actions can be realized through two modalities by utilizing either continuous or discrete decision spaces. Given deployment of the Deep Q Network algorithm, decision making is rendered by the neural network contingent upon a discrete decision space.

The domain of actions is defined as a three-element vector. Each vector component represents uniform differentials of the steering angle, which are manifested in both lateral directions—right and left—inclusive of an idling state. Given the specification of steering angle differentials, a network equipped with three output neurons is capable of covering the entire action space of the steering actuator in accordance with the given equal distribution.

In alignment with the mentioned factors, the action space is formulated as shown in Equation (2):

$$action = \begin{bmatrix} +0.1 \\ 0 \\ -0.1 \end{bmatrix} rad, \tag{2}$$

where each of the newly acquired steering angle values is determined at 100 ms intervals.

4.3. Rewarding Strategies

Determining the reward strategy stands as a core responsibility within Reinforcement Learning. Ascertaining which physical attributes should be employed, to enable our agent to comprehend the physics governing its motion, constitutes a complex endeavor. In the context of lane keeping, the majority of approaches rely on the position of the vehicle's center of gravity and orientation to define an appropriate reward function. Nonetheless, determining the exact metrics and their respective weighting often necessitates an empirical approach. The objective of this research is not the identification of a global optimum concerning reward functions but rather illustrating the combined application of the Monte Carlo Tree Search and Deep Q Network. Consequently, a selection of five distinct reward strategies are evaluated, thus guided by insights drawn from diverse scholarly references.

4.3.1. Sample Reward #1 [24]

This reward strategy penalizes the distance from the lane centerline (d), in addition to minimizing the degree of the vehicle's relative yaw angle (ψ), by applying a cosine function, as shown in Equation (3). It is conceivable that smaller absolute deviations and an orientation perfectly aligning with the lane centerline result in higher reward values.

$$R_1 = \begin{cases} cos(\psi) - |d| & \text{if} \quad R > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

4.3.2. Sample Reward #2 [26]

This reward function forms a construct analogous to the aforementioned reward, with the exception that it does not incorporate merely the orientation but also the decomposition of velocities into longitudinal and lateral components, as formulated in Equation (4).

$$R_2 = \begin{cases} -200 & \text{leaving the track} \\ v_x cos(\psi) - v_x sin(|\psi|) - v_x|d| & \text{otherwise} \end{cases} \tag{4}$$

4.3.3. Sample Reward #3 [26]

Although this reward has been introduced in a traffic junction scenario, where three possible outcomes may occur, collision avoidance is not part of the task, due to the presence of a single vehicle on the track. In this case, the λ and η weighting constants play a significant role in the rewarding strategy, as illustrated in Equation (5).

$$R_3 = \begin{cases} p_c & \text{collision} \\ p_0 & \text{leaving the track} \\ (v_x cos(\psi) - \lambda|d|) \times \eta & \text{otherwise} \end{cases} \tag{5}$$

4.3.4. Sample Reward #4

The final two reward functions share a similar objective: both motivate the agent for remaining within the designated lane. The distinction lies in the implementation, wherein the second scenario not only recognizes the absence of reward, such as the one shown in Equation (6), but also imposes a penalty value for deviating from the track, as described in Equation (7).

$$R_4 = \begin{cases} 0 & \text{leaving the track} \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

4.3.5. Sample Reward #5

$$R_5 = \begin{cases} -1 & \text{leaving the track} \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

5. Methodology

In the realm of control system engineering, an abundant array of alternative solutions emerges for addressing control tasks. These solutions encompass strategies rooted in the principles of modern control theory, such as Model Predictive Control (MPC). Nevertheless, complicating factors, such as computational latency and challenges in providing detailed physical system descriptions, often justify the adoption of methodologies derived from soft computing paradigms. Within the domain of soft computing, there is a gradual integration of autonomous vehicles with models derived from Machine Learning algorithms, particularly in sensory data processing. Furthermore, there is an emergent requirement to accomplish tasks of specific nature using ML strategies. This necessity primarily arises in scenarios where the computational demand of these tasks surpass the capabilities offered by traditional approaches, thus rendering them impractical due to their intensive computational capacity needs.

Given the above-mentioned computational demands, Deep Learning-based solutions come into consideration for addressing tasks related to behavior and decision making. This ensures real-time applicability of a system, thereby owing to reduced requirements for online computation. This advancement is not only evident in traditional control theory realizations but also in Machine Learning algorithms, such as tree search-based decision making. However, the computational time of these methods depends on task complexity, thereby limiting their effectiveness, especially in scenarios lacking real-time decision-making capabilities, which constitutes a drawback of, for instance, Model Predictive Control.

As a consequence of this limitation and the discrete decision space, we addressed the problem using two distinct approaches: a tree search algorithm, being the Monte Carlo Tree Search, representing a conventional Machine Learning technique known for its ability to determine the global optimum given an infinite amount of iterations and a Deep Q Network agent, which is a value-based Reinforcement Learning method suitable for rapid real-time application, where output neurons correspond to each legal decision from a given state. However, it is important to note that the DQN lacks an explicit mathematical guarantee regarding the quality of its decisions.

