Nonlinear Dyn (2024) 112:8363–8391
https://doi.org/10.1007/s11071-024-09509-8

ORIGINAL PAPER

Monte Carlo tree search control scheme for multibody dynamics applications

Yixuan Tang · Grzegorz Orzechowski · Aleš Prokop · Aki Mikkola

Received: 14 June 2023 / Accepted: 7 March 2024 / Published online: 3 April 2024

Abstract There is considerable interest in applying reinforcement learning (RL) to improve machine control across multiple industries, and the automotive industry is one of the prime examples. Monte Carlo Tree Search (MCTS) has emerged and proven powerful in decision-making games, even without understanding the rules. In this study, multibody system dynamics (MSD) control is first modeled as a Markov Decision Process and solved with Monte Carlo Tree Search. Based on randomized search space exploration, the MCTS framework builds a selective search tree by repeatedly applying a Monte Carlo rollout at each child node. However, without a library of available choices, deciding among the many possibilities for agent parameters can be intimidating. In addition, the MCTS poses a significant challenge for searching due to the large branching factor. This challenge is typically overcome by appropriate parameter design, search guiding, action reduction, parallelization, and early termination. To address these shortcomings, the overarching goal of this study is to provide needed insight into inverted pendulum controls via vanilla and modified MCTS agents, respectively. A series of reward functions are well-designed according to the control goal, which maps a specific distribution shape of reward bonus and guides the MCTS-based control to maintain the upright position. Numerical examples show that the reward-modified MCTS algorithms significantly improve the control performance and robustness of the default choice of a constant reward that constitutes the vanilla MCTS. The exponentially decaying reward functions perform better than the constant value or polynomial reward functions. Moreover, the exploitation vs. exploration trade-off and discount parameters are carefully tested. The study's results can guide the research of RL-based MSD users.

Keywords Monte Carlo Tree Search · Multibody dynamics · Reward functions · Parametric analysis · Artificial intelligence control · Inverted pendulum

Y. Tang (✉) · G. Orzechowski · A. Mikkola
Department of Mechanical Engineering, LUT University, 53850 Lappeenranta, Finland
e-mail: Yixuan.Tang@lut.fi; Yixuan.Tang@aliyun.com

A. Prokop
Faculty of Mechanical Engineering, Brno University of Technology, Technicka 2896/2, 616 69, Brno, Czech Republic

1 Introduction

Artificial intelligence (AI), often realized by applying machine learning (ML), has recently been attracting considerable interest and has been on the rise in academia and industry. The objective of this work is to use AI to control multibody system dynamics, such as inverted pendulum control, and improve the performance of classical AI controllers. The most common ML approach is to train learning algorithms with a lot of data to solve decision-making and prediction tasks, which has brought many groundbreaking innovations. Reinforcement learning (RL) [1] is one of the

main branches of ML. Depending on how the agents learn from the environment model to get a known transition probability, the RL algorithms can be grouped into model-free and model-based RL algorithms [2]. To maximize cumulative reward, the RL algorithms generate data through continuous interaction with the environment. Each interaction cycle in RL intends to find the optimal decision to take action. The agent in model-free RL algorithms uses the data sampled from experience and learns the value function and policy. Model-free RL algorithms are suitable when using the deep learning (DL) framework to sample large-scale data and conduct network training. Due to the large amount of data sampling required, however, the model-free RL algorithms are computationally intensive. At the same time, the agent in model-based RL algorithms samples the simulation data to interact with the environment model and plan value function and policy. Ideally, if the model is sufficiently accurate, a good policy can be learned from it [3]. However, a compounding error [4] will develop between the model and the actual environment.

The optimization process for RL algorithms primarily uses a mathematical framework known as the Markov decision process [5]. Monte Carlo Tree Search (MCTS) can be categorized as a model-based reinforcement learning approach, which involves planning for a sequential decision-making algorithm using standard Markov decision process logic. MCTS consists of searching combinatorial spaces represented by trees and Monte Carlo methods. Monte Carlo is one of the fundamental RL algorithms used to solve any Markov decision process problem [6]. Decision-making involves identifying and selecting the best alternative based on personal values and preferences. Decision-making processes are integral to games, involving players making strategic choices to achieve their objectives. In 2006, Rémi Coulomb [7] developed the MCTS algorithm in the 9×9 Go-chess-playing program, Crazy Stone. Before that, almost all Go programs used deterministic search algorithms. In 2017, the AlphaGoZero [8] (trained without human examples) defeated the strongest previous versions of AlphaGo [9] (trained with human expert data). MCTS was one of the main ideas behind the success of AlphaGo [9] and AlphaGoZero [8].

The MCTS is often combined with ML models to achieve great success in games. Notably, AlphaGo [9] combined MCTS with deep neural network is currently state-of-the-art for games in the class of chess and Go [10]. It has inspired many approaches for games sharing the same basic structure with a trained shallow value [11] and policy [12] networks. Moreover, MCTS also acted as a trainer for reinforcement learners to build ML models. Pinto and Coutinho [13] used MCTS as a possible options framework and prepared options for the video games. Kartal et al. [14] used MCTS as a demonstrator in hard-exploration domains because the negative rewards may have profound consequences acts.

In addition to its remarkable achievements in games, MCTS also attracted interest from real-world problems beyond games [15]. This study only focuses on the application of mechanics and control. Shen et al. [16] proposed the M-walk to overcome the sparse reward problem, where the agent comprises an MCTS and a deep neural network to generate trajectories with more positive rewards. Using the same cost function and system model in autonomous driving, the MCTS algorithm achieves a much lower cost than classical Model Predictive Control (MPC) [17]. However, the scope of the study [17] is limited, and more insight is needed to draw more general conclusions about MCTS characteristics as compared with other control schemes, including MPC. A novel perspective to use MCTS to reconstruct the state space from a single time series for detection and observation was proposed in [18]. The Monte Carlo simulations of the surrogate models provided highly accurate approximations and significantly reduced the computational cost of structural head models [19].

Although MCTS has proven powerful in sequential decision-making problems, it poses a significant challenge for searching due to the search tree's complexity [15,20]. Lacking computational power limits the potential of building an exhaustive search tree from the whole state space. The performances of MCTS can be significantly below algorithm capacity if the parameters are limitedly designed [21]. This can be clearly seen in Sect. 4.3, where multiple MCTS control examples are evaluated with varying parameters. In recent years, several MCTS modifications intended to tackle the decision-making problems in games and have made new achievements in the following aspects.

(1) In case of the tree is growing sideways with limited chances for the in-depth inspection of promising branches, the MCTS did action reduction modified

to narrow down the number of available actions [22,23].

(2) Real-time strategy is a challenge and hot-spot for RL algorithms due to considerable decision space and partial observability, which can benefit from methods of action reduction, early termination, sampling strategy [20] and parallel algorithm.

(3) Making the trade-off between exploitation and exploration is one of the challenges facing the use of RL algorithms that has been studied by mathematicians for many decades [24]. An exploitation and exploration trade-off depends mainly on the various tree policies with different parameters. Updating the policy parameters increased the strength of MCTS [25].

(4) To guide the search-based algorithm to a good solution, the reward function design is the key to solution evaluation problem [23]. Learning reward functions through deep neural networks has become popular in RL for complex tasks involving goals that are poorly defined or hard to specify [26]. In addition, the reward design avoiding the sparse reward problem [16] or reflecting the optimal reward problem [21] improved the performance of MCTS.

To overcome the limitations of the Monte Carlo Tree Search (MCTS) algorithm, which are mainly caused by the complexity and large number of branches in the tree, the paper examines the solutions proposed in points 2, 3, and 4 of the aforementioned list. After point 2, the early termination is done by limiting the time horizon of the simulation rollouts. After point 3, the paper explores the impact of the exploration parameter on the algorithm's performance, to study the trade-off between exploitation and exploration. After point 4, the MCTS algorithm is guided by several reward functions. However, the study only focuses on two specific actions, so the tree's sideways growth is not considered.

Computer technology can be used for the dynamic simulation of complex mechanical systems, and AI has also attracted much attention and seen rapid progress in multibody system dynamics (MSD). Therefore, reliable and efficient models are crucial in various MSD applications via AI technology. Angeli et al. [27] established a nonlinear map via DL to minimal rigid multibody coordinates with the Kalman filter to speed up the simulations. Peng et al. [28] established a general friction model via DL to determine the contact surface and characterize friction properties. In addition, the dynamics response predictions for rigid MSD [29] and flexible MSD [30] via DL successfully met the research needs for real-time. Although the development of AI in MSD applications is remarkable, little has been done to date to explore the MSD applications with MCTS. In MSD, decision-making processes usually involve setting a control objective and then using optimization techniques to find the control inputs that can either maximize or minimize this objective. The modifications made to MCTS in games can be applied to address decision-making challenges in MSD, such as making the right moves to keep an inverted pendulum upright. Efficient implementation of MSD numerical solutions requires time discretizing a set of continuous ordinary or partial differential equations. The motion of a rigid MSD in 2D or 3D can serve as a series of decision-making problems of the Markov decision process and apply MCTS to solve them. Thus, the discrete action space in MCTS can be naturally added to the dynamic equations due to the discrete time.

In this study, we consider the classical control MSD task of stabilizing inverted pendulums using vanilla and modified MCTS to reach the level of automatic drive. By limiting choices to left or right movements in the inverted pendulum search tree, the sideways tree growth challenge is eliminated. Making a balance of exploitation vs. exploration trade-off and manipulating the reward function can improve learning results, which are widely recognized in games and will also be discussed in MSD applications. The main contributions are summarized as follows.

(1) This marks the inaugural application of MCTS to multibody applications. While reinforcement learning typically favours along-sighted algorithms in virtual games [1], there is limited work on this concept in mechanical systems. In this mechanical study, the discount parameter incrementally grows from 0 to 1, revealing superior performance by long-sighted agents compared to short-sighted counterparts.

(2) Opting for minor exploitation and more exploration is expected to result in significant improvement in control when compared to other trade-off parameter combinations. Typically, maximized exploitation, focusing only on specific previous actions, is too greedy to utilize other chances for success. Conversely, maximizing exploration involves consid-

ering both actions simultaneously, creating another opportunity for success.

(3) A series of reward functions guiding the MCTS-based controller to maintain the upright position is designed, which significantly improve the control performance of the default constant reward used in the benchmark reinforcement learning cart pole environment [31].

(4) The exponential reward functions perform better than the polynomial reward functions. Among them, the angle penalty and tip displacement penalty were shown to control the multi-inverted pendulum well.

(5) The redesigned reward functions improve the control robustness of MCTS. With prolonged simulation time, the vanilla MCTS exhibits diminished control robustness, while the modified MCTS maintains stable performance.

The objective of this work is to identify the effects of parameters in the vanilla MCTS via the two inverted pendulum control systems and improve the vanilla MCTS through reward designs. Section 2 offers a detailed background on building an MCTS, which combines a tree search approach with MC simulations. A modified MCTS algorithm with several frameworks of reward functions is proposed, whose distribution shapes guide the search algorithm towards higher reward regions to control the inverted pendulums upright. The rigid MSD based on semi-recursive formulation is illustrated in Sect. 3. The effects of parameters in the vanilla MCTS and the improved performances in the modified MCTS are accordingly demonstrated via two numerical examples in Sect. 4. Section 5 describes the findings and future studies.

2 Monte Carlo tree search in control

The MCTS is a decision-making algorithm that combines a tree search approach with Monte Carlo simulations and uses the outcome of these simulations to evaluate states in a look-ahead decision tree. See Fig. 1 for details of such a combination. The Monte Carlo is one of the fundamental RL algorithms used to solve any Markov decision process problem [6]. And the Markov decision process, discussed in Sect. 2.1, is the logic behind the MCTS.

In the field of control, there are two key components that are interrelated: the controller (or agent) and the plant (or environment) being controlled. These terms are often used interchangeably based on the context. In control theory, the terms controller and plant are used to describe the components, while in the context of reinforcement learning, the terms agent and environment are used instead.

In Figs. 1 and 5, we use the terms controller and plant. However, in Fig. 2 and discussion in the reinforcement learning context, we refer to the agent and environment.

2.1 Markov decision process

The Markov decision process is the logic behind the MCTS. Modeling a control task as a Markov decision process is a key concept in RL. In a Markov decision process setting, the RL problem is formulated by the tuple $M = (S, A, R, P, \gamma)$ [24].

(1) S is the state space (also called the observation), which contains the information about the environment and the agent at a given moment.

(2) A is the action space, which defines the interactions between the agent and the environment.

(3) $R(s, a)$ is the reward function to be learned, which gives a numeric signal of goodness to the moves the agent chooses.

(4) $P \in [0\ 1]$ is the transition probability distribution function (mostly to be learned), which represents the probability of being in the next state s' after the agent performed action a in state s. $p = P(s'|s, a)$ is learned from the previous experience for the stochastic control system and typically follows the Boltzmann, uniform, or categorical distribution [10]. While $p = P(s, a) = 1$ is for the deterministic control, such a move will always be chosen and there is no need to learn. The sum of the probabilities across all possible next states is equal to 1, i.e., $\sum_s P(s'|s, a) = 1, \forall s \in S$, where $\forall s \in S$ represents for any $s \in S$ holds.

(5) $\gamma \in [0\ 1]$ is the discount factor, which adjusts the importance of rewards over time and how much the affections of the future are discounted. A discount rate close to 1, i.e., $\gamma \to 1$, means that the agent is long-sighted and future rewards are given similar weight to immediate rewards optimizing over the long-term. Whereas a low discount rate, i.e., $\gamma \to 0$, means that the agent is short-sighted and leads to preferring only short-term time horizons [10].

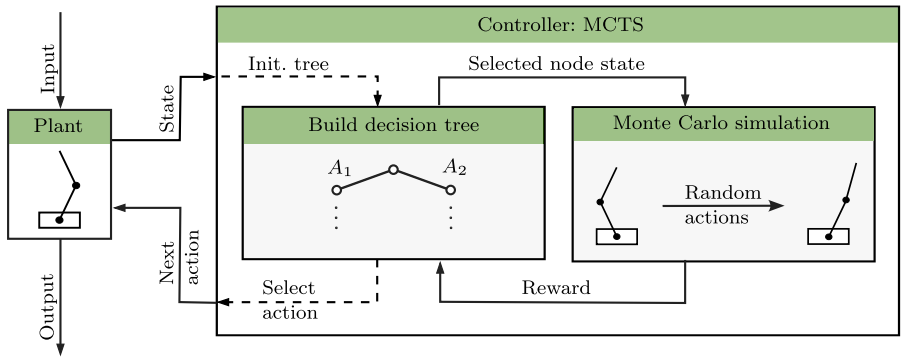


Fig. 1 One-time interval MCTS open-loop control of the plant. At each control time step, the current state of the plant initializes the tree. The plant takes the best-approximated action after the tree is built (typically when the computational budget is exhausted or the allotted time is reached). Internally, the MCTS controller expands the tree by selecting an appropriate node and performing a Monte Carlo simulation starting at the selected node's state. After the Monte Carlo rollout, the performance measure is returned as a reward. The controller is explained in details in Sect. 2.3 and Fig. 3

For a state $s \in S$, the agent takes a control action $a \in A$, then a reward $r = R(s, a)$ is received, and a next state s' is reached in the environment. In RL, s and a appear in pairs, that is $(S_t, A_t, S_{t+1}, A_{t+1}, \ldots)$. The subscript t represents the steps, and the corresponding reward is recorded as R_{t+1}. The probability of moving from the current state S_t to the next state S_{t+1} follows the so-called Markov property. This property says that the probability is independent of the history and regardless of all previous states or actions encountered,

$$P(S_{t+1}|S_t, A_t) = P(S_{t+1}|S_t, A_t, S_{t-1}, A_{t-1}, \ldots, S_0, A_0). \quad (1)$$

Within a Markov decision process, the current state contains enough information to choose optimal actions to maximize future rewards. However, models require previous experience to predict the next action, such as the autoregressive models. As a result, such models fail to meet the so-called Markov property.