5.1. Reinforcement Learning

Reinforcement Learning [27] deviates significantly from the other branches of Machine Learning, as network tuning is not based on prearranged training samples, but rather on a so-called agent's own experiences. Its application enables the acquisition of complex behavioral patterns, thereby covering a distinct area within technical sciences compared to Supervised and Unsupervised Learning [28]. While not directly related to the specific issue at hand, one advantage of RL lies in its ability to decompose the agent's tasks into various subtasks within a multiagent system, thereby effectively resolving the problem of complex environments.

The fundamental concept of RL is the interaction between two entities, where an agent seeks to determine an optimal sequence of actions within an environment by continuously assessing and evaluating individual decisions [29], as illustrated in Figure 6. Each decision made is evaluated by the environment, which updates the state and provides feedback, along with a numerical representation of decision quality known as the reward [28]. Obtaining optimal decision-making patterns occurs through the maximization of the cumulative weighted reward, with an increasing trend indicating successful training that converges towards a value representing the environment's limit. The mathematical formalization of the cumulative weighted reward is shown in Equation (8) as

$$G_t = \sum_{t=0}^{T} \gamma^t \cdot r_t \tag{8}$$

where G_t is the cumulative weighted reward at time step t, T is the time horizon through which the agent is planning, γ is the discount factor determining the importance of immediate versus future rewards, and r_t is the immediate reward received at time step t.

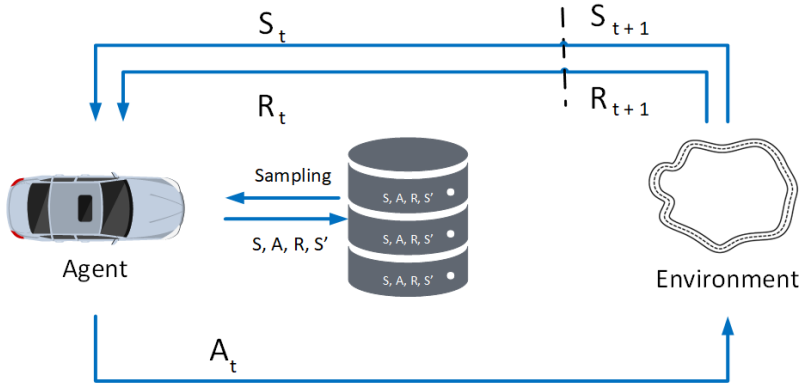

Figure 6. Reinforcement Learning training loop.

In the field of RL, state values determine the favorability of certain states. These values estimate how profitable it is for an agent to follow a certain policy in the long term. The concept of state value can be articulated as expressed in Equation (9):

$$V_\pi(s) = \mathbf{E}_\pi[G_t | S_t = s] \tag{9}$$

where V is the value function, π is the policy the agent would need to follow after time step t, s denotes the state, $\mathbf{E}$ is the expected value operator, and G_t denotes the cumulative weighted reward.

The quality of decision making emerges along state transitions, which can be described with the state–action value function, also known as the Q function. These state transitions in many cases are stored in a memory and used afterwards for parameter adjustment of a neural network via stochastic sampling from a uniform distribution. The Q function is formulated as shown in Equation (10):

$$Q_\pi(s, a) = \mathbf{E}_\pi[G_t | S_t = s, A_t = a] \tag{10}$$

where Q is the state–action value function, π is the policy the agent would need to follow after time step t, s is the state, a denotes the action, $\mathbf{E}$ means the expected value operator, and G_t is the cumulative weighted reward.

Consequently, the mathematical framework of Reinforcement Learning, rooted in Markov Decision Processes [28], encompasses the state, an action from the set of legal actions, the probability of transitioning from state s_t to s_{t+1} via action a_t, and the reward value. The global mathematical goal of the decision maker is to maximize its cumulative reward G_t, as shown in Equation (8).

Given the dynamically generated training samples throughout the learning process, engaging in exploratory actions becomes imperative. While forming part of the learning strategy, these actions play a crucial role in maximizing potential rewards. This process facilitates the discovery and comprehension of the ideal decision set that would yield the highest outcomes in certain situations. The balance between exploration and exploitation is determined by the ϵ-greedy policy as a standard approach in this field. This policy transitions over time from complete exploration to the process of making decisions deemed fully optimal according to the agent as the system encounters more and more training samples.

Following a concise introduction of fundamental concepts, it is relevant to consider the utilization of tools. In addition to conventional methods, such as tree search-based algorithms, soft computing also deserve consideration. Nonetheless, as selection of the appropriate algorithm highly depends on the task's inherent nature and complexity, thorough analysis is warranted.

Deep Q Network [30]

Concerning Reinforcement Learning, two primary avenues exist: value-based and policy-based algorithms. While policy-based algorithms directly acquire the desired behavior, value-based algorithms utilize a value function to evaluate the required actions in a given scenario. The advantages of these approaches can be effectively merged through actor–critic algorithms [31], which incorporate both value-based and policy-based networks.

Q Learning represents a value-centric approach as a subcategory in value-based Reinforcement Learning that employs either tabular or Deep Learning techniques to construct a value function, thus discerning advantageous actions from detrimental ones within specific contexts. In this framework, so-called Q values are assigned to each state transition, which are defined by an update equation that considers the current state and the scenario, thus yielding the maximum achievable Q value from the potential next states. The update step is carried out following the Bellman equation, as shown in Equation (11).

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)] \tag{11}$$

where $Q(s,a)$ is the Q value associated with the state transition from state s via action a, α is the learning rate, and γ denotes the discount factor to lower the relevant weights of rewards realized in future.

In this research, the so-called Deep Q Network algorithm has been utilized as a specific training method, which incorporates principles of Q Learning methods but additionally utilizes a double-component neural network architecture to interact with the value calculated from the Bellman equation. Notably, among these interactions, only the reward contributes numerically to the outcome, thereby maintaining a close relationship with the resulting Q value, while the γ and α constants play a weighting role within the equation. Additionally, the necessity for a second network arises from the limitation that subsequent state transitions inherently include correlation. To address this aspect, a secondary, target network is utilized. Of the two algorithms implemented, this will be the primary one utilized for vehicle control. However, it is evident that training parameters must be determined empirically. To mitigate this limitation and complement the DQN, the Monte Carlo Tree Search is being introduced in the following sections.

5.2. Tree Search Algorithms

The general idea behind tree search-based planning agents involves the construction of viable state transition sequences in the form of a graph theory-motivated abstract tree representation, where the current state acts as the root of the tree. The main parameter, namely the branching factor, is determined by the number of legal actions from a given state and intricately shapes the complexity of the search problem. Following a predetermined number of iterations, a decision is made from the root, thus relying solely on the values derived from the initial layer.

Tree search algorithms exhibit notable properties. They provide a systematic approach to exploring the solution space, thus allowing for a comprehensive evaluation of potential outcomes. These algorithms are adaptable to various problem domains and can accommodate diverse action spaces and state representations. Additionally, their ability to balance exploration and exploitation is advantageous in situations where an optimal solution needs to be identified amid uncertainty.

However, inherent drawbacks exist. Uninformed search methods, though capable of delivering optimal solutions given sufficient planning, often encounter challenges due to

resource-intensive computational requirements. This limitation can hinder their applicability in real-world scenarios where computational resources are constrained.

Conversely, methods leveraging heuristics offer a pragmatic means to guide the search, thus making them computationally more efficient. Yet, the trade-off is the absence of guarantees, as heuristic-based approaches may not always ensure optimal solutions and can be sensitive to the quality of the heuristic function.

Monte Carlo Tree Search

Striking a balance between exploration and exploitation on the one hand and addressing computational demands to a certain extent on the other, the Monte Carlo Tree Search offers a proper trade-off by integrating precision inherent in tree search methodologies with expansive generalization capacity exhibited by Monte Carlo sampling techniques.

The MCTS operates through an iterative process, as shown in Figure 7, which is characterized by stages of selection, expansion, simulation, and backpropagation. The algorithm dynamically constructs a decision tree by iteratively exploring and expanding nodes based on a designated heuristic selection policy. Subsequently, random simulations estimate the value of unexplored nodes, and the obtained values are systematically backpropagated through the tree, thus refining the information at each visited node.

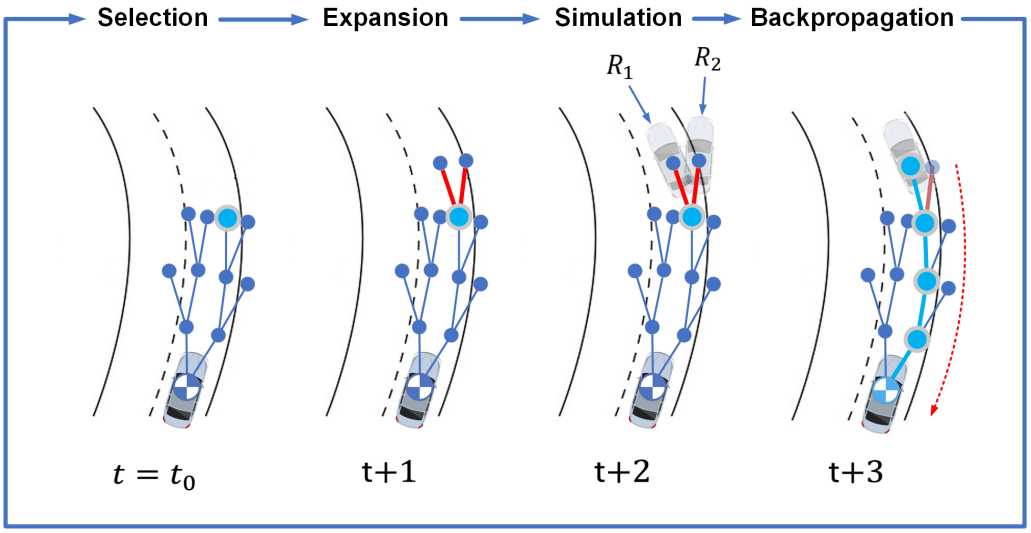

Figure 7. Monte Carlo Tree Search planning iteration.