Some important terms are introduced below to help understand the algorithms as follows.

(1) Cycle – A tuple of state, action, reward, and next state, such as $(S_t, A_t, R_{t+1}, S_{t+1})$, which is the interaction cycle between the agent and the environment. See Fig. 2. At each step, the agent observes the environment, picks an action, and then receives a new reward and state. And every step has an opportunity for learning and improving performance.

(2) Episode e [1] – An episodic task is a task that has an ending, either because the clock stops or because the agent reaches a terminal state. The episode is a sequence of cycles from the beginning to the end that describes a complete interaction process, such as

$$e : (S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots, A_{T-1}, R_T, S_T). \quad (2)$$

Therefore, there is a finite number of steps in an episodic task. Mostly, such interactions go on for several cycles in an episode. In a tree search plan, an agent may take several episodes, numbered as n_e, to learn to solve a task, especially in a pure Markov decision process, $n_e = 1$. The episode is always the first thing that the agent needs to generate.

(3) Search budget n – The maximum search computational cost of given. As in [32], the given search budget is divided into n_e episodes as

$$n = n_e H, \quad (3)$$

where H is the total steps in an episode, termed a horizon.

The effect of the follow-up of step t can be summed up as the time-discounted cumulative rewards G_t by γ, termed a discount return. The objective of the agent is to take actions that maximize the expected value of the sum of future discounted rewards G_t (or discounted cumulative rewards) during an episode, which is equal to

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \cdots + \gamma^{T-t-2} R_T) \end{aligned}$$

$$= R_{t+1} + \gamma G_{t+1}, \tag{4}$$

where T is the final step.

The agent can be designed to learn mappings from states to actions called policies $\pi(a|s)$. When the agent follows policy $\pi(a|s)$ starting from state s to adopt a combination of random variables, each strategy will eventually produce a time-discounted cumulative return G_t. The expectation $\mathbb{E}$ of these cumulative returns is defined as the state-value function or v-function. The state-value function represents the expected return when following policy $\pi(a|s)$ from state s, which is equal to

$$v(s) = \mathbb{E}_\pi[G_t|S_t = s]. \tag{5}$$

Substituting Eq. (4) into Eq. (5) results in the state-value expression of the model-free RL as follows.

$$v(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s]. \tag{6}$$

The Bellman equation for state-value in the Markov decision process is defined as follows [1].

$$v(s) = \sum_a \pi(a|s) \sum_{s'} p(r + \gamma v(s')), \forall s \in S. \tag{7}$$

Notice that the state-value in Eq. (7) is obtained via the transition probability P with the knowledge of the environment model, which is used for model-based RL.

After evaluating the value of the state in Eq. (6) or Eq. (7), the state-action pairs (s, a) must be evaluated to determine what results will be produced after taking this state. By comparing different actions under the same policy, the best action is selected, and the policies are improved. To evaluate the above-mentioned state-action pairs (s, a), the action-value function, q-function, or mean reward is defined as the expected return if the agent follows policy $\pi(a|s)$ after taking action a in state s,

$$\begin{aligned} q(s, a) &= \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a]. \end{aligned} \tag{8}$$

The Bellman equation for action-value is defined as [1]

$$q(s, a) = \sum_{s'} p(r + \gamma v(s')), \forall s \in S, \forall a \in A(s). \tag{9}$$

Eq. (8) and Bellman Eq. (9) are used for model-free RL and model-based RL, respectively. Figure 2 shows the differences in the algorithm used in an interaction cycle between the Monte Carlo and MCTS.

Substituting Eq. (9) into Eq. (7), one can get the relationship between the v-function and q-function in the model-based RL as follows.

$$v(s) = \sum_a \pi(a|s) q(s, a), \forall s \in S, \forall a \in A(s). \tag{10}$$

2.2 Monte Carlo methods

To solve finite Markov decision problems, Dynamic Programming (DP), Monte Carlo (MC) methods, and Temporal Difference (TD) learning are the three fundamental solution approaches [6]. The Monte Carlo methods are used in this study, which are based on the statistical (or large number) principle and require only experience-sample sequences of states, actions, and rewards from actual or simulated interaction with an environment [24]. Therefore, they can be used for direct learning without a model to be learned with the functions of transition probability distribution P. The core idea is that agents constantly interact with the environment to generate a series of historical trajectories and an index of the cumulative return G_t under a specific state and action in the historical trajectories. The Monte Carlo methods do not bootstrap because their estimations for each state are independent and do not build upon the estimate of any other state [24].

In the Monte Carlo methods, the values of the v and q functions are estimated by, respectively, V and Q. The estimated V-function for a state s is the mean return for that state,

$$V_T(s) = \frac{1}{N(s)} \sum_{t=1}^{T} G_t = \frac{1}{N(s)} (G_T + \sum_{t=1}^{T-1} G_t), \tag{11}$$

where the superscript T represents the end step, which is also the end of an episode. Notice that $V_T(s)$ is calculated only at the end of an episode because G_T in Eq. (11) depends on the end step T. The estimated V-function for the step $(T-1)$-th is

$$V_{T-1}(s) = \frac{1}{N(s) - 1} \sum_{t=1}^{T-1} G_t,$$

which can be submitted into Eq. (11) yields.

$$\begin{aligned} V_T(s) &= \frac{1}{N(s)} \Big(G_T + (N(s) - 1) V_{T-1}(s) \Big) \\ &= V_{T-1}(s) + \frac{1}{N(s)} \Big(G_T - V_{T-1}(s) \Big). \end{aligned} \tag{12}$$

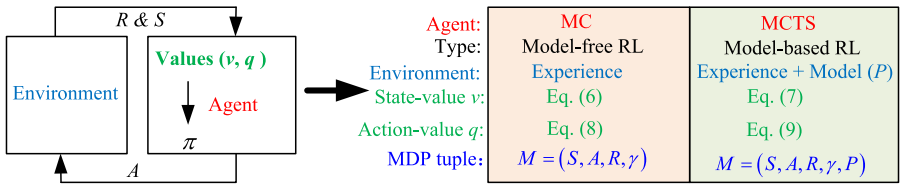


Fig. 2 Algorithm differences of an interaction cycle between Monte Carlo and MCTS–Without the environment model, the agent of Monte Carlo can't get the distribution of P, and therefore, the Markov decision process tuple is $M = (S, A, R, \gamma)$. The state-value and action-value are calculated via Eqs. (6) and (8) due to the lack of P in the model-free RL. For MCTS, the agent learns P from the environment model. Therefore, $M = (S, A, R, \gamma, P)$. And the state-value and action-value are calculated via Eqs. (7) and (9)

Similarly, the estimated Q-function for the state-action pairs (s, a) is the mean return for that pair.

$$\begin{aligned} Q_T(s,a) &= \frac{1}{N(s,a)} \sum_{t=1}^{T} G_t \\ &= Q_{T-1}(s,a) + \frac{1}{N(s,a)} \Big(G_T - Q_{T-1}(s,a) \Big). \end{aligned} \tag{13}$$

When T approaches infinity, the estimate values of Monte Carlo in Eqs. (11) and (13) will approach the true values of model-free Markov decision process in Eqs. (5) and (8), respectively.

$$\begin{aligned} \lim_{T\to\infty} V_T(s) &= \lim_{T\to\infty} \frac{1}{N(s)} \sum_{t=1}^{T} G_t = \mathbb{E}_\pi[G_t | S_t = s] \\ &= v(s), \end{aligned} \tag{14}$$

and

$$\begin{aligned} \lim_{T\to\infty} Q_T(s,a) &= \lim_{T\to\infty} \frac{1}{N(s,a)} \sum_{t=1}^{T} G_t \\ &= \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = q(s,a). \end{aligned} \tag{15}$$

That is, when $T \to \infty$, Monte Carlo methods $\to$ model-free Markov decision process. This implicitly regularizes the Monte Carlo methods towards the previous model-free Markov decision process.

2.3 Vanilla Monte Carlo tree search

Monte Carlo Tree Search is one of the simulation-based search methods that emerged and became popular in gaming. The MCTS (= Monte Carlo methods + Tree search algorithm) framework develops a selective search tree by repeatedly applying a Monte Carlo evaluation (see Sect. 2.2) at each child (or leaf) node and then uses a tree structure to update and select node values efficiently. Unlike traditional planning and learning algorithms, MCTS does not acquire knowledge from the past, i.e., from steps that have already been done. The main characteristic of the simulation-based search is the strategy to look ahead and plan the future best action based on the results of a large number of simulations. The sample returns are observed between each trajectory's initial and termination state, and the values estimate the most promising episodes in the search tree. These simulations are performed to forecast the outcomes of possible decisions and plan in time.

The core of MCTS is an iterative process divided into four steps. See Fig. 3.

(1) Selection – In this step, the algorithm begins at the root node and traverses the tree according to the so-called tree policy. The strategy's main goal is to select the best node to maximize the estimated value and reach a child node.
(2) Expansion – Add a single child node to the search tree under the node selected in the previous step, as shown in Fig. 3.
(3) Rollout (or simulation) – Unlike games, an application to multibody system dynamics involves real-world physical models. Therefore, in this context, the models of inverted pendulums will be primarily utilized. Based on random Monte Carlo simulations from the added child node, the rollout policy is used with a termination outcome with a specific reward. For example, for the double inverted pendulums, once the cart deviates more than a settled distance from the center of the pole tilt exceeds a predetermined degree, the task will be terminated with a reward of 0. Otherwise, the task gets a reward of 1 from each interaction cycle. In addition, the states visited will not be stored in the lookahead

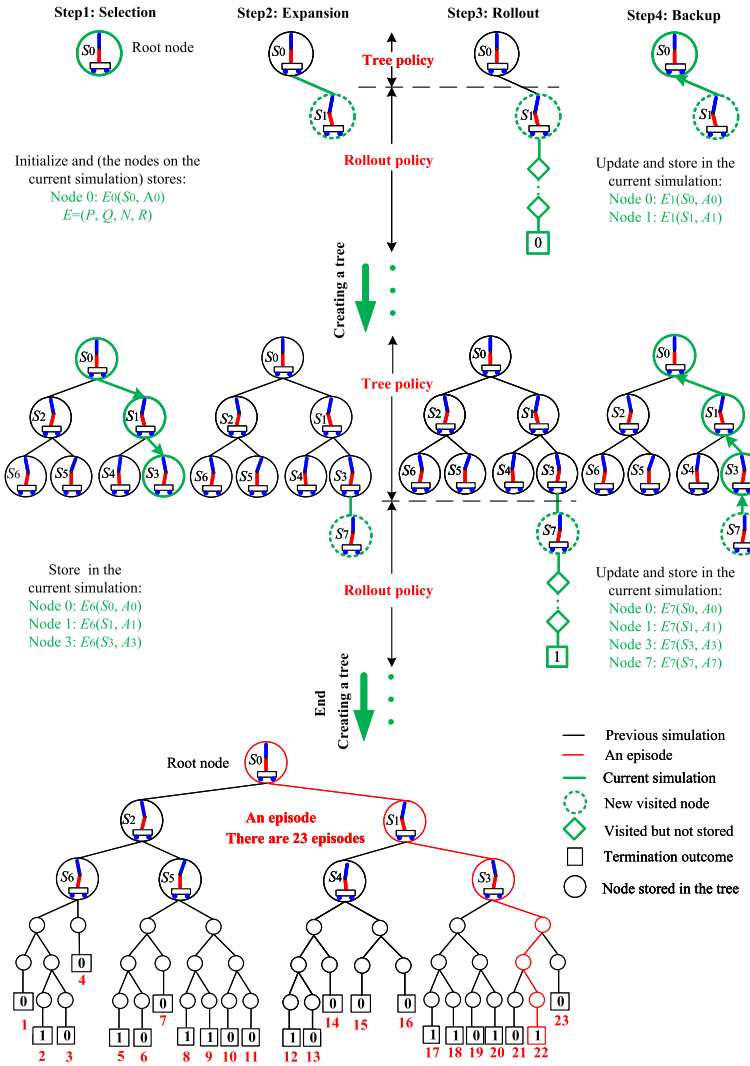


Fig. 3 Tree build scheme in the Monte Carlo Tree Search algorithm. The algorithm is divided into iteration cycles involving four steps (selection, expansion, rollout, and backup). The tree-building process will stop when the search budget in Eq. (3) is exhausted. For illustration purposes, the scheme depicts an environment of double inverted pendulums on a cart with two discrete actions (moving left and moving right) and three states. However, the MCTS scheme is universal and applies to any environment with discrete actions

tree except for the started child node during the rollout. Again, see Fig. 3. Future discounted cumulative rewards can be obtained from Eq. (4) as

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{d-t-1} R_d \\ &= R_{t+1} + \gamma G_{t+1}, \quad 0 < t < d, \end{aligned} \tag{16}$$

where d is the lookahead tree depth and t represents the t-th interaction cycle in this tree. In addition, the rollout includes the states visited but not stored.

(4) Backup (or backpropagation) – After reaching the terminal state in d-th tree depth, the visit count and action-value in Eq. (13) of the current simulation are updated through all nodes up to the root node as

$$N_t(s,a) = N_{t-1}(s,a) + 1, \tag{17}$$

and

$$Q_t(s,a) = Q_{t-1}(s,a) + \frac{G_t - Q_{t-1}(s,a)}{N_t(s,a)}, \tag{18}$$

as shown in Fig. 3. And each edge (s, a) stores a set of statistics as

$$E(s,a) \equiv (P(s,a), Q(s,a), N(s,a), R(s,a)), \tag{19}$$

where $P(s,a)$, $Q(s,a)$, $N(s,a)$, and $R(s,a)$ are a transition probability, an action value, a visit count, and a reward, respectively. When the agent does the t-th interaction cycle, for node i, the following set is stored here (Fig. 3),

$$E_t(S_i, A_i) \equiv (P_t(S_i, A_i), Q_t(S_i, A_i), N_t(S_i, A_i), R_t(S_i, A_i)). \tag{20}$$

In addition, the node must be visited several times to ensure the estimate is reliable.

The iteration of steps 1–4 in MCTS will continue until the search budget n in Eq. (3) is exhausted. There are two policies in the MCTS; one is the tree policy in the selection step, and the other is the rollout policy in the rollout step. Some of the parameters of the MCTS are shown in Table 1. At each interaction cycle, the agent observes the current state of the environment and selects an action that will cause the environment to move to a new state, i.e., left and right. At the same time, the environment will return a reward to the agent, reflecting the result of the action.

2.4 Tree policy

As mentioned in the introduction, there are two challenges facing the tree policy: a sideway growing tree and exploitation vs. exploration trade-off. The former challenge does not exist for the inverted pendulums because only two actions can be chosen here to make the tree grow in-depth.

Exploitation and exploration are two aspects that drive the decision during the tree policy, a trade-off representing real-life problem-solving approaches in RL. The starting point for the trade-off investigation is to give the readers a preliminary impression of how the trade-off parameter affects the control performance of the inverted pendulums. This study uses the tree policy of PUCT (Predictor + Upper Confidence with Tree-based Search policy) [33] to select an action, which is one of the bandit strategies. For child node i, the PUCT formula can be expressed as follows,

$$\pi_{puct}(a|s) = \text{argmax}\left[\underbrace{Q(S_i, A_i)}_{\text{exploitation, greedy}} + \underbrace{U(S_i, A_i)}_{\text{exploration}}\right]. \tag{21}$$

(1) Exploitation: $Q(S_i, A_i)$ is the action-value from Eq. (13), representing the exploitation of the visited good paths.