In the realm of Reinforcement Learning, a prominent approach for achieving a balance between exploration and exploitation is the Upper Confidence Bound (UCB) algorithm, which utilizes uncertainty estimates to guide decision making, thus assigning confidence bounds to the estimated values of each action. This enables the agent to prioritize actions with higher potential, as expressed in the formula detailed in Equation (12):

$$UCB(i) = \frac{Q(i)}{N(i)} + C \cdot \sqrt{\frac{ln(N(p))}{N(i)}} \tag{12}$$

where $UCB(i)$ is the value of node i, $Q(i)$ is the total simulated reward of node i, $N(i)$ is the number of times node i has been visited, C is a constant controlling balance between the exploration and exploitation terms, and $N(p)$ is the number of times the parent node p has been visited.

In the context of Monte Carlo Tree Search applications, a modified version of the UCB method, known as the Upper Confidence Bound for Trees (UCT), is employed most commonly as a selection policy. This variant manages to achieve a delicate equilibrium between maximizing cumulative reward and acquiring valuable information that controls the growth of the tree towards an asymmetrical configuration, thus resulting in a notable

reduction in computational costs, time requirements, and memory usage when compared to alternative search algorithms. The UCT value utilized for identifying the most promising child node is delineated in Equation (13) as

$$UCT(i) = \bar{x_i} + 2 \cdot c \cdot \sqrt{\frac{2 \cdot ln\, N}{n_i}} \tag{13}$$

where $UCT(i)$ is the upper confidence bound value of node i, $\bar{x_i}$ is the average simulated reward of node i, N is the number of times the parent of node i has been visited, and n_i is the number of times node i has been visited.

While the Monte Carlo Tree Search demonstrates commendable performance, particularly in complex environments, it is crucial to recognize its limitations compared to Reinforcement Learning-based alternatives. Reinforcement Learning models, leveraging Deep Learning paradigms, often surpass the MCTS in terms of scalability. Furthermore, the reliance of the MCTS on Monte Carlo sampling may lead to substantial computational costs in scenarios characterized by extensive state spaces. Although theoretically capable of identifying the global optimum in search problems given an infinite amount of computational power, practical constraints preclude its real-time applicability in domains such as numerous tasks in the field of autonomous vehicle control. In essence, the exponential growth in tree complexity with the extension of the applied time horizon renders the MCTS unsuitable for such applications.

5.3. Monte Carlo Tree Seach for Reward Function Evaluation

While the Monte Carlo Tree Search algorithm is typically unsuitable for immediate real-time decision making due to its computational complexity, the possibility of its indirect utility via reward function evaluation lies in its ability to converge to the global optimum.

The rewarding mechanism stands as a pivotal component within the framework of Reinforcement Learning, as it constitutes an exclusive channel through which the agent understands the outcomes of its actions. Consequently, the judicious selection of rewards holds the key to the efficacy of the training process. Given the nontrivial nature of formulating the rewarding concept, originally the researcher's intuition exerts a profound impact on the attainable performance outcomes.

However, with the help of UCT value formalization and Monte Carlo sampling, the MCTS is able to predict the quality of a rewarding strategy, thus comparing them in terms of performance and selecting the most suitable one without having to train a Reinforcement Learning model for each reward function through a trial-and-error strategy. The mathematical formalization of the UCT value is expressed in Equation (14) as

$$UCT^*(s_{t,i}) = \bar{r}(s_{t+1}) + 2 \cdot c \cdot \sqrt{\frac{2 \cdot ln\, N}{n_s}} \tag{14}$$

where $UCT^*(s_{t,i})$ is the UCT value associated with future state s_i at time step t, $\bar{r}$ denotes the mean reward received for state transitions initiated from state s_i at time step t, c is the constant controlling the balance between the exploration and exploitation terms, N is the number of times the parent node of state $s_{t,i}$ has been visited, and n is the number of times state $s_{t,i}$ has been visited. Eventually, a summary schematic diagram highlighting the distinctions between pipelines of the traditional approach and our methodology is depicted in Figure 8.

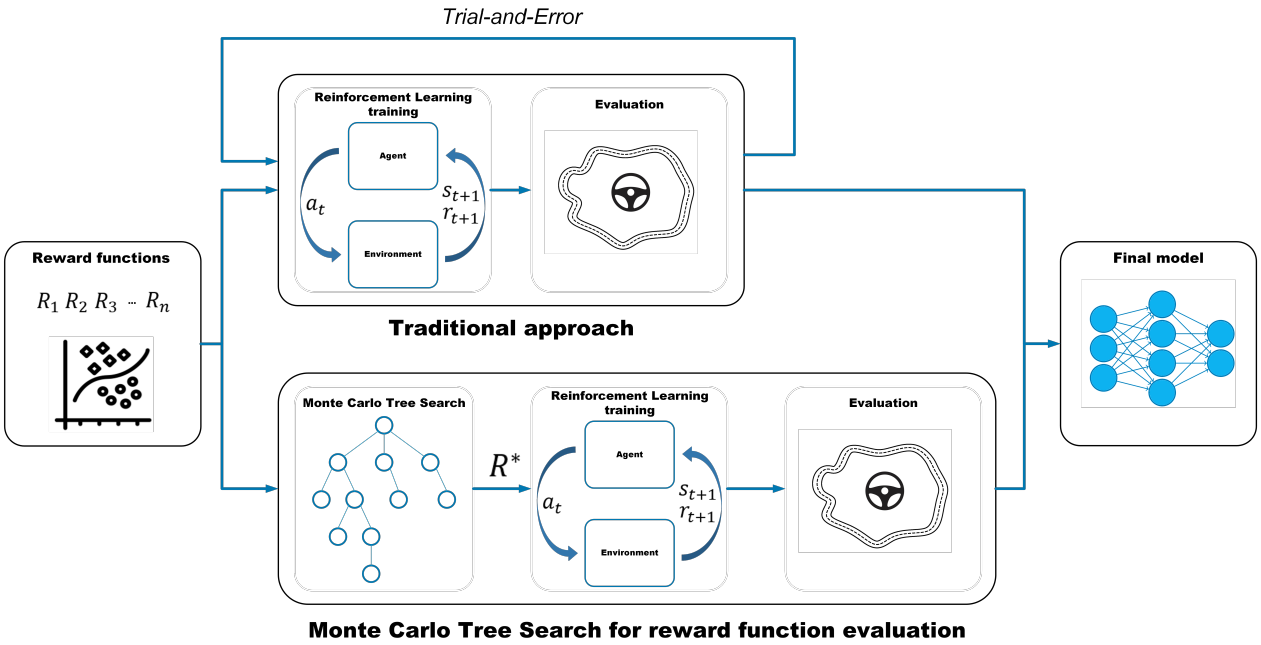

Figure 8. Schematic design of Monte Carlo Tree Search for reward function evaluation in Reinforcement Learning.

6. Results

As determined during the initial motivation phase, the research objective was to contrast various reward structures concerning Deep Q Network and Monte Carlo Tree Search agents. The consistency of adaptability across both methods suggests the viability of backing training with an initial evaluation of reward performance. Expanding on previous discoveries, the lane keeping issue was utilized to demonstrate wider applicability. Within this study's framework, the adaptability of the generated agents was demonstrated through alterations in environmental parameters, with an assessment of individual agent quality focused on trajectory feasibility.

Before presenting the new findings, the suitability in traffic situations is demonstrated through a brief discussion of the previous results. The prevailing guideline in modern traffic management design focuses on energy-efficient implementation and emission reduction. Thus, the aim of this previous study has been to enhance these indicators and increase the average speed, thereby achieving a larger traffic flow. As depicted in Figure 9 and in Table 1, the integration of these methodologies showcased advancements by constructing the same performance hierarchy of reward functions among the MCTS and PG agents, while taking mean values of 1000 consecutive seeded training runs on the demonstrated environmental configuration, as illustrated in Figure 3. Regarding all the sustainability parameters, Reward 1 exhibited the highest performance, followed by Rewards 2, 3, and 4 (the numbering of rewards utilized in the current context differs from the original paper). For a comprehensive explanation of the distinct reward strategies employed, refer to [20].

The study in this paper examined the effectiveness of agents trained with various reward structures compared to simulations with the Monte Carlo Tree Search in the case of increasing the longitudinal velocity of the kinematic model. Owing to the increased longitudinal velocity and consistent time step interval, this experiment evaluated the agents' ability to utilize current state information to anticipate future outcomes, thereby avoiding contact with lane edges. Both visual aids (Table 2 and Figure 10) present the mean values of discrete time steps taken by an agent before encountering a lane limit, thus leading to the termination of the episode. To ensure a fair comparison, both sets of tracks were generated using the same seed values in each instance, and the same initial positions have been set. Furthermore, the same hyperparameter set and iteration number (for the MCTS) has been

utilized among every given rewarding strategy. This approach enabled the incorporation of a comparative sample consisting of 1000 distinct, randomly generated tracks, thus allowing for an examination of the average number of steps taken by different agents.

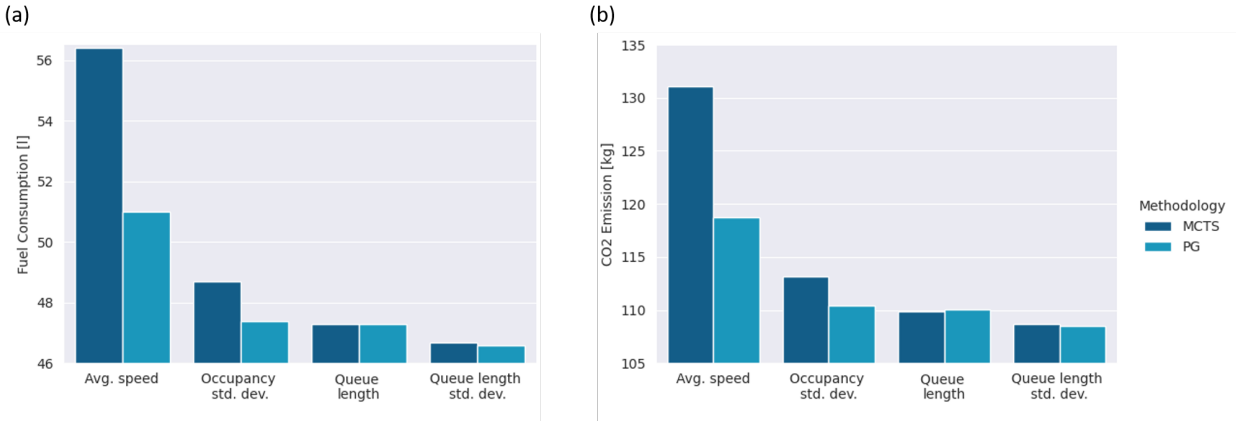

Figure 9. Sustainability metric results gathered on a single-traffic intersection scenario: (**a**) Fuel consumption and (**b**) CO_2 emission.