(2) Exploration: $U(S_i, A_i)$ is the understand level of child node i, which represents the exploration of the unvisited paths. And,

$$U(S_i, A_i) = C_{puct} P(S_i, A_i) \frac{\sqrt{\sum_a N(s,a)}}{N(S_i, A_i) + 1}, \tag{22}$$

where $N(S_i, A_i)$ is the number of child node i has been visited. $\sum_a N(s,a)$ corresponds to the total number of visits done so far and evaluated in the parent node. The state-action pairs in the search tree are edges (s,a), and each node i contains edges (s,a) for all legal actions $\forall a \in A(s)$. C_{puct} [9] is a constant exploration parameter determining the level of exploration, which will be discussed in detail in the numerical examples. When $C_{puct} = 0$, the PUCT policy degenerates to a pure greedy-based policy, e.g., regression tree. The PUCT policy of Eq. (21) initially prefers actions with high transition probability P and low visit count N, but asymptotically prefers actions with high action-value Q [8]. The PUCT formula strengthens the convergence of known reward nodes and encourages the exploration of nodes that have not been visited. In the stochastic control for the uniformly distributed case,

$$P(S_i, A_i) = P(s'|s,a) = \frac{1}{\sum_a n_c(s,a)}, \tag{23}$$

where n_c is the sum of the number of children for the parent.

The environment of an inverted pendulum has only two actions. Exploration involves considering these two actions simultaneously, and their choices are random, discovering new paths with the potential for success. Pure exploitation (or greedy-based) policy lacks exploration and only considers the good existing actions, which is too greedy to utilize other chances for success. As the tree is transversed, the balance between exploitation and exploration must be retained to obtain the maximum value of the PUCT formula, i.e., Eq. (21).

2.5 Reward variants in Monte Carlo tree search

The reward function is developed to evaluate which actions are worth doing, which is vital to achieving

Table 1 Parameters used in the MCTS

Parameter	Symbol	Role	Value	Eqution
Discount parameter	γ	Discounted effects of future predictions	[0 1]	(4)
Exploration parameter	C_{puct}	Determine the level of exploration	[0 ∞)	(22)
Transition probability	P	Select the edge (s, a)	[0 1]	(23)
Search budget	n	Maximum search cost	(0 ∞)	(3)
Episode	e	Sequence of cycles from the beginning to the end	–	(2)
Horizon	H	Total steps in an episode	–	(3)
Cumulative rewards	G_t	Future discounted cumulative reward in an episode	–	(4)
State-value	V	Expected return via policy π from state s	–	(12)
Action-value	Q	Expected return via policy π after taking action a	–	(13)

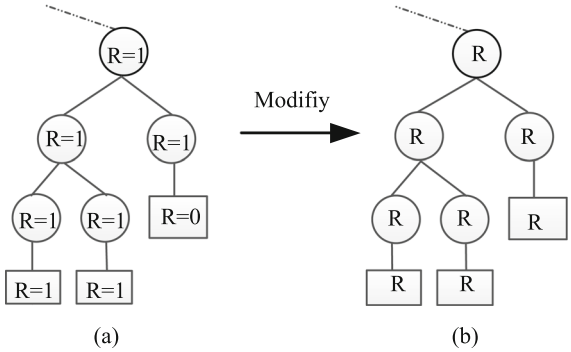


Fig. 4 The reward default MCTS vs. The reward modified MCTS

good learning results. It is nontrivial to design a good reward function. In the MCTS, the reward function is essential to estimate an action-value (see, Eq. (18)) that is next used to exploit the tree. A deep understanding of task logic can also help engineers design effective reward functions for optimal action generation. The default reward function in the Gymnasium cart-pole environment has only two values of 0 or 1 [31]. See Fig. 4a. Unlike some games with clear reward signals, i.e., win ($R = 1$) or lose ($R = 0$), the outcome of multibody system control often needs careful evaluation with a continuous reward function R, see Fig. 4b. The structure of the reward function is important to the agent, which guides the planning for decision-making strategy in the right searching direction towards higher reward regions. Therefore, the reward function should be carefully designed to encourage a better solution.

In Sect. 4, where the numerical examples are presented, several reward functions are evaluated. The baseline is set by the simple reward of a constant 1 value for each successful step. It follows the rule of reward design that the reward should express the desired behavior of the system and not indicate how to achieve such a goal [34]. However, the agent's guidance based on expert knowledge is desirable, and reward design may be one of the easiest ways to achieve it. Therefore, Sect. 4 presents additional rewards for the n-inverted pendulum systems. They provide the highest reward for the pendulum in the equilibrium position (inverted link(s) in the upward vertical position, and, if the cart is also considered, for its position at the center). The further from the zero position, the reward is smaller. Several variants were considered, and they are introduced below.

2.6 The MCTS controller for each time interval

In classical control, the simulation model of the control system is called the plant. Figure 5 illustrates the open loop control system scheme via MCTS for the inverted pendulums on a cart. The MCTS is used as a controller, and the plant could be Single, Double, N-th order Inverted Pendulums (SIP, DIP, and NIP, respectively) on the cart or any other system. For every time interval, a new search tree is built from the current state to decide the following action by comparing the final backup discounted cumulative rewards in Eq. (4) to take an actual left or right action. High-reward actions are more likely to keep the inverted pendulums upright. Therefore, in the inverted pendulum task, the control goal is to stabilize the pendulum and make the task run for as long as possible.

The control concept involves utilizing a reward function, as described in Sect. 2.5, to assess the worth of

actions in achieving good learning outcomes. In the following numerical examples, reward functions are carefully designed based on human intelligence to align with the control objective, shaping a specific distribution of reward bonuses and guiding the MCTS-based control to maintain the upright position. The innovative nature of this control strategy is underscored by the novel application of the MCTS algorithm in controlling multibody systems. The enhanced reward functions have shown to yield superior control performance, compared to the standard MCTS algorithm.

3 Multibody system dynamics

Understanding the dynamic behaviors of complex mechanical systems such as industrial robots, vehicles, machine mechanisms, and rotating structures requires a sophisticated simulation toolset. The multibody system dynamics approach is ideally suited as an effective and efficient dynamics modeling tool for various applications.

In multibody system dynamics, equations of motion can be constructed using two alternative procedures. The common procedure consists of a family of global methods applied to define the absolute translations and rotations of each body of a system with a set of coordinates. Joints are accounted for via constraint equations that couple global coordinates. An alternative to global methods is the family of topological methods, which uses relative coordinates between bodies. Using relative coordinates can lead to a computationally effective procedure of semi-recursive formulation and its variations. In this study, the global formulation is used. The global position of an arbitrary particle can be defined as

$$\mathbf{r}_{A_P} = \mathbf{R}_A + \mathbf{A}_A \overline{\mathbf{u}}_{A_P}, \tag{24}$$

where $\mathbf{R}_A$ is the position of the reference frame of body A with respect to the global reference frame, $\mathbf{A}_A$ is the rotation matrix of body A, and $\overline{\mathbf{u}}_{A_P}$ is the location of particle P in the reference frame of body A. Joints that couple bodies together and limit their motion possibilities can be expressed in terms of constraint equations as

$$\mathbf{C} = [C_1 \quad C_2 \quad \cdots \quad C_{N_c}]^{\mathrm{T}} = \mathbf{0} \in \mathbb{R}^{N_c}, \tag{25}$$

where C_i is i-th constraint equations of the system and N_c is the number of constrain equations.

By following the concept of virtual work [35], the forces acting on a multibody system can be written in terms of generalized coordinates,

$$\delta \mathbf{F} = \delta \mathbf{q}^{\mathrm{T}} \left(\mathbf{M}\ddot{\mathbf{q}} + \mathbf{Q}_v - \mathbf{Q}_e \right), \tag{26}$$

where $\delta \mathbf{q} \in \mathbb{R}^{6N_b}$ is the virtual displacement vector, N_b is the number of bodies, $\mathbf{M} \in \mathbb{R}^{6N_b \times 6N_b}$ is the mass matrix of the system, $\mathbf{Q}_v \in \mathbb{R}^{6N_b}$ is the quadratic velocity vector and $\mathbf{Q}_e \in \mathbb{R}^{6N_b}$ is the vector of externally applied generalized forces. Equation (26) can be set to zero if virtual displacement $\delta \mathbf{q}$ is kinematically admissible. In practice, kinematically admissible displacement can be expressed using the concept of coordinate partitioning.

To derive kinematically admissible displacements, coordinate partitioning begins by grouping generalized coordinates as dependent or independent. This partitioning can be expressed as

$$\delta \mathbf{q} = \begin{bmatrix} \delta \mathbf{q}_d^{\mathrm{T}} & \delta \mathbf{q}_i^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{27}$$

where $\delta \mathbf{q}_d$ is the vector of dependent generalized coordinates and $\delta \mathbf{q}_i$ is the vector of independent generalized coordinates. The dependent generalized coordinate displacements $\delta \mathbf{q}_d$ can be determined using the displacement of the independent generalized coordinates $\delta \mathbf{q}_i$. To this end, constraints (i.e., Eq. (25)) concerning the generalized coordinates must be differentiated to get the Jacobian matrix of the system. When using the concept of coordinate partitioning in the Jacobian matrix, the relationship between all virtual displacement of generalized coordinates and virtual displacement of independent generalized coordinates can be obtained as

$$\delta \mathbf{q} = \begin{bmatrix} -\mathbf{C}_{\mathbf{q}d}^{-1} \mathbf{C}_{\mathbf{q}i} \\ \mathbf{I} \end{bmatrix} \delta \mathbf{q}_i = \mathbf{B} \delta \mathbf{q}_i, \tag{28}$$

where $\mathbf{C}_{\mathbf{q}d}$ is the dependent part of the Jacobian matrix and $\mathbf{C}_{\mathbf{q}i}$ is independent. Using Eq. (28), in Eq. (26), the equations of motion can be written as

$$\mathbf{B}^{\mathrm{T}} \mathbf{M} \mathbf{B} \ddot{\mathbf{q}}_i + \mathbf{B}^{\mathrm{T}} \mathbf{M} \mathbf{D} - \mathbf{B}^{\mathrm{T}} \mathbf{Q}_v - \mathbf{B}^{\mathrm{T}} \mathbf{Q}_e = \mathbf{0}, \tag{29}$$

where matrix $\mathbf{D}$ can be obtained by differentiating the constraints twice, respecting the time, and using coordinate partitioning [35]. Equation (29) uses the minimum number of coordinates to represent the system dynamics.

In summary, the multibody system approach presents a general framework for modeling and analyzing multiphysics systems. In this study, the framework is used to

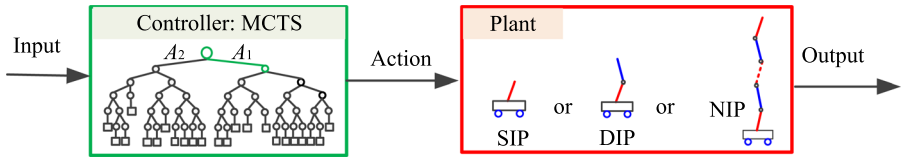


Fig. 5 The control system scheme via MCTS for inverted pendulums on the cart–The plants are controlled in the open loop in this study. The inputs are the parameters of MCTS and the environment, and the output is the state of the plant. In the controller, (1) at every time interval, a new search tree is built from the root node of the current state to decide which next action A_1 or A_2 will be taken for the plant; (2) the search tree is finished until the search budget is exhausted; and (3) the interaction with the simulation environment happens during the tree building process ahead of time with the prescribed time horizon

model both single and double-inverted pendulums on a cart, with the primary purpose of facilitating the roll-out phase of the MCTS algorithm. Although other formalisms could be used to derive the simulation model, the flexibility and versatility of the multibody approach make it an excellent fit for control approaches that rely heavily on model evaluation. Furthermore, the multibody model can be easily modified to account for phenomena such as friction or to include rapid design changes.

4 Numerical examples

Investigating our control concept utilization leads to the study of benchmark performance results.The single and double inverted pendulums on the cart mechanisms are used for benchmark purposes. Because of their nonlinearity and strongly coupled and under-actuated control, such systems are commonly used in the field of control engineering techniques both for theoretical and real-time physical implementation. The control strategies can be classified into three main groups: (1) balance control in the inverted equilibrium point [36], (2) swing-up control to inverted equilibrium [37,38], and (3) random motion control from one equilibrium state to another [39]. This study addressed the balance control task category, where the main objective is to stabilize the pole upwards.

All the examples are programmed using the Julia programming language [40], and all used codes are available online under an MIT license [41,42].

4.1 Simulation environments

The multibody dynamic systems used will be called environments here. In this study, the two planar simulation environments used are single and double inverted pendulums on the cart. They are available on GitHub in Julia package Environments.jl [41]. The single inverted pendulum on the cart is implemented after the OpenAI Gymnasium cart pole environment [31,43] inspired initially by the research of Barto et al. [44]. A double-inverted pendulum on the cart is a variant of the single-inverted pendulum with a second link added. See Fig. 6. The single pendulum variant has two degrees of freedom and coordinates x and θ_1. The double pendulum variant has three DOFs and coordinates x, θ_1, and θ_2. The parameters of pendulum-cart systems are listed in Table 2. Mechanisms operate in a gravitational field, and control input dictates the direction of applied force F acting on the cart body in the $\hat{x}$ axis direction.

For a dynamic model of a double inverted pendulum on the cart, as described above, the main control objective is to keep both poles balanced upwards. The initial state of the simulation is based on the vertical position of both poles, while the parameter that can be regulated is the direction of the actuation force. Its magnitude is given in Table 2. The simulation state time interval is set to 0.02 s, while after each state update, a controller or an agent decision is required about which action in Table 3 will be taken. See Fig. 5.

4.2 Reward for the simulation environment

As discussed in Sect. 2.5, one of the key components of the reinforcement learning environment is the design of the reward function. For the episodic MDP problem where, as an analogy to a real-life problem, the goal is to maximize the time the poles of the mechanism remain upwards. After each state, the reward value that the agent receives is updated.

Table 2 Physical parameters of the SIP and DIP on the cart

Parameter	Symbol	SIP	DIP	Unit
Cart mass	m_c	1	1	kg
Pendulum 1 mass	m_1	0.1	0.1	kg
Pendulum 2 mass	m_2	NA	0.1	kg
Pendulum 1 length	l_1	1	1	m
Pendulum 2 length	l_2	NA	1	m
Pendulum 1 moment of inertia	I_1	8.33×10^{-3}	8.33×10^{-3}	kg.m^2
Pendulum 2 moment of inertia	I_2	NA	8.33×10^{-3}	kg.m^2
Excitation force value	F	10	20	N

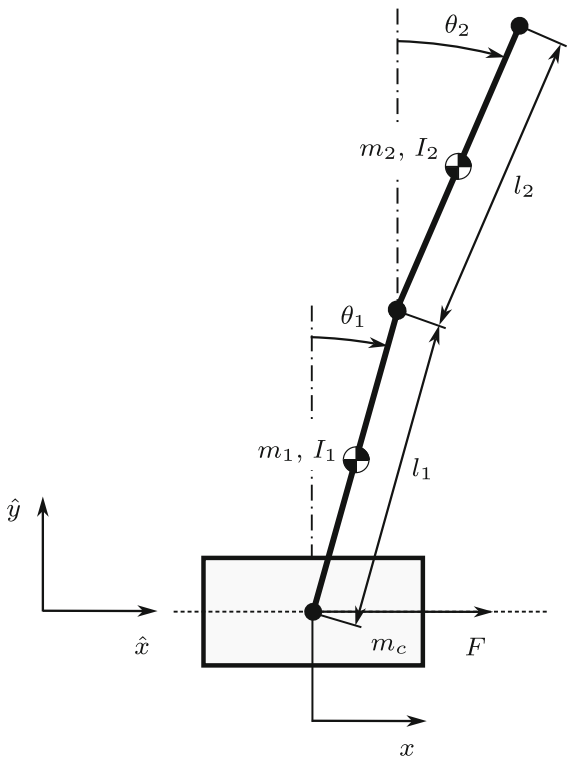


Fig. 6 Double inverted pendulum on a cart–The system has three generalized coordinates: cart displacement x, and pendulum angles θ_1 and θ_2

Termination of the control occurs if any of the conditions in Table 4 is met. The equations for the rewards R shown in this section are valid only when those termination conditions are not met. Specifically cart displacement must follow $|x| \leq x_{\max}$, where $x_{\max}$ is the cart displacement limit, and pole angles $|\theta_i| \leq \theta_{\max}$, $i = 1, 2$ where $\theta_{\max}$ is the angle limitation adopted in this study.

In the following sections, the rewards that are evaluated in the numerical tests are present. Most of those rewards use $x_{\max}$ and $\theta_{\max}$ as scaling factors.

4.2.1 Constant reward

The simple reward of a constant value for each successful step sets the baseline. In this case, the agent can receive $+1$ in all states until the episode terminates.

$$R = 1 \tag{30}$$

It follows the rule of reward design that the reward should express the desired behavior of the system and not indicate how to achieve such a goal [34]. However, the agent's guidance based on expert knowledge is desirable, and reward design may be one of the easiest ways to achieve it. Therefore, below, we present additional rewards that provide the highest reward for the pendulum in the equilibrium position (inverted link in an upward vertical position, and if the cart is also considered for its position at zero). The further from the zero position, the reward is smaller. Several variants were considered, and they are introduced below.

4.2.2 Polynomial reward functions

The first reward function in this category is based on the pole angles and cart displacement.

$$R = 1 - w\frac{1}{n}\sum_{i=1}^{n}\left(\frac{|\theta_i|}{\theta_{\max}}\right)^{p_\theta} - (1-w)\left(\frac{|x|}{x_{\max}}\right)^{p_x} \tag{31}$$

where w is weighting factor (for $w = 1$ only angle is considered, for $w = 0$ only cart displacement), while

p_θ and p_x are power coefficients for angle and displacement. n denotes the number of poles that are stabilized.

The second variant of the polynomial reward traces the tip of the topmost pendulum. The reward is given by equation

$$R = 1 - w\left(\frac{|y^e|}{y^e_{\max}}\right)^{p_y} - (1-w)\left(\frac{|x^e|}{x_{\max}}\right)^{p_x} \tag{32}$$

where $y^e = l\sum_{i=1}^{n}(1-\cos\theta_i)$ is the vertical displacement of the endpoint of the topmost pole concerning the vertical position in an equilibrium state, $y^e_{\max} = nl(1-\cos\theta_{\max})$ is maximum allowed y^e before termination, and $x^e = x - l\sum_{i=1}^{n}\sin\theta_i$ is the horizontal displacement of the endpoint. p_y is the power coefficient for the vertical displacement component.

4.2.3 Exponential reward functions

This reward function is given as

$$R = w\frac{1}{n}\sum_{i=1}^{n}\exp\left[-\left(\frac{\theta_i}{q_\theta\theta_{\max}}\right)^2\right] + (1-w)\exp\left[-\left(\frac{x}{q_x x_{\max}}\right)^2\right] \tag{33}$$

When end tip displacement is considered, the following equation is used.

$$R = w\exp\left[-\left(\frac{y^e}{q_y y^e_{\max}}\right)^2\right] + (1-w)\exp\left[-\left(\frac{x^e}{q_x x_{\max}}\right)^2\right] \tag{34}$$

Parameters q_θ, q_x, and q_y determine how fast the exponential function decreases value from the zero for respective reward components.

To illustrate differences between specific reward functions, Fig. 7 plots the default, polynomial (linear, quadratic, and square root), and exponential reward functions when $w = 1$. The probability of winning by keeping the inverted pendulum upright at $\theta = 0°$ is much greater than $\theta = 12°$, and it decreases as θ deviation away from $\theta = 0°$. This is not considered in the default reward function with a uniform distribution of 1. Therefore, the value distribution of redesigned reward functions is dropping from 1 to 0 as the θ deviation away from $\theta = 0°$, which encourages the planning searching direction towards higher reward regions (upright position). Different dropping slopes $\frac{\partial R}{\partial \theta}$ can result in significant differences in the control performance. See the additional results of the current study online [45].

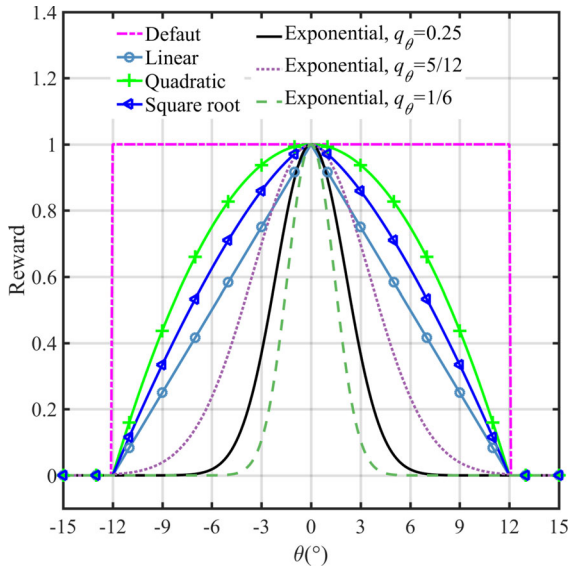


Fig. 7 Reward functions when $w = 1$ and $|x| \le x_{\max}$

4.3 Reinforcement learning environment

The computational study was carried out for two variants of the mechanism: single and double inverted pendulum on the cart. The environments were controlled using the MCTS algorithm (Sect. 2) implemented in the software package PureMCTS.jl [42]. Control algorithm parameters were determined for each environment to perform a sensitivity study to determine the effectiveness of the control process. Table 5 contains individual control parameters indicating their range and step. Each combination of parameters was calculated 10 times for different seed values. The same seed values are chosen to ensure consistent initial conditions for each parameter combination. Two figures are presented to evaluate the control performance of a set of combinations of parameters. The left column presents the mean of all 10 runs, while the right column presents the standard deviation of all 10 runs.

The so-called heatmap plots are presented to evaluate the control performance of a combination of parameters, where each data point is presented explicitly without interpolation. The color map was set so that light colors indicated the desired outcome on both

Table 3 Action space parameters

Parameter	Symbol
Force applied to the cart in positive $\hat{x}$ axis direction	1
Force applied to the cart in negative $\hat{x}$ axis direction	0

Table 4 Termination scenario conditions

Scenario description	Termination value	Condition
Cart reaches the border of graphic window	$x_{\max} = 2.4\text{m}$	$\lvert x\rvert > x_{\max}$
Pendulums tilt angle has exceeded the limit	$\theta_{\max} = 12^\circ$	$\lvert\theta_1\rvert > \theta_{\max}$ or $\lvert\theta_2\rvert > \theta_{\max}$
The number of time steps reaches the limit in the control system	$T_s^{\max}$	$T_s > T_s^{\max}$

Table 5 Single and double pendulum control parameter range

Parameter	Symbol	Single pendulum		Double pendulum	
		Range	Step	Range	Step
Discount parameter	γ	[0.5 1.0]	0.05	[0.7 1.0]	0.05
Exploration parameter	C_{puct}	0 and $[2^1\ 2^{10}]$	$\times 2^1$	0 and $[2^1\ 2^5]$	$\times 2^1$
Search budget (rounded)	n	$[10^{1.5}\ 10^5]$	$\times 10^{0.05}$	$[10^3\ 10^6]$	$\times 10^{0.1}$
Horizon	H	[4 30]	1	[10 40]	2

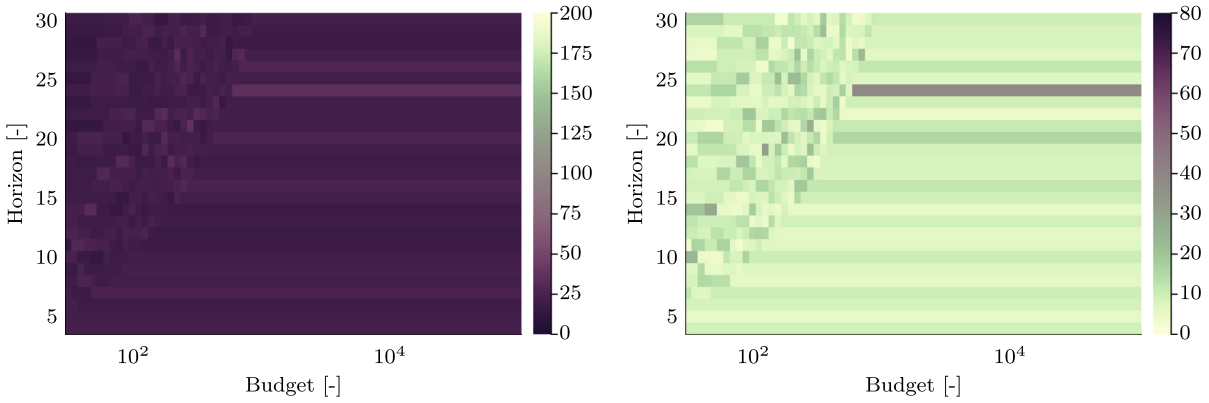
means (left) and standard deviation (right) plots. A light color for the mean means that the pendulum was stabilized for a sufficient duration (at most for T_s time steps). A bright color on the standard deviation plots indicates low solution divergence for various seed settings. Therefore, the most desired solution is a large, continuous bright color area in both mean and standard deviation plots. Such a solution would indicate that the control grade is not sensitive to specific budget and horizon choices.

To quantify the results presented in the plots, we introduce a simple measure of the control performance, the *success rate* S. S expresses the percentage of the successful control episodes to the total number of control episodes. In this context, the single control episode denotes the single point on the heatmap (that is, the control episode is composed of ten control runs for the same agent parameters). The control episode is considered successful when the system is stabilized for all T_s steps. Therefore, the successful control episodes are the brightest possible points in the plots (referring to the maximum mean value of T_s and minimum deviation of 0). As such, S is the ratio of the brightest possible area to the total area of the plot expressed in percentage.

As shown in Table 5, multiple parameter variations are tested. Therefore, only selected results are shown here to indicate the specific behavior of the system. More precisely, only plots for one selected γ value and three different C_{puct} values are shown. For the single pendulum example, $\gamma = 0.85$ and $C_{puct} = 2, 4, 1024$. For the double pendulum example, $\gamma = 1.0$ and $C_{puct} = 8, 32, 128$. The choice of the parameters is made to present the results of the most considerable control quality for simulated examples for constant reward from Eq. (30) – $\gamma = 0.85$, and $C_{puct} = 2$ for the single pendulum and $\gamma = 1.0$, and $C_{puct} = 32$ for double. The example of high and low C_{puct} shows results degradation when less optimal parameters are chosen. The interested reader can see all generated results during the current study online [45]. In addition, all the plots can be generated using script files found in PureMCTS.jl package [42].

4.3.1 Single inverted pendulum

Figures 8, 9, 10, 11 and 12 are the chosen typical results for the single inverted pendulum on the cart for default, constant and exponential reward functions. Table 6 shows all the parameter settings details that generated



(a) Mean T_s for $C_{puct} = 0$, $S = 0\%$ (b) Standard deviation T_s for $C_{puct} = 0$

Fig. 8 Simulation results for single inverted pendulum on cart environment when $T_s = 200$ and $\gamma = 0.85$. (Default case A in Table 6)

corresponding control results are shown online [45]. Analysis of the presented figures reveals the following results.

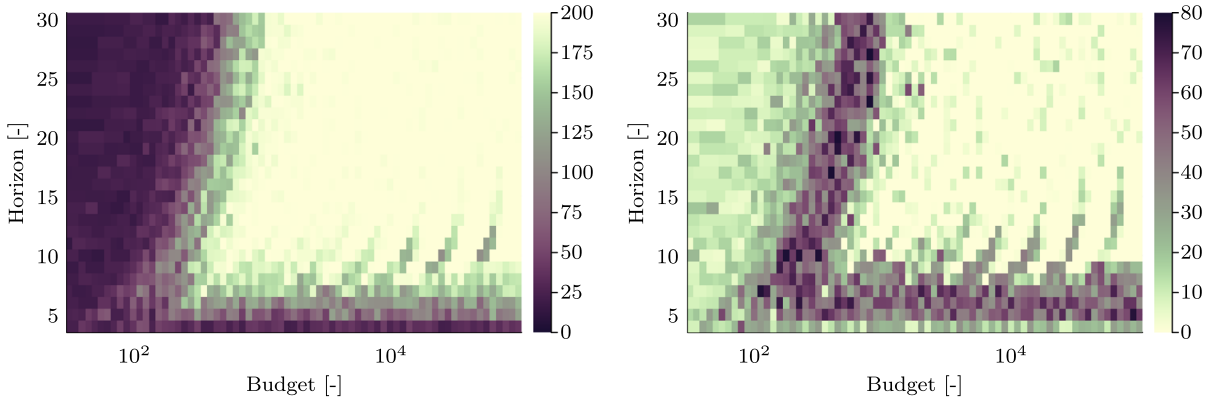
(1) The results in Fig. 8 present the control performance when $C_{puct} = 0$, representing the agent using a purely greedy policy. As pointed out in Sect. 2.4, the pure greedy-based policy lacks exploration and only considers the good existing actions, which is too greedy to utilize other chances for success. All the numerical results in [45] show similar failures for a single/double inverted pendulum on the cart, no matter the discount parameter's value. This indicates that exceptionally poor performance results from the MCTS agent only learning from the previous experience with no exploration involved.

(2) The plots for $C_{puct} = 2$ show a significant area of high control effectiveness in the mean plot. This area begins for budgets of about 1000 and horizon values of 10 to 15. When increasing the value of C_{puct}, such as from 4, 8, 16,32, 64, 128, 256, 512 to 1024, the results are getting worse [45]. The plot for $C_{puct} = 1024$ (see Fig. 9) reveals the pattern where the areas of high control effectiveness are directly adjacent to areas of low control effectiveness. This is achieved for low standard deviation. The adjacent areas of low and high control effectiveness often differ by just one step in the tree-building process (Fig. 3). While the results (as shown for $C_{puct} = 1024$) allow for finding good control results with a reasonably small budget, they also reveal the large sensitivity to parameter selection. Larger C_{puct} values also seem to require larger γ to control the system effectively.

(3) An increase in horizon typically requires a larger budget, which is expected because more simulations are required to build a tree of the same size when a smaller horizon is used. However, the standard deviation plot shows that some combinations of parameters result in worse performance than their neighbors. A similar result pattern was observed for γ from 0.8 up to 1.0 and for C_{puct} up to 64. Moving away from optimal settings results in a smaller area of a high mean T_s and a noisier standard deviation.