Table 1. Statistical comparison of sustainability measures on the Traffic Signal Control problem.

Agents	CO_2 Emission [kg]	NO_x Emission [g]	Fuel Consumption [L]
PG—Reward 1	108.5	47.2	46.6
PG—Reward 2	110.1	48.0	47.3
PG—Reward 3	110.4	48.1	47.4
PG—Reward 4	118.7	52.0	51.0
MCTS—Reward 1	108.7	47.3	46.7
MCTS—Reward 2	109.9	47.3	47.3
MCTS—Reward 3	113.2	49.4	48.7
MCTS—Reward 4	131.1	57.7	56.4

An analysis of the results, presented in Table 2 and in Figure 10, reveals that the generated order is identical in both scenarios. Specifically, both MCTS and DQN agents attained optimal results utilizing Reward #3, followed by a sequence of decreasing performance trends employing Reward #2, Reward #1, and Reward #4, and the worst performing agents have been trained on this task applying Reward #5. It confirms that extensive training can effectively address this issue through preliminary evaluation of the reward strategies. As a result, as the MCTS is able to transform the prior need of multiple trainings due to the diverse set of rewarding strategies concerning the initial RL problem to a planning task, thereafter only the most suitable reward function needs to be employed in the sole Reinforcement Learning training run, which is contrary to the traditional approach. In practical implementations, the application of this method allows the MCTS to minimize the required training sessions from n to 1, where n represents the number of eligible reward functions considered in the experiment, thereby optimizing the process to obtain the best neural network model with Reinforcement Learning efficiently.

Furthermore, as depicted in Figure 11, it is visually apparent that within the observed trajectories, certain agents outperform others in their ability to maximize future rewards, even if they occasionally deviate from the lane centerline. This suggests that despite receiving the same lookahead point information, these exceptional agents demonstrate superior foresight by effectively managing future rewards through their predicted action sequences.

Moreover, the generated search tree of the MCTS-based agent is shown in Figure 12 along a curvature track segment. Here, the optimal trajectory arc, converging to the lane centerline, is evident through the expansion of future states in the planning space.

The intensity of transition colors over nodes corresponds to the visit count n_i, with darker shades indicating more frequent visits. Thus, it becomes apparent how the initial position, orientation of the vehicle, and environmental constraints collectively shape the optimal trajectory, which are guided by a specific reward function, as evidenced by the darkest state transition sequence.

Table 2. Average steps per episode along 1000 seeded evaluation runs on the task of lane keeping.

Agents	Deep *Q* Network	Monte Carlo Tree Search
Reward #3 (Section 4.3.3)	919.9	460.9
Reward #2 (Section 4.3.2)	822.1	415.0
Reward #1 (Section 4.3.1)	735.5	236.7
Reward #4 (Section 4.3.4)	677.1	126.6
Reward #5 (Section 4.3.5)	593.3	71.6

Figure 10. Comparison of average steps per episode along 1000 seeded evaluation runs on the task of lane keeping.

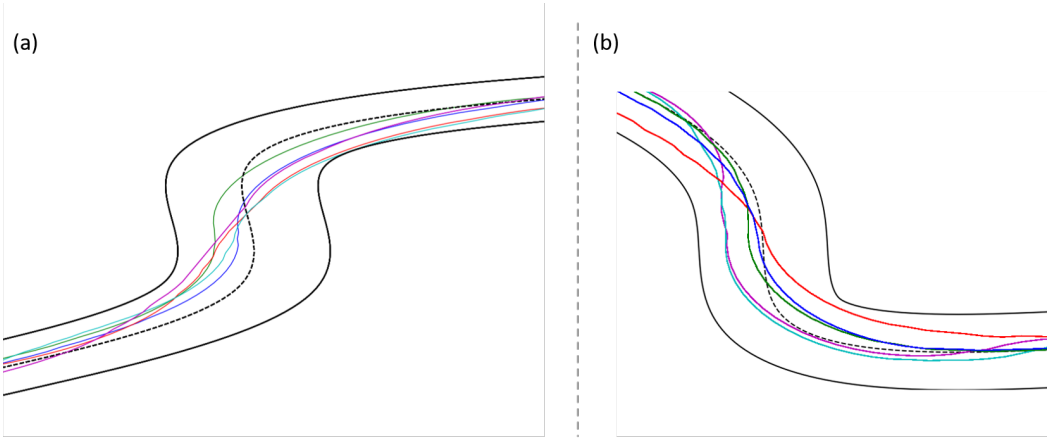

Figure 11. Trajectories realized by agents based on different methodologies following 5 distinct reward strategies: (**a**) Deep *Q* Network and (**b**) Monte Carlo Tree Search.