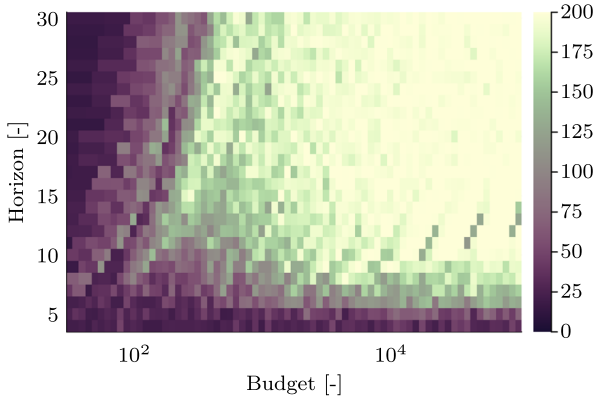
(4) Significantly improvements in the control performance can be found when comparing proposed redesigned reward functions (shown in Sects. 4.2.2 and 4.2.3) to default constant reward. See results in Figs. 9 and 11. The success rate S increases from the 30.3% for case A to 78.0% for the modified case Q. The improvement is more remarkable for longer simulations with $T_s = 500$, where S is just 1.3% for case B and 66.4% for modified case U. These improvements can also be found when using other redesigned reward functions. More results can be found in [45]. These results validate the effectiveness of the proposed reward functions.

(5) A poor control robustness can be found when using the default constant reward by comparing the Figs. 9 and 10. The control performance drops sharply with the same set of parameters when increasing the control time, and almost no combination of parameters can provide acceptable

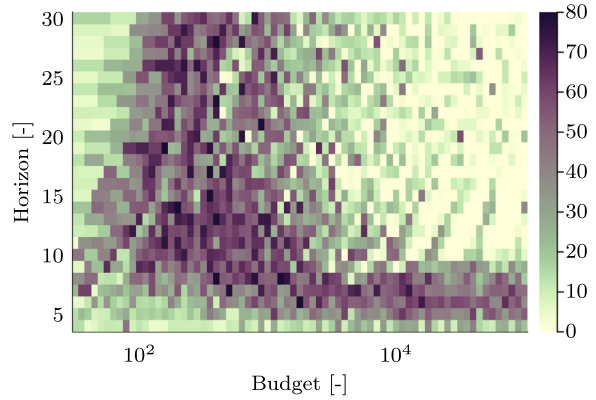


(a) Mean T_s for $C_{puct} = 2$, $S = 30.3\%$

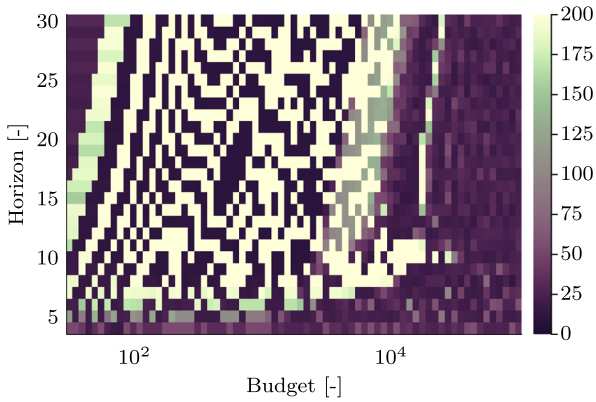
(b) Standard deviation T_s for $C_{puct} = 2$



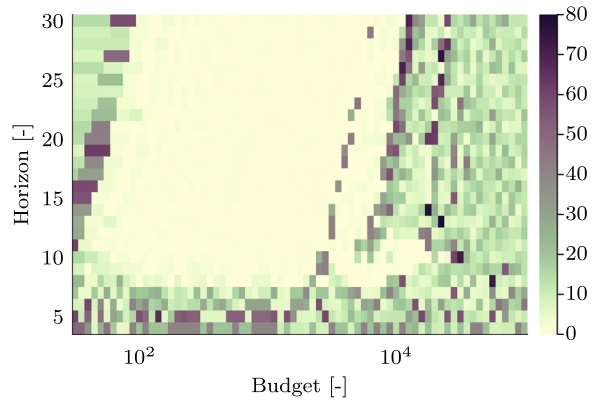
(c) Mean T_s for $C_{puct} = 4$, $S = 14.0\%$



(d) Standard deviation T_s for $C_{puct} = 4$



(e) Mean T_s for $C_{puct} = 1024$, $S = 26.0\%$

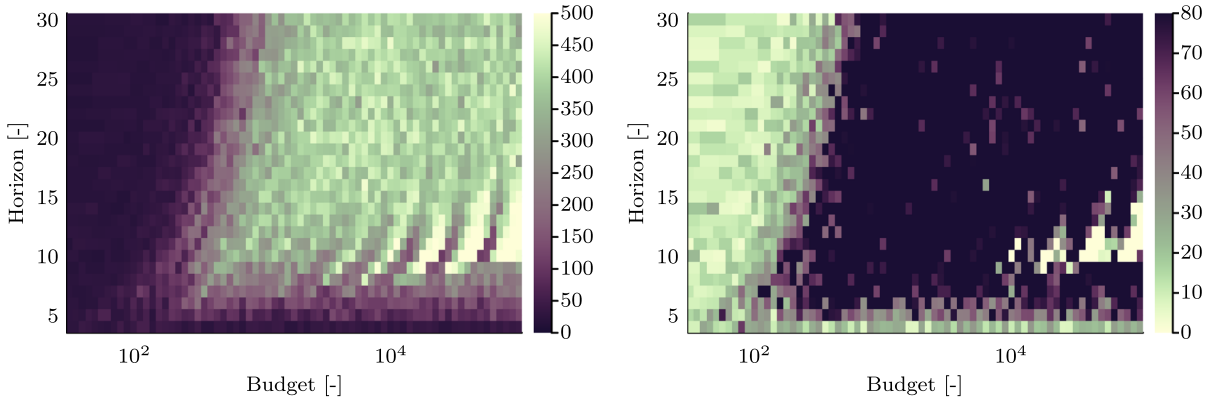


(f) Standard deviation T_s for $C_{puct} = 1024$

Fig. 9 Simulation results for single inverted pendulum on cart environment when $T_s = 200$ and $\gamma = 0.85$. (Default case A in Table 6)

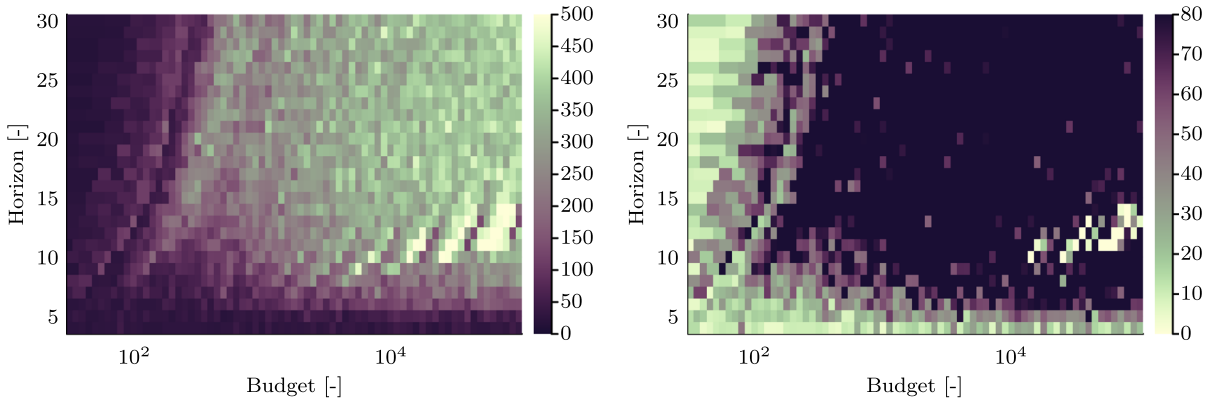
results. However, significant improvements in the control robustness can be found when using the redesigned reward functions by comparing Figs. 11 and 12. Similar significant improvements can also be found using the polynomial reward functions (Eqs. (31) and (32)) and exponential reward functions (Eqs. (33) and (34)). Again, more results can be found in [45].

(6) Comparing all the generated results online [45] whose parameters are listed in Table 6, the exponential reward function of angle penalty when $w = 1$ and $p_\theta = 0.25$ (cases Q and U) shows the best con-



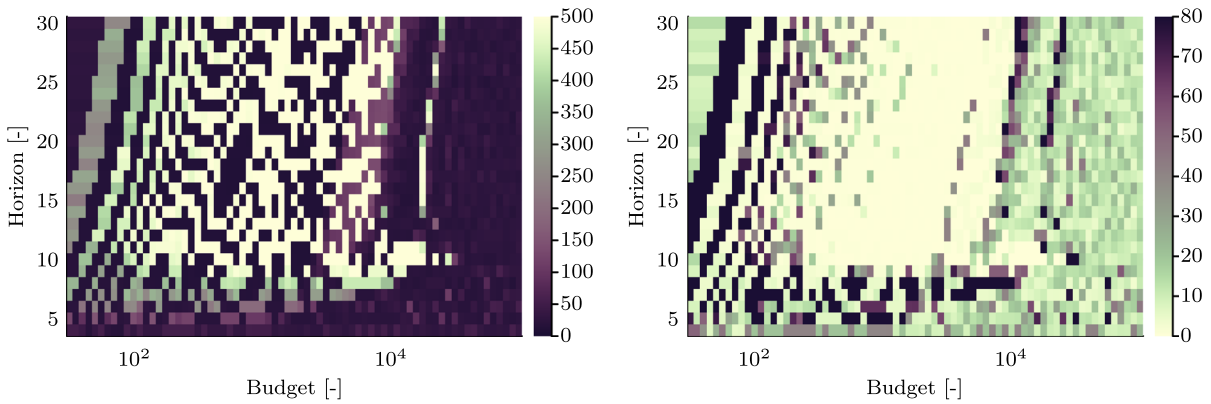
(a) Mean T_s for $C_{puct} = 2$, $S = 1.3\%$

(b) Standard deviation T_s for $C_{puct} = 2$



(c) Mean T_s for $C_{puct} = 4$, $S = 0.6\%$

(d) Standard deviation T_s for $C_{puct} = 4$

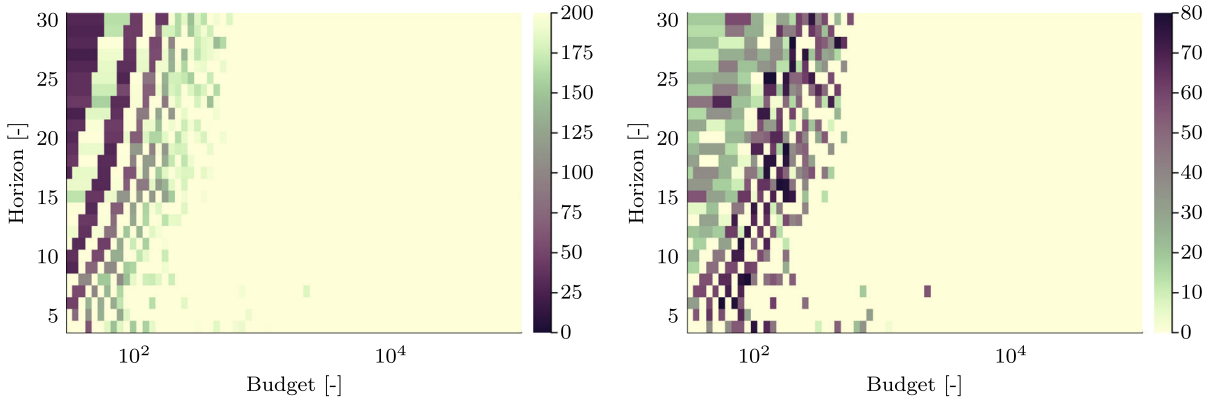


(e) Mean T_s for $C_{puct} = 1024$, $S = 13.9\%$

(f) Standard deviation T_s for $C_{puct} = 1024$

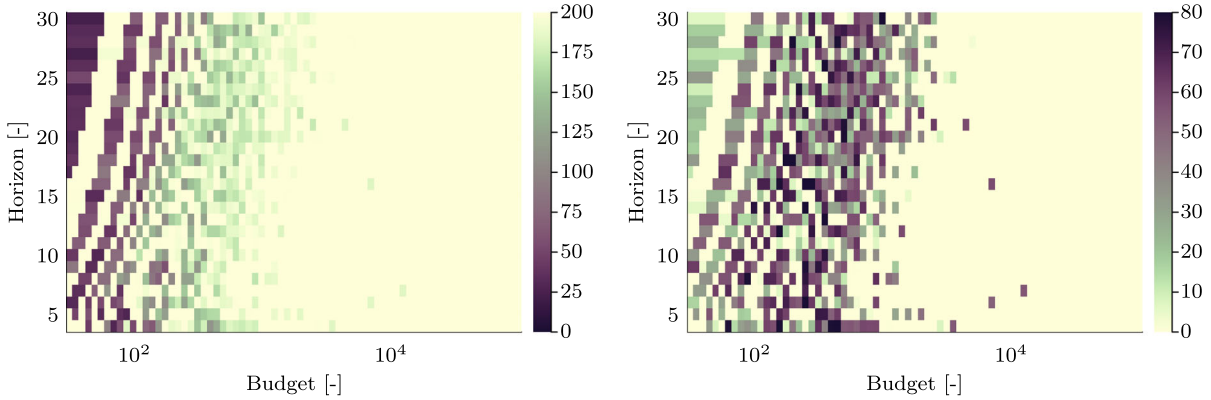
Fig. 10 Simulation results for single inverted pendulum on cart environment when $T_s = 500$ and $\gamma = 0.85$. (Default case B in Table 6)

trol effectiveness for both $T_s = 200$ and $T_s = 500$ in general. For example, the results of Q (angle penalty) show better performance than results of AD (angle penalty and tip displacement penalty) when $C_{puct} = 2, 4$ and 8. Considering only the cart displacement penalty (case C) makes the results worse than the default one (case A) when $T_s = 200$. The reward function with a combination of angle and cart displacement penalties performs not as well as the one that only considers the angle penalty, such as case Q and cases V, W, X, Y, Z, and AA.



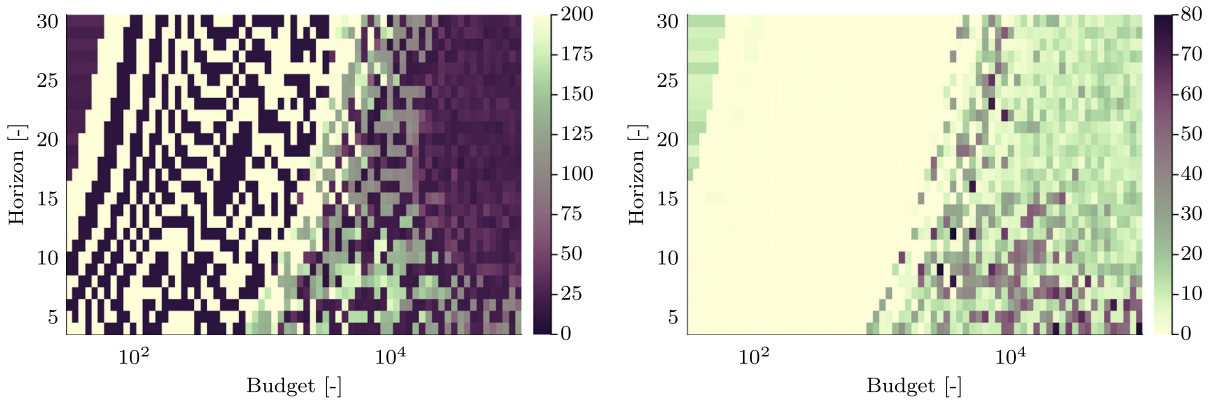
(a) Mean T_s for $C_{puct} = 2$, $S = 78.0\%$

(b) Standard deviation T_s for $C_{puct} = 2$



(c) Mean T_s for $C_{puct} = 4$, $S = 65.3\%$

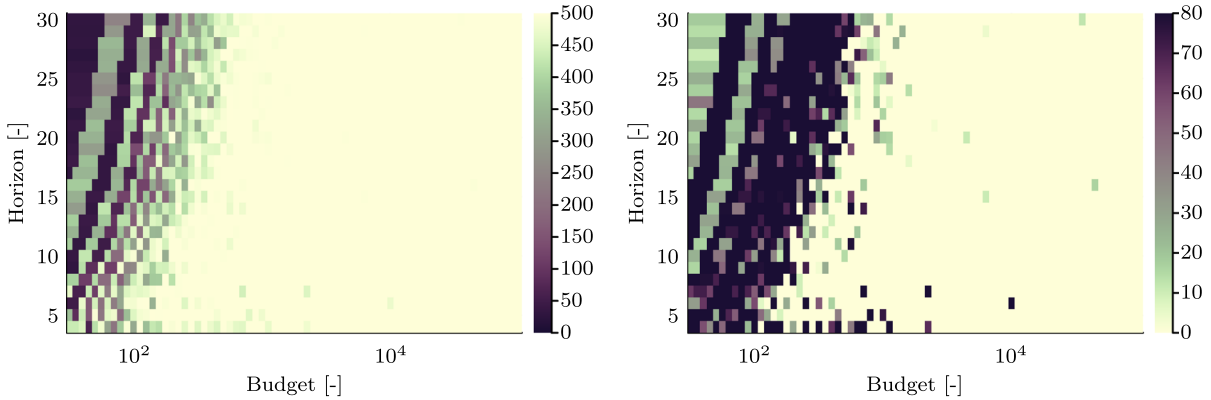
(d) Standard deviation T_s for $C_{puct} = 4$



(e) Mean T_s for $C_{puct} = 1024$, $S = 29.2\%$

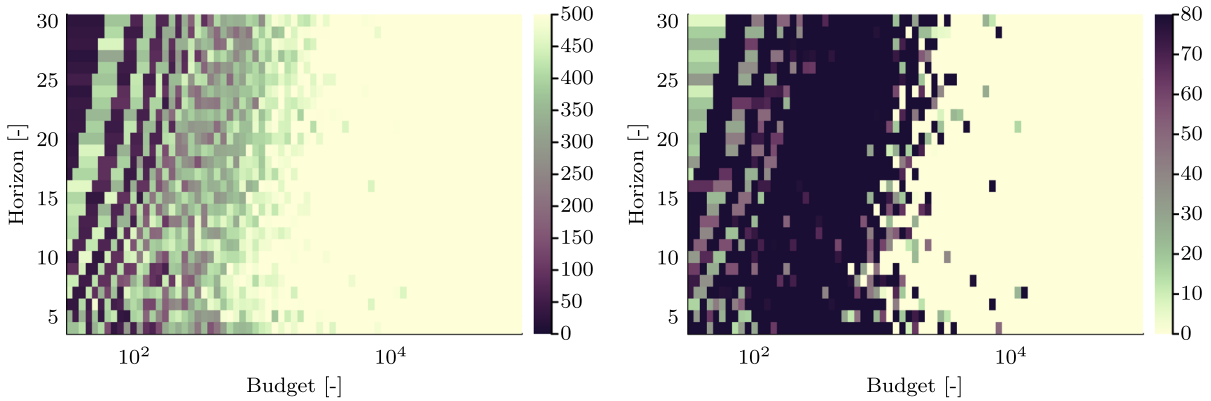
(f) Standard deviation T_s for $C_{puct} = 1024$

Fig. 11 Simulation results for single inverted pendulum on cart environment with modified reward function when $T_s = 200$ and $\gamma = 0.85$. (Modified case Q in Table 6)



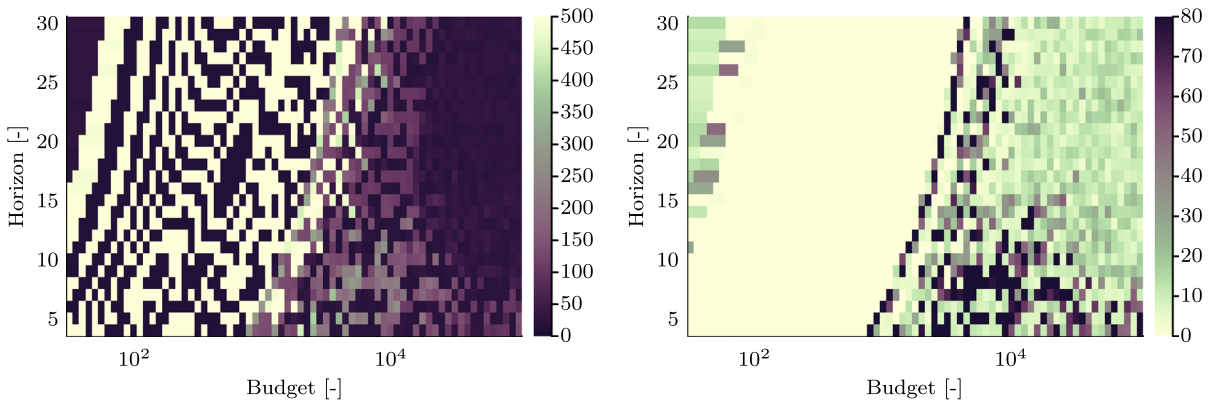
(a) Mean T_s for $C_{puct} = 2$, $S = 66.4\%$

(b) Standard deviation T_s for $C_{puct} = 2$



(c) Mean T_s for $C_{puct} = 4$, $S = 48.6\%$

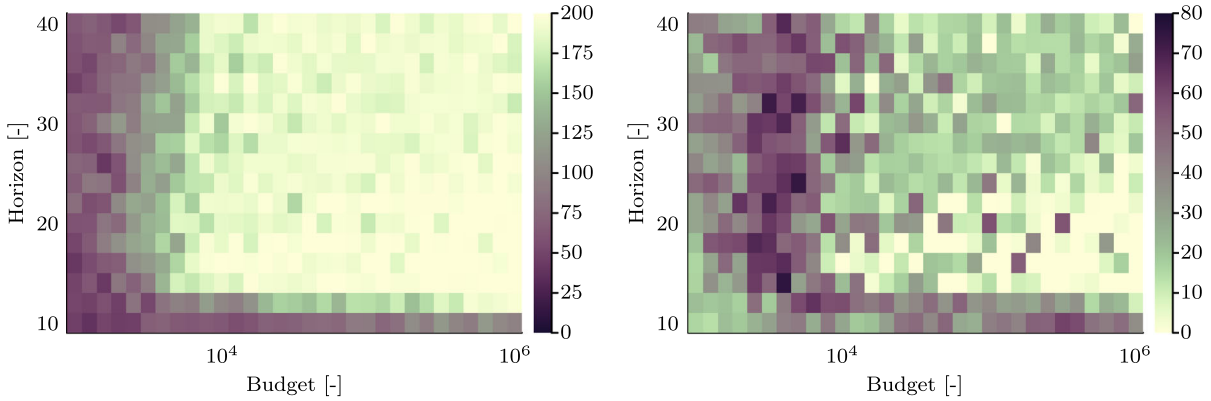
(d) Standard deviation T_s for $C_{puct} = 4$



(e) Mean T_s for $C_{puct} = 1024$, $S = 26.0\%$

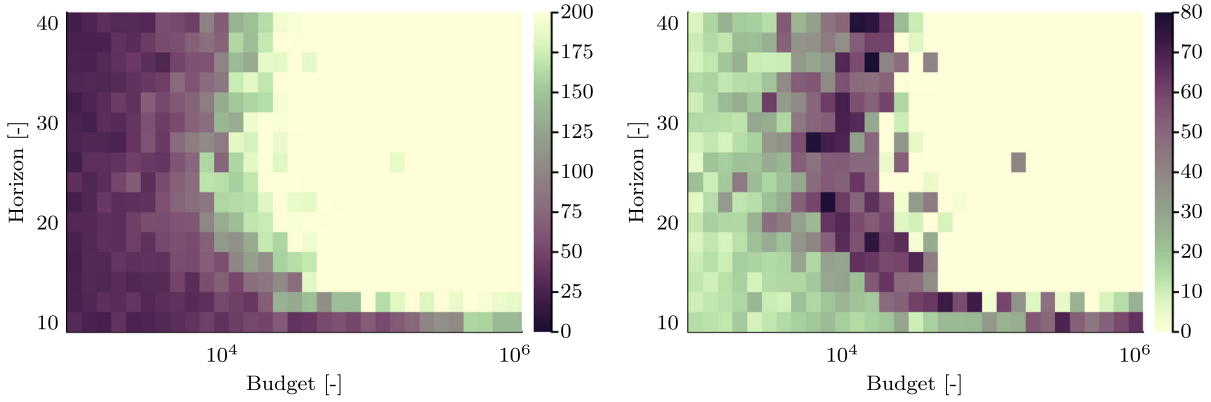
(f) Standard deviation T_s for $C_{puct} = 1024$

Fig. 12 Simulation results for single inverted pendulum on cart environment with modified reward function when $T_s = 500$ and $\gamma = 0.85$. (Modified case U in Table 6)



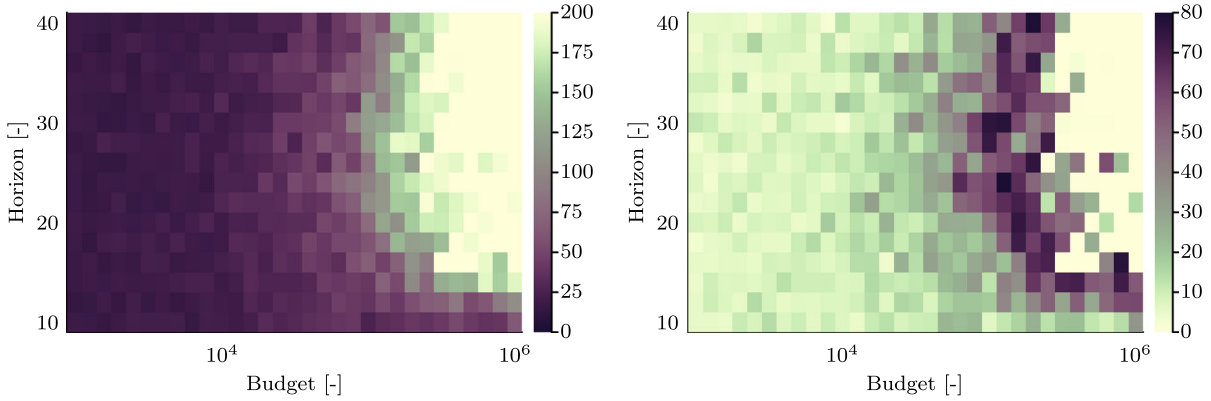
(a) Mean T_s for $C_{puct} = 8$, $S = 8.7\%$

(b) Standard deviation T_s for $C_{puct} = 8$



(c) Mean T_s for $C_{puct} = 32$, $S = 42.3\%$

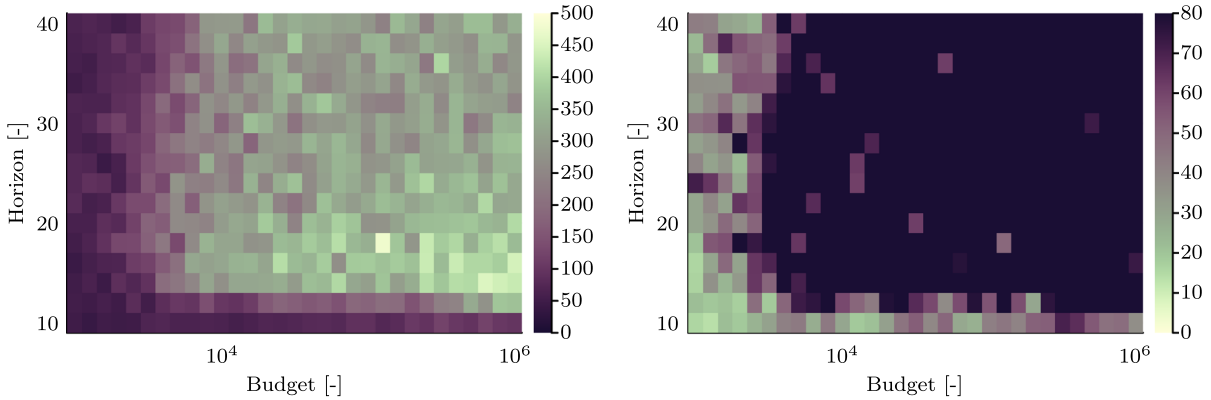
(d) Standard deviation T_s for $C_{puct} = 32$



(e) Mean T_s for $C_{puct} = 128$, $S = 10.1\%$

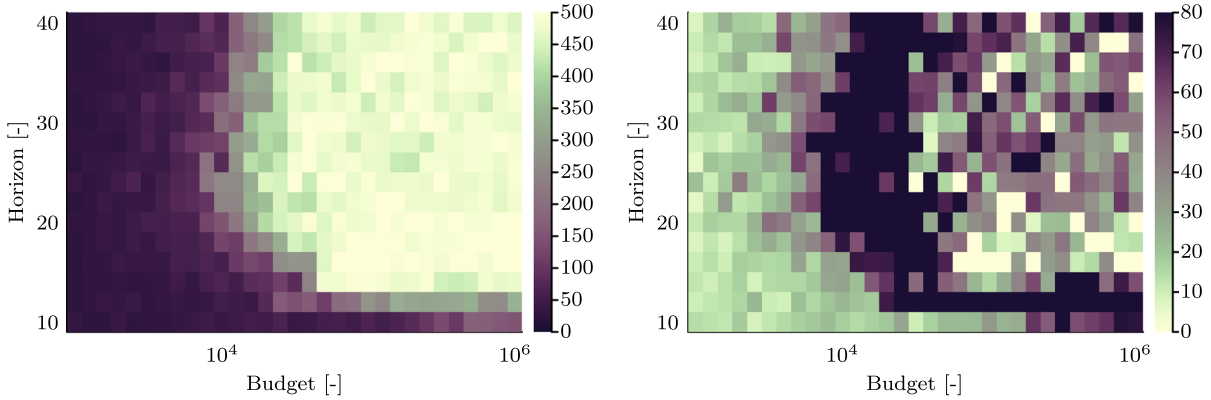
(f) Standard deviation T_s for $C_{puct} = 128$

Fig. 13 Simulation results for double inverted pendulum on cart environment with angle penalty reward when $T_s = 200$ and $\gamma = 1$. (Modified case G in Table 7)