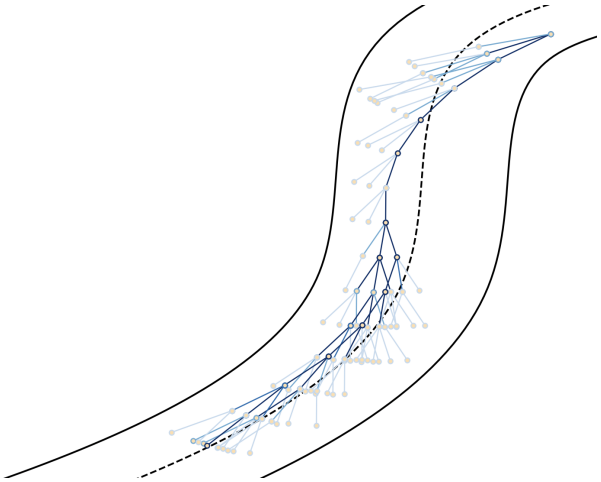

Figure 12. MCTS-generated asymmetrical search tree along a curve.

7. Conclusions

This paper provides solutions for the task of lane keeping within the context of a kinematic bicycle model. The objective for the agents is to determine the optimal steering action sequence that simultaneously optimizes the track by leveraging a given reward strategy. Two distinct methods were employed to tackle the problem: a Deep Q Network agent was trained to apply five different rewarding concepts to address the control task, and the same reward functions were utilized by the Monte Carlo Tree Search algorithm. The employed reward functions, detailed in Section 4.3, are based on the most commonly used strategies found in the literature. In addition to the new application environment, the results also show that MCTS is not only a useful tool for PG, but since it has been used in this research as an aid to the DQN, it can also be concluded that the MCTS algorithm can help other RL algorithms to be more effective.

As defined in the motivation, the objective of this research was to apply the MCTS to an additional problem formulation, analyze its performance over the investigated reward functions, and compare the resulting metrics, thereby supporting our previous theoretical hypothesis and showcasing that it can mitigate training resources not only for a specific task but, being a generalizable methodology avoiding the traditional trial-and-error approach in RL for defining the appropriate reward functions, it can be applied regardless of the problem at hand. As introduced formerly in a Traffic Signal Control problem and in vehicle trajectory planning at this time, an assessment can be carried out through preliminary simulations to determine which strategy yields an agent for a given application such that, while optimally adapting to environmental changes, it still designs feasible trajectories. With the aid of Deep Learning, real-time decision making and consequent actuations can be realized in the physical system.

In conclusion, the study highlights a unique and valuable attribute of the Monte Carlo Tree Search algorithm. Particularly, the MCTS proves adept at comparing and prioritizing various rewarding strategies, thereby notably shortening the time needed to select the optimal reward for training Reinforcement Learning agents and eliminating the necessity for excessive training iterations. This efficiency leads to a significant decrease in resource utilization throughout the entire process.

In future endeavors, we intend to evaluate the robustness of this aspect of the Monte Carlo Tree Search algorithm across a range of sequential decision-making problems to ascertain its reliability and potential power in reward function evaluation. Moreover, when employing the MCTS as a tool for this purpose, it appears reasonable to consider establishing a threshold for the number of tree search iteration steps necessary for accurate outcomes. Additionally, our investigation will extend to address the Traffic Signal Control problem within a more intricate setup involving multiple interconnected intersections as a formulation of Multiagent Reinforcement Learning.

Author Contributions: Conceptualization, T.B. and B.K.; methodology, B.K. and B.P.; software, B.P. and I.G.K.; validation, T.B. and B.K.; investigation, T.B. and B.K.; resources, T.B.; writing—original draft preparation, B.P. and I.G.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems (RRF-2.3.1-21-2022-00002). The research reported in this paper is part of project no. BME-NVA-02, which has been implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development, and Innovation Fund, which has been financed under the TKP2021 funding scheme. T.B. was supported by BO/00233/21/6: the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data and source are available at the authors by request.