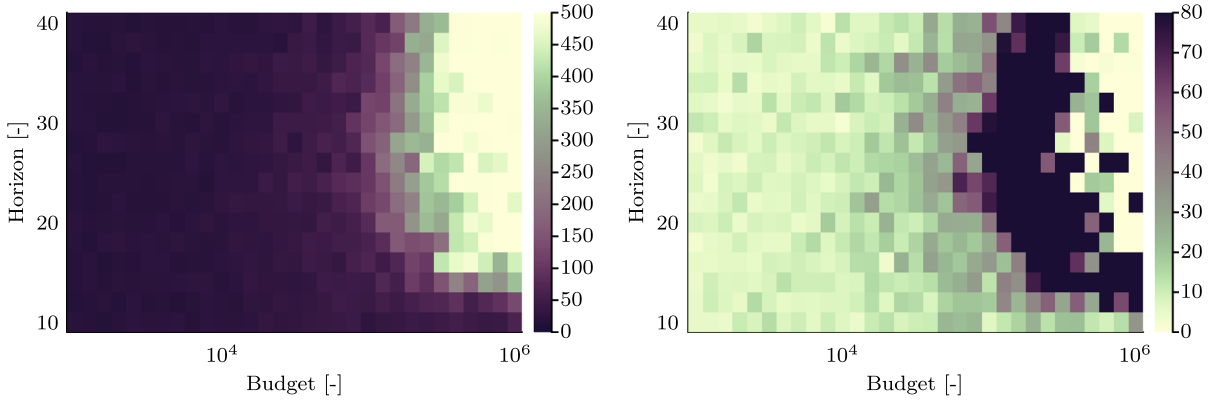
(a) Mean T_s for $C_{puct} = 8$, $S = 0\%$

(b) Standard deviation T_s for $C_{puct} = 8$



(c) Mean T_s for $C_{puct} = 32$, $S = 4.6\%$

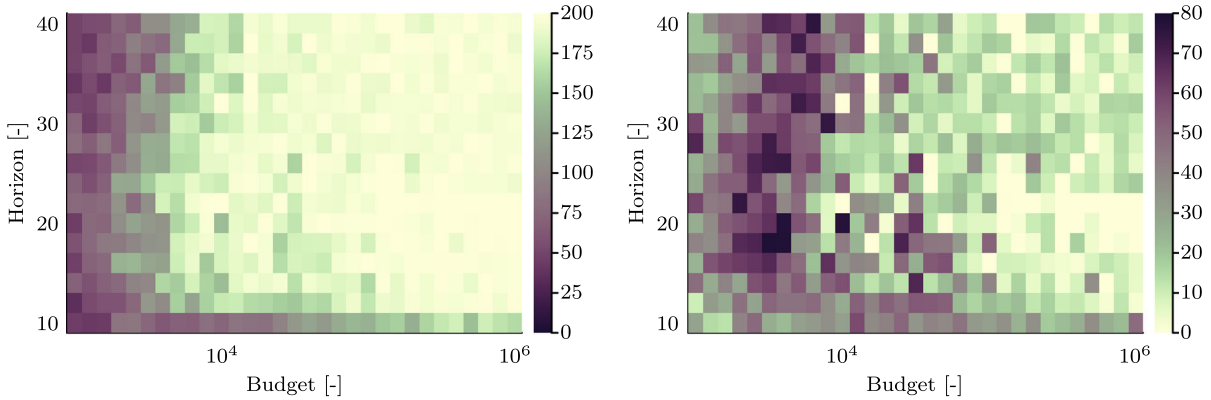
(d) Standard deviation T_s for $C_{puct} = 32$



(e) Mean T_s for $C_{puct} = 128$, $S = 5.2\%$

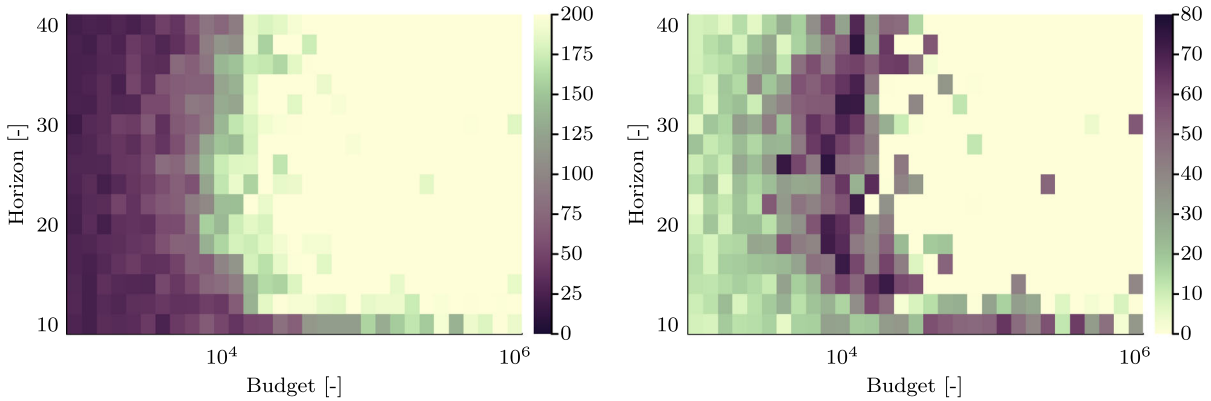
(f) Standard deviation T_s for $C_{puct} = 128$

Fig. 14 Simulation results for double inverted pendulum on cart environment with angle penalty reward when $T_s = 500$ and $\gamma = 1$. (Modified case I in Table 7)



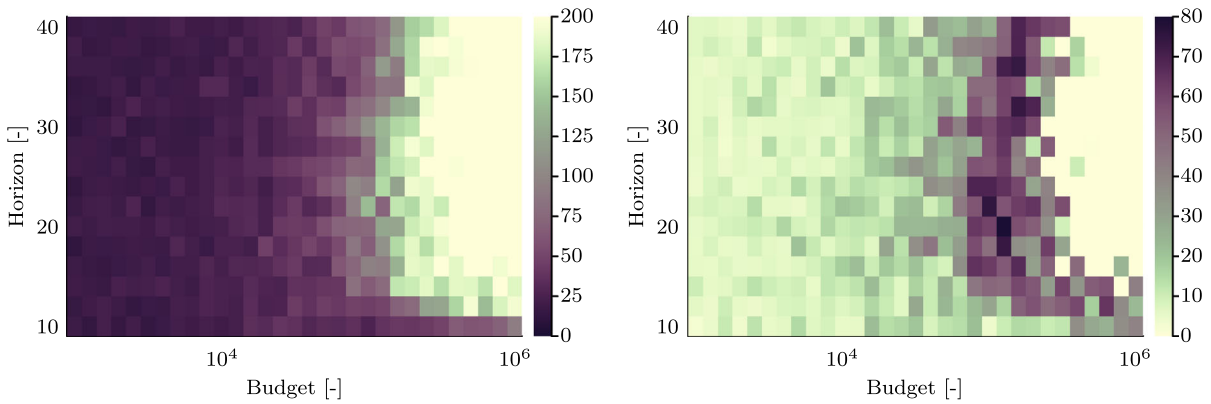
(a) Mean T_s for $C_{puct} = 8$, $S = 7.9\%$

(b) Standard deviation T_s for $C_{puct} = 8$



(c) Mean T_s for $C_{puct} = 32$, $S = 43.1\%$

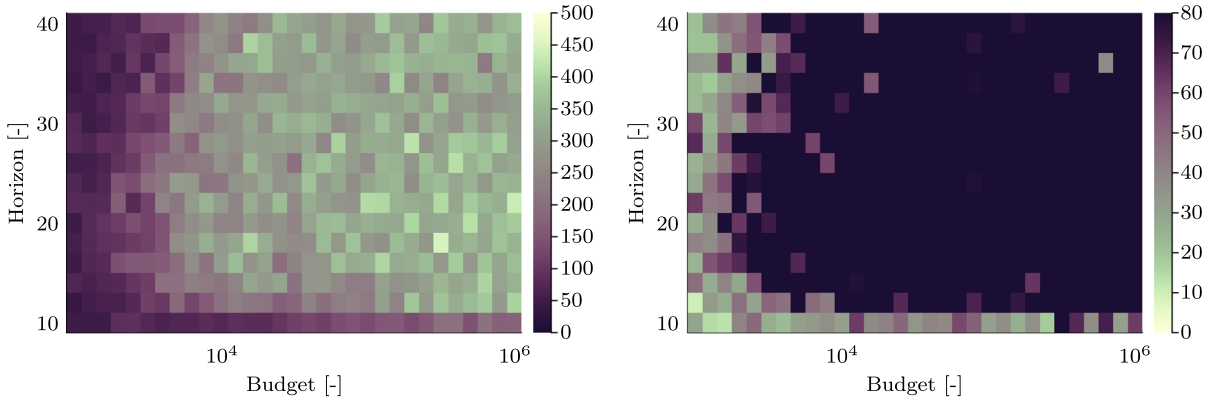
(d) Standard deviation T_s for $C_{puct} = 32$



(e) Mean T_s for $C_{puct} = 128$, $S = 13.9\%$

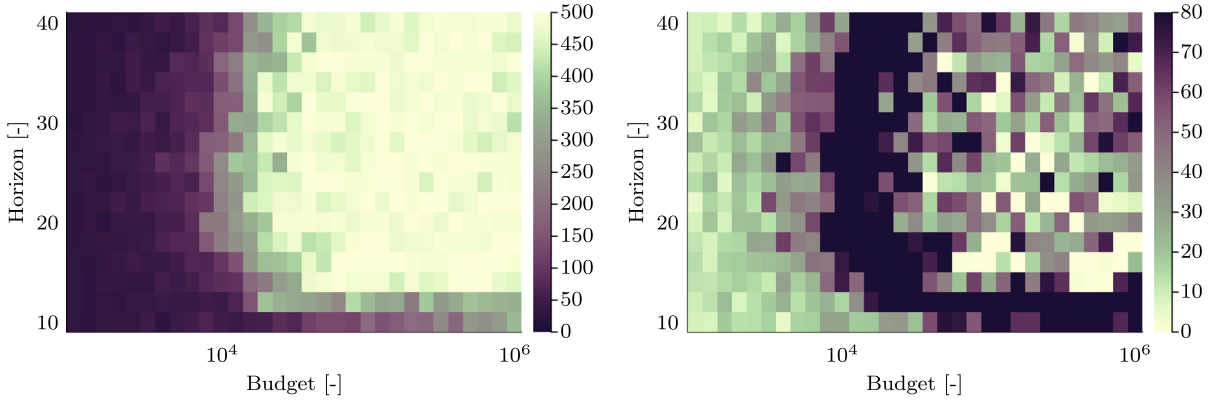
(f) Standard deviation T_s for $C_{puct} = 128$

Fig. 15 Simulation results for double inverted pendulum on cart environment with tip drop reward when $T_s = 200$ and $\gamma = 1$. (Modified case Q in Table 7)



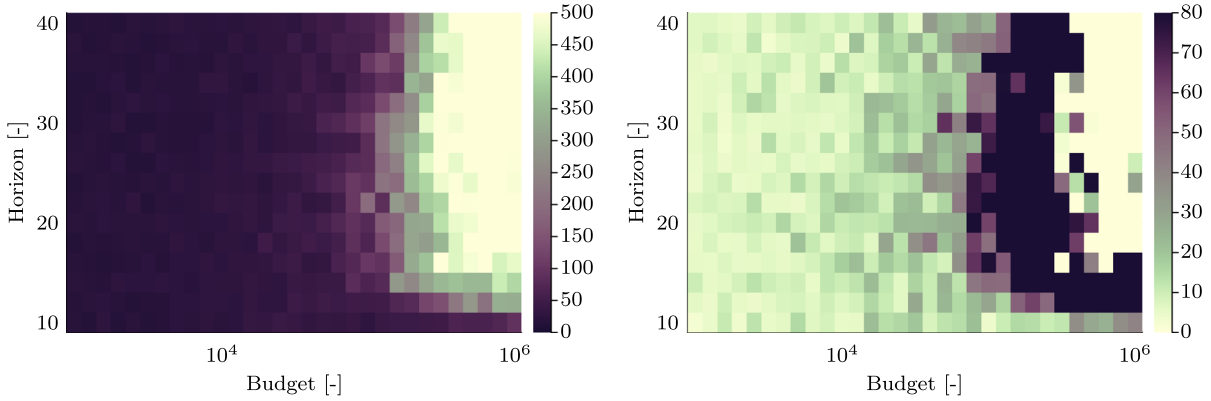
(a) Mean T_s for $C_{puct} = 8$, $S = 0\%$

(b) Standard deviation T_s for $C_{puct} = 8$



(c) Mean T_s for $C_{puct} = 32$, $S = 5.0\%$

(d) Standard deviation T_s for $C_{puct} = 32$



(e) Mean T_s for $C_{puct} = 128$, $S = 9.9\%$

(f) Standard deviation T_s for $C_{puct} = 128$

Fig. 16 Simulation results for double inverted pendulum on cart environment with tip drop reward when $T_s = 500$ and $\gamma = 1$. (Modified case T in Table 7)

Table 6 Test cases for the single pendulum

ID	Name	$T_s^{\max}$	Figure	Equation	w	p_θ or p_y	p_x	q_θ or q_y	q_x
A	Constant	200	8 and 9	(30)	–	–	–	–	–
B	Constant	500	10	(30)	–	–	–	–	–
C	Linear to x	200	–	(31)	0	–	1	–	–
D	Linear to θ_1	200	–	(31)	1	1	–	–	–
E	Linear to θ_1 and x I	200	–	(31)	0.5	1	1	–	–
F	Linear to θ_1 and x II	200	–	(31)	0.75	1	1	–	–
G	Linear to θ_1 and x III	200	–	(31)	0.25	1	1	–	–
H	Linear to θ_1 and x II	500	–	(31)	0.75	1	1	–	–
I	Quadratic to θ_1	200	–	(31)	1	2	–	–	–
J	Quadratic to x	200	–	(31)	0	–	2	–	–
K	Quadratic to θ_1 and x	200	–	(31)	0.75	2	2	–	–
L	Square root to θ_1	200	–	(31)	1	0.5	–	–	–
M	Polynomial to θ_1 and x	200	–	(31)	0.75	2	6	–	–
N	Linear to y^e and x^e I	200	–	(32)	0.5	1	1	–	–
O	Linear to y^e and x^e II	200	–	(32)	0.75	1	1	–	–
P	Exponential to θ_1 I	200	–	(33)	1	–	–	5/12	–
Q	Exponential to θ_1 II	200	11	(33)	1	–	–	0.25	–
R	Exponential to θ_1 III	200	–	(33)	1	–	–	2/3	–
S	Exponential to θ_1 IV	200	–	(33)	1	–	–	1/6	–
T	Exponential to θ_1 V	200	–	(33)	1	–	–	1/12	–
U	Exponential to θ_1 II	500	12	(33)	1	–	–	0.25	–
V	Exponential to θ_1 and x I	200	–	(33)	0.75	–	–	0.25	0.25
W	Exponential to θ_1 and x II	200	–	(33)	0.75	–	–	0.25	0.2
X	Exponential to θ_1 and x III	200	–	(33)	0.75	–	–	0.25	0.3
Y	Exponential to θ_1 and x IV	200	–	(33)	0.85	–	–	0.25	0.25
Z	Exponential to θ_1 and x V	200	–	(33)	0.75	–	–	0.25	0.1
AA	Exponential to θ_1 and x VI	200	–	(33)	0.75	–	–	0.25	0.4
AB	Exponential to θ_1 and x IV	500	–	(33)	0.85	–	–	0.25	0.25
AC	Exponential to θ_1 and x VII	500	–	(33)	0.95	–	–	0.25	0.25
AD	Exponential to y^e I	200	–	(34)	1	–	–	0.25	–
AE	Exponential to y^e II	200	–	(34)	1	–	–	0.1	–
AF	Exponential to y^e III	200	–	(34)	1	–	–	0.4	–
AG	Exponential to y^e II	500	–	(34)	1	–	–	0.1	–

Similar results also can be found when $T_s = 500$, such as in case U and cases AB, and AC.

4.3.2 Double inverted pendulum

The results for the double inverted pendulum on the cart (see Figs. 13, 14, 15 and 16) show similar behavior to their single pendulum counterpart when the large area of the good control effectiveness is achieved. However, a larger budget and horizon values are needed to stabilize the system due to the more considerable complexity of the dynamics of the double-inverted pendulum system than the single one. This translates to far higher computational costs. In addition, only the redesigned reward functions with exponential form are studied in the double-inverted pendulum, as they exhibit the best control performance in the single-inverted pendulum, as stated in the last result in Sect. 4.3.1. Table 7

Table 7 Test cases for the double pendulum

ID	Name	$T_s^{\max}$	Figure	Equation	w	q_θ or q_y	q_x
A	Constant	200	–	(30)	–	–	–
B	Constant	500	–	(30)	–	–	–
C	Exponential to θ_1 I	200	–	(33)	1	0.25	–
D	Exponential to θ_1 II	200	–	(33)	1	0.1	–
E	Exponential to θ_1 III	200	–	(33)	1	0.4	–
F	Exponential to θ_1 IV	200	–	(33)	1	0.55	–
G	Exponential to θ_1 V	200	13	(33)	1	0.7	–
H	Exponential to θ_1 VI	200	–	(33)	1	0.85	–
I	Exponential to θ_1 V	500	14	(33)	1	0.7	–
J	Exponential to θ_1 and x I	200	–	(33)	0.75	0.25	0.25
K	Exponential to θ_1 and x II	200	–	(33)	0.85	0.7	0.7
L	Exponential to θ_1 and x III	200	–	(33)	0.7	0.7	0.7
M	Exponential to θ_1 and x IV	200	–	(33)	0.85	0.7	0.4
N	Exponential to θ_1 and x II	500	–	(33)	0.85	0.7	0.7
O	Exponential to θ_1 and x III	500	–	(33)	0.7	0.7	0.7
P	Exponential to θ_1 and x IV	500	–	(33)	0.85	0.7	0.4
Q	Exponential to y^e I	200	15	(34)	1	0.25	–
R	Exponential to y^e II	200	–	(34)	1	0.1	–
S	Exponential to y^e III	200	–	(34)	1	0.4	–
T	Exponential to y^e I	500	16	(34)	1	0.25	–
U	Exponential to y^e II	500	–	(34)	1	0.1	–
V	Exponential to y^e III	500	–	(34)	1	0.4	–
W	Exponential to y^e and x^e I	200	–	(34)	0.95	0.25	0.25
X	Exponential to y^e and x^e II	200	–	(34)	0.8	0.25	0.25
Y	Exponential to y^e and x^e I	500	–	(34)	0.95	0.25	0.25
Z	Exponential to y^e and x^e II	500	–	(34)	0.8	0.25	0.25

shows the parameter settings details that generated corresponding control results are shown online [45]..

(1) For $C_{puct} = 32$, the area of high control effectiveness is relatively uniform and valid for horizon values above 20 and a budget larger than $5\times^4$. However, even in this region, there are points of visibly worse control effectiveness. With varying C_{puct} and γ values, the effectiveness of the MCTS degraded. Within the tested range of parameters, other regions with relatively good control effectiveness were observed for $C_{puct} = 4$ and $\gamma = 0.85$, $C_{puct} = 8$ and $\gamma = 0.9$, $C_{puct} = 16$ and $\gamma = 0.95$, $C_{puct} = 32$ and $\gamma = 0.95, 1.0$, and $C_{puct} = 64, 128$ and $\gamma = 1.0$. As C_{puct} values increase, larger γ are required. Other sets of parameters result in a lack of clear and continuous areas of high mean and low standard deviation for T_s.

(2) The redesigned reward functions can improve the control robustness and performance of the vanilla MCTS. Similar to the single pendulum case, rewards that combine angle and cart displacement penalties perform not as good as the one with only the angle penalty when $T_s = 200$, such as case G and cases J, K, L, and M. See the results online [45]. Similar results can be found when $T_s = 500$ in case N and cases O and P.

(3) When using a high value of C_{puct}, one new result revealed the areas of high control effectiveness are directly adjacent to areas of low control effectiveness for single-inverted pendulum is no longer

observed for the double-inverted pendulum. This is likely due to its more complex dynamics.

(4) Comparing all the generated results online [45] whose parameters are listed in Table 7, one new result is that the tip displacement penalty's exponential reward functions perform as well as the angle penalty's. For example, with the same parameters' setting of the rest, the results of C (angle penalty) and Q (angle penalty and tip displacement penalty) perform almost the same. This makes the exponential reward functions of angle penalty and tip displacement penalty both promising to control the multi-inverted pendulum.

5 Conclusions

Monte Carlo Tree Search is a decision-making algorithm based on the standard Markov Decision Process logic, which combines the Monte Carlo simulations with the tree search algorithm. It is the main success behind many games because of its simple probability logic and easy implementation via computer technology. Furthermore, parametric analysis can be used to identify the effects of input parameters on model results, which are difficult to estimate for multibody systems and cannot be acquired from the literature. This work aims to provide new insights into the performance of MCTS with the role and impact of its parameters in multibody dynamics control applications. The innovative aspect of this control approach lies in its pioneering use of the MCTS algorithm for controlling multibody systems, with the modified reward functions demonstrating superior control performance compared to the standard MCTS algorithm. In summary, our findings show that:

(1) Compared with a short-sighted agent with a low value of the discount parameter, a long-sighted agent with a high value of the discount parameter, such as $r \in [0.85\ 1]$, often controls the task better.

(2) The exploitation-exploration trade-off is crucial, which suggests using minor exploitation and more exploration can expect significant improvement in the control of mechanical systems. In the context of real-world physical systems, the balance between leveraging prior knowledge (exploitation) and actively searching for new information or actions (exploration) is essential to design systems that are robust against uncertainties like system nonlinearities or parameter modifications. This trade-off is not unique to reinforcement learning but is universally related to decision-making under uncertainty and limited budget.

(3) The proposed reward functions, which can guide the search tree to a higher rewards region, greatly improve the control performance compared to the simple constant reward. This can be easily observed in a single pendulum simulation for $T_s = 500$ and $C_{puct} = 2$ where the success rate S raises from 1.3% (as shown in Fig. 10) to 66.4% (Fig. 12).

(4) The exponential reward functions perform better than the polynomial reward functions. Among them, the angle penalty and tip displacement penalty are most promising for effectively controlling multiple inverted pendulums. On the other hand, penalizing the cart displacements shows little to no improvements.

(5) The vanilla MCTS with constant reward exhibit much worse control performance when control time is extended. The modified MCTS controls the inverted pendulums well when increasing the simulation time.

The current MCTS controller has a limited capability of explaining its decisions. Although we have developed algorithms based on reward functions that can provide interpretable explanations for their decisions, MCTS still lacks transparency, making it difficult to understand how it arrived at a particular decision. The current MCTS controller can predict the control performance for future actions, while the classical control design, such as PID controller, cannot. The predictive ability is powerful and promising in solving more complex systems. Future studies should focus on real-time applications and usage of MCTS with learning for a suitable surrogate model to avoid costly Monte Carlo rollouts. Moreover, the methodology presented could be used to control more complex systems, such as autonomous cranes.

Acknowledgements The authors wish to acknowledge CSC–IT Center for Science, Finland, for computational resources.

Funding Open Access funding provided by LUT University (previously Lappeenranta University of Technology (LUT)). The authors gratefully acknowledge funding provided by Business Finland's "Kumppanuusmalli - SANTTU - LUT" project under grant 8859/31/2021

Data Availability Statement The data that support the findings of this study are available on GitHub. The whole discussed numerical cases are available at https://gorzech.github.io/mcts-pendulum-results/, Reference Number [29]. The data that support the findings of this study are available in Mendeley Data. The codes are available at http://doi.org/10.17632/grk34bx9vf.1, Reference Number [30], and http://doi.org/10.17632/vtfx27gwbz.1, Reference Number [31].

Declarations

Conflict of 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.

References

1. Morales, M.: Grokking deep reinforcement learning. Manning publication, New York (2020)
2. Kurinov, I., Orzechowski, G., Hämäläinen, P., Mikkola, A.: Automated excavator based on reinforcement learning and multibody system dynamics. IEEE Access **8**, 213998–214006 (2020)
3. Luo, F., Xu, T., Lai, H., Chen, X., Zhang, W., Yu, Y.: A survey on model-based reinforcement learning. arXiv preprint arxiv:2206.09328 (2022)
4. Xiao, C., Wu, Y., Ma, C., Schuurmans, D., Müller, M.: Learning to combat compounding-error in model-based reinforcement learning. arXiv:1912.11206 (2019)
5. Puterman, M.: Markov decision processes: discrete stochastic dynamic programming. Wiley, New York (2013)
6. Paniri, M., Dowlatshahi, M.B., Nezamabadi-pour, H.: Ant-td: ant colony optimization plus temporal difference reinforcement learning for multi-label feature selection. Swarm Evol. Comput. **64**, 100892 (2021)
7. Coulom, R.: in the 5th international conference on computer and games (2006), pp. 72–83
8. Silver, D., et al.: Mastering the game of Go without human knowledge. Nature **550**(7676), 354–359 (2017)
9. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree. Nature **529**, 484–489 (2016)
10. Zai, A., Brown, B.: Deep reinforcement learning in action. Manning publications, New York (2020)
11. Zhuang, Y., Li, S., Peters, T.V., Zhang, C.: in 2015 IEEE Conference on Computational Intelligence and Games (CIG) (IEEE, 2015), pp. 314–321
12. Hu, Z., Tu, J., Li, B.: in 2019 IEEE 39th international conference on distributed computing systems (ICDCS) (IEEE, 2019), pp. 2037–2046
13. Pinto, I.P., Coutinho, L.R.: Hierarchical reinforcement learning with monte carlo tree search in computer fighting game. IEEE Trans. Games **11**(3), 290–295 (2018)
14. Kartal, B., Hernandez-Leal, P., Taylor, M.E.: in Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment, vol. 15 (2019), pp. 153–159
15. Świechowski, M., Godlewski, K., Sawicki, B., Mańdziuk, J.: Monte carlo tree search: a review of recent modifications and applications. Artif. Intell. Rev. **56**(3), 2497–2562 (2023)
16. Shen, J., Chen, Y., Huang, P., Guo, Y., Gao, J.: In: Intelligent Robots and Systems (IROS) (In NeurIPS, 2018)
17. Yao, H., Nosrati, M., Rezaee, K.: In: NIPS Workshop on Machine Learning for Intelligent Transportation Systems (MLITS) (2017)
18. Kraemer, K.H., Gelbrecht, M., Pavithran, I., Sujith, R., Marwan, N.: Optimal state space reconstruction via monte carlo decision tree search. Nonlinear Dyn. **108**(2), 1525–1545 (2022)
19. Upadhyay, K., Giovanis, D., Alshareef, A., Knutsen, A., Johnson, C., Carass, A., Ramesh, K.: Data-driven uncertainty quantification in computational human head models. Comput. Meth. Appl. Mech. Eng. **398**, 115108 (2022)
20. Ontanón, S.: Combinatorial multi-armed bandits for real-time strategy games. J. Artif. Intell. Res. **58**, 665–702 (2017)
21. Guo, X., Singh, S., Lewis, R., Lee, H.: Deep learning for reward design to improve monte carlo tree search in atari games. arXiv preprint arXiv:1604.07095 (2016)
22. De Waard, M., Roijers, D.M., Bakkes, S.: In: 2016 IEEE Conference on Computational Intelligence and Games (CIG) (IEEE, 2016), pp. 1–8
23. Chen, L., Liu, J., Jiang, S., Wang, C., Liang, J., Xiao, Y., Zhang, S., Song, R.: In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 32 (2022), pp. 35–43
24. Sutton, R., Barto, A.: Reinforcement learning: an introduction, 2nd edn. MIT press, Cambridge (1998)
25. Graf, T., Platzner, M.: Adaptive playouts for online learning of policies during monte carlo tree search. Theoret. Comput. Sci. **644**, 53–62 (2016)
26. Christiano, P.F., Leike, J., Brown, T., Martic, M., Legg, S., Amodei, D.: Deep reinforcement learning from human preferences. Adv. Neural Inf. Process. Syst. **30** (2017)
27. Angeli, A., Desmet, W., Naets, F.: Deep learning of multibody minimal coordinates for state and input estimation with Kalman filtering. Mult. Syst. Dyn. **53**(2), 205–223 (2021)
28. Peng, H., Song, N., Li, F., Tang, S.: A mechanistic-based data-driven approach for general friction modeling in complex mechanical system. J. Appl. Mech. **89**(7), 071005 (2022)
29. Choi, H., An, J., Han, S., Kim, J., Jung, J., Choi, J., Orzechowski, G., Mikkola, A., Choi, J.: Data-driven simulation for general-purpose multibody dynamics using deep neural networks. Mult. Syst. Dyn. **51**(4), 419–454 (2021)
30. Han, S., Choi, H., Choi, J., Choi, J., Kim, J.: A DNN-based data-driven modeling employing coarse sample data for

real-time flexible multibody dynamics simulations. Comput. Meth. Appl. Mech. Eng. **373**, 113480 (2021)
31. Gymnasium Documentation–Cart Pole. https://gymnasium.farama.org/environments/classic_control/cart_pole/
32. Jonsson, A., Kaufmann, E., Ménard, P., Domingues, O., Leurent, E., Valko, M.: In: Advances in Neural Information Processing Systems (NeurIPS) (2020), pp. 1253–1263
33. Rosin, C.D.: Multi-armed bandits with episode context. Ann. Math. Artif. Intell. **61**, 203–230 (2011)
34. Sutton, R., Barto, A.: Reinforcement learning: An introduction. MIT press, Cambridge (2018)
35. Shabana, A.: Dynamics of multibody systems, vol. 9781107042650, pp. 1–384. Cambridge University Press, Cambridge (2013). https://doi.org/10.1017/CBO9781107337213
36. Joonho, L., Ranjan, M., Hassan, K.: Output feedback stabilization of inverted pendulum on a cart in the presence of uncertainties. Automatica **54**, 146–157 (2015). https://doi.org/10.1016/j.automatica.2015.01.013
37. Hesse, M., Timmermann, J., Hüllermeier, E., Trächtler, A.: A reinforcement learning strategy for the swing-up of the double pendulum on a cart. Procedia Manuf. **24**, 15–20 (2018). https://doi.org/10.1016/j.promfg.2018.06.004
38. Knut, G., Michael, T., Michael, Z.: Swing-up of the double pendulum on a cart by feedforward and feedback control with experimental validation. Automatica **43**(1), 63–71 (2007). https://doi.org/10.1016/j.automatica.2006.07.023
39. Benjamin, J., Lars, W., Johann, R.: On the design of stable periodic orbits of a triple pendulum on a cart with experimental validation. Automatica **125**, 109403 (2021). https://doi.org/10.1016/j.automatica.2020.109403
40. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671
41. Orzechowski, G.: Environments.jl. Mendeley Data, (2023). https://doi.org/10.17632/grk34bx9vf.1
42. Orzechowski, G.: Puremcts.jl. Mendeley Data, (2023). https://doi.org/10.17632/vtfx27gwbz.1
43. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym (2016)
44. Barto, A., Sutton, R., Anderson, C.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans. Syst. Man Cybernet. **SMC–13**(5), 834–846 (1983). https://doi.org/10.1109/TSMC.1983.6313077
45. Orzechowski, G.: Summary of MCTS results. https://gorzech.github.io/mcts-pendulum-results/