Conflicts of Interest: Author Bálint Kővári was employed by the company Asura Technologies Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Szántó, M.; Kobál, S.; Vajta, L.; Horváth, V.G.; Lógó, J.M.; Barsi, Á. Building Maps Using Monocular Image-feeds from Windshield-mounted Cameras in a Simulator Environment. *Period. Polytech. Civ. Eng.* **2023**, *67*, 457–472. [CrossRef]
2. Bimbraw, K. Autonomous Cars: Past, Present and Future—A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology. In Proceedings of the 2015 12th international conference on informatics in control, automation and robotics (ICINCO), Colmar, France, 21–23 July 2015; Volume 1, pp. 191–198. [CrossRef]
3. Fehér, Á.; Aradi, S.; Bécsi, T. Fast prototype framework for deep reinforcement learning-based trajectory planner. *Period. Polytech. Transp. Eng.* **2020**, *48*, 307–312. [CrossRef]
4. Aradi, S.; Becsi, T.; Gaspar, P. Policy gradient based reinforcement learning approach for autonomous highway driving. In Proceedings of the 2018 IEEE Conference on Control Technology and Applications (CCTA), Copenhagen, Denmark, 21–24 August 2018; IEEE: New York, NY, USA, 2018; pp. 670–675.
5. Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; Pedreschi, D. A survey of methods for explaining black box models. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–42. [CrossRef]
6. Cottier, B. Trends in the Dollar Training Cost of Machine Learning Systems. 2023. Available online: https://epochai.org/blog/trends-in-the-dollar-training-cost-of-machine-learning-systems (accessed on 2 March 2024).
7. Karalakou, A.; Troullinos, D.; Chalkiadakis, G.; Papageorgiou, M. Deep Reinforcement Learning Reward Function Design for Autonomous Driving in Lane-Free Traffic. *Systems* **2023**, *11*, 134. [CrossRef]
8. Chaslot, G.M.J.B.C. Monte-Carlo Tree Search. Ph.D. Thesis, Maastricht University, Maastricht, The Netherlands, 2010. [CrossRef]
9. Gao, Y.; Lin, J.; Zhou, Y.; Lin, R. The application of traditional machine learning and deep learning techniques in mammography: A review. *Front. Oncol.* **2023**, *13*, 1213045. [CrossRef] [PubMed]
10. Han, W.; Mehta, V. Fake news detection in social networks using machine learning and deep learning: Performance evaluation. In Proceedings of the 2019 IEEE International Conference on Industrial Internet (ICII), Orlando, FL, USA, 11–12 November 2019; IEEE: New York, NY, USA, 2019; pp. 375–380.
11. Soori, M.; Arezoo, B.; Dastres, R. Artificial intelligence, machine learning and deep learning in advanced robotics, a review. *Cogn. Robot.* **2023**, *3*, 54–70. [CrossRef]
12. Sandamini, C.; Maduranga, M.W.P.; Tilwari, V.; Yahaya, J.; Qamar, F.; Nguyen, Q.N.; Ibrahim, S.R.A. A review of indoor positioning systems for UAV localization with machine learning algorithms. *Electronics* **2023**, *12*, 1533. [CrossRef]
13. Wang, F.Y.; Zhang, J.J.; Zheng, X.; Wang, X.; Yuan, Y.; Dai, X.; Zhang, J.; Yang, L. Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond. *IEEE/CAA J. Autom. Sin.* **2016**, *3*, 113–120. [CrossRef]
14. Best, G.; Cliff, O.M.; Patten, T.; Mettu, R.R.; Fitch, R. Dec-MCTS: Decentralized planning for multi-robot active perception. *Int. J. Robot. Res.* **2019**, *38*, 316–337. [CrossRef]
15. Sándor, H.; Bálint, K.; Máté, K.; Tamás, G.; Dániel, V.; József, R.; György, B. Area Scanning with Reinforcement Learning and MCTS in Smart City Applications. *Repüléstudományi Közlemények* **2020**, *32*, 137–153.
16. Asta, S.; Karapetyan, D.; Kheiri, A.; Özcan, E.; Parkes, A.J. Combining Monte-Carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem. *Inf. Sci.* **2016**, *373*, 476–498. [CrossRef]
17. Rossi, L.; Winands, M.H.; Butenweg, C. Monte Carlo Tree Search as an intelligent search tool in structural design problems. *Eng. Comput.* **2022**, *38*, 3219–3236. [CrossRef]
18. Mańdziuk, J. New shades of the vehicle routing problem: Emerging problem formulations and computational intelligence solution methods. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *3*, 230–244. [CrossRef]

19. Świechowski, M.; Godlewski, K.; Sawicki, B.; Mańdziuk, J. Monte Carlo tree search: A review of recent modifications and applications. *Artif. Intell. Rev.* **2023**, *56*, 2497–2562. [CrossRef]
20. Kövári, B.; Pelenczei, B.; Bécsi, T. Monte Carlo Tree Search to Compare Reward Functions for Reinforcement Learning. In Proceedings of the 2022 IEEE 16th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 25–28 May 2022; IEEE: New York, NY, USA, 2022; pp. 123–128.
21. Kong, J.; Pfeiffer, M.; Schildbach, G.; Borrelli, F. Kinematic and dynamic vehicle models for autonomous driving control design. In Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, Republic of Korea, 28 June–1 July 2015; IEEE: New York, NY, USA, 2015; pp. 1094–1099.
22. Wang, J.; Yan, Y.; Zhang, K.; Chen, Y.; Cao, M.; Yin, G. Path planning on large curvature roads using driver-vehicle-road system based on the kinematic vehicle model. *IEEE Trans. Veh. Technol.* **2021**, *71*, 311–325. [CrossRef]
23. Smith, J.; Johnson, E. Kinematic Vehicle Models for Autonomous Driving. *J. Auton. Veh.* **2020**, *15*, 201–215.
24. Kővári, B.; Hegedüs, F.; Bécsi, T. Design of a Reinforcement Learning-Based Lane Keeping Planning Agent for Automated Vehicles. *Appl. Sci.* **2020**, *10*, 7171. [CrossRef]
25. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
26. Wu, Y.; Liao, S.; Liu, X.; Li, Z.; Lu, R. Deep reinforcement learning on autonomous driving policy with auxiliary critic network. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 3680–3690. [CrossRef]
27. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
28. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
29. Mahesh, B. Machine learning algorithms-a review. *Int. J. Sci. Res.* **2020**, *9*, 381–386.
30. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
31. Konda, V.; Tsitsiklis, J. Actor-critic algorithms. *Adv. Neural Inf. Process. Syst.* **1999**, *12*, 1008–1014.