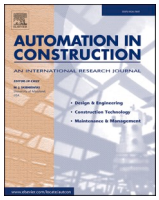
Automation in Construction 165 (2024) 105505

Contents lists available at ScienceDirect

Automation in Construction

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



Towards reinforcement learning - driven TBM cutter changing policies

Tom F. Hansen [a,b,1,*], Georg H. Erharter [b,c,1,*], Thomas Marcher [c]

[a] University of Oslo, Informatics Institute, Blindern, 0316 Oslo, Norway
[b] Norwegian Geotechnical Institute, Sandakerveien 140, 0484 Oslo, Norway
[c] Graz University of Technology – Institute for Rock Mechanics and Tunnelling, Rechbauerstraße 12, 8010 Graz, Austria

ARTICLE INFO

Keywords:
Tunnel boring machine
Cutter wear
Reinforcement learning
Predictive maintenance

ABSTRACT

Optimizing the cutter changing process for tunnel boring machines (TBMs) is crucial for minimizing maintenance costs and maximizing excavation efficiency. This paper introduces TunnRL-CC, a computational framework that utilizes reinforcement learning to autonomously determine cutter-changing strategies. TunnRL-CC's realistic simulation models cutter wear under varying rock conditions, including hard rock and blockyness. A reinforcement learning agent is trained to learn optimal cutter-changing policies based on a reward function that balances cutter conditions and operational costs. The agent demonstrates innovative decision-making, adapting to changing excavation conditions. TunnRL-CC's proposed methodology significantly differs from traditional cutter changing practices, which rely heavily on operator experience. Although TunnRL-CC has not been applied in practical projects, its theoretical basis and comprehensive computational experiments demonstrate its capability to significantly improve TBM cutter maintenance procedures.

1. Motivation

The efficient excavation of tunnels using tunnel boring machines (TBM) hinges on the optimal maintenance of its cutterhead, particularly replacing worn cutters. While cutter wear has been extensively studied [1–7], the process of changing worn cutters has received limited attention. Gehring (1995) [1] pioneered performance and wear forecasts in mechanized tunnel construction, employing empirical data and performance analysis. Thuro (2002) [2] delved into the geological and rock mechanics fundamentals of rock breaking in tunnel construction, providing a theoretical framework for understanding cutter interaction with various rock types. Wang et al. (2012) [3] introduced the energy method to predict disc cutter wear, a novel approach that correlates the energy consumed during excavation with wear extent, offering a predictive model for hard rock TBMs. Hassanpour et al. (2014) [4] proposed an empirical model specifically for predicting TBM cutter wear in pyroclastic and mafic igneous rocks, validated by a case history of the Karaj water conveyance tunnel in Iran. Plinninger et al. (2018) [5] identified rock mass-scale factors affecting tool wear in hard rock mechanized tunnelling, emphasizing the influence of geological conditions on wear rates. She et al. (2023) [6] developed a new method for wear estimation of TBM disc cutter based on energy analysis, enhancing the accuracy of wear prediction. Lastly, Zhang et al. (2023) [7] introduced a new index for cutter life evaluation and an ensemble model for predicting cutter wear, pushing forward the capabilities of data-driven wear assessment.

Some recent studies present data-driven models that predict the wear state of individual cutters as a basis for decision-making for cutter changing [8,9]. The study of Farrokh et al. (2021) [10] presents one of the few exceptions that present actual data on the cutter change time and cutter consumption for hard rock TBMs and therefore is one of the few studies that investigates the cutter changing process itself as opposed to the cutter wear mechanisms. This study introduces a model for predicting cutter consumption but does not offer an adaptable approach for determining the optimal timing and process for cutter disc replacement, movement, and machine intervention, which are critical for efficient and autonomous maintenance.

Conventional cutter changing practices often rely on operator experience, lacking a systematic and data-driven approach. This paper presents TunnRL-CC (Tunnel automation with Reinforcement Learning for cutter changing), a computational framework that enhances TBM cutter changing through reinforcement learning (RL) to autonomously identify optimal cutter-changing strategies. The authors made the first approach in this direction in a small-scale feasibility study [11]. The herein-presented computational framework is based on reinforcement

* Corresponding author.
E-mail address: tom.frode.hansen@ngi.no (T.F. Hansen).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.autcon.2024.105505
Received 28 July 2023; Received in revised form 12 March 2024; Accepted 31 May 2024

learning (RL), which was introduced to geotechnics by Erharter et al. (2021) [12], where the authors presented the first "TunnRL" setup. Slowly RL starts to attract attention in construction in general [13] and also in this field [14,15] although the development of RL in geotechnics is still in its infancy. The TunnRL-CC framework aims to be adaptable across various TBM construction sites without being restricted to specific site data, enhancing cutter-changing processes through a systematic, RL-based approach.

TunnRL-CC's realistic simulation models of cutter wear under varying rock conditions, including hard rock and blockyness, provide a comprehensive representation of TBM excavation dynamics. A reinforcement learning agent is trained to learn optimal cutter-changing policies based on a reward function that balances cutter conditions and operational costs. The agent's ability to adapt to changing excavation conditions and learn complex decision-making algorithms represents a departure from traditional cutter-changing practices. While TunnRL-CC has not yet been directly applied in practical projects, its algorithmic framework and extensive computational experiments demonstrate its potential to improve TBM cutter maintenance procedures.

This research introduces an algorithmic framework for TBM cutter changing that utilizes Reinforcement Learning (RL) to directly optimize cutter-changing policies, diverging from traditional studies focused on cutter wear prediction. This method provides several advantages:

- Autonomous decision-making: TunnRL-CC's RL agent learns to make decisions without explicit instructions, adapting to changing excavation conditions and complex interactions between cutter wear, TBM performance, and operational factors.
- Balanced optimization: The reward function in TunnRL-CC balances the conflicting objectives of maximizing cutterhead lifespan and minimizing maintenance effort, ensuring a holistic approach to cutter maintenance.
- Model-free learning: TunnRL-CC's RL agent learns from experience, without requiring prior knowledge of the cutter wear or excavation dynamics. This allows for adaptation to real-world scenarios with varying rock conditions and TBM configurations.
- Scalability: TunnRL-CC's framework is designed to be scalable to different TBM types and tunnel lengths, making it applicable to a wide range of excavation projects.

The proposed methodology represents a step forward in TBM cutter changing, paving the way for fully automated cutter maintenance systems. The extensive computational experiments presented in this paper demonstrate the effectiveness of the TunnRL-CC framework in optimizing cutter-changing strategies and reducing maintenance costs. Although further real-world validation is needed, the study's encouraging outcomes suggest the potential of RL to significantly improve TBM cutter maintenance practices.

In section 2, we present current best practices for TBM cutter changing. Section 3 explains the TunnRL-CC computational framework for cutter-changing policy optimization, focusing on the geotechnical simulation of the cutter wear and excavation process. The main technical description of the RL agent optimization is given in Appendix A. The results and a discussion are presented in sections 4 and 5, and lastly, a conclusion will be drawn in section 6. The complete Python code of the TunnRL-CC framework is given in the Research Data section at the end of the paper.

2. TBM cutter changing – Current best practice

This section provides an overview of current best practices for cutter-changing policies in TBMs, including methodologies, decision-making criteria, and factors affecting cutter wear. Published literature about disc cutter changing is sparse (except for [10]), and the information given in this section is mostly based on the authors' experience and direct reports from construction sites.

Currently, cutter changing in TBM operations is performed manually during planned maintenance intervals, although there are developments to automate the physical cutter changing process [16,17]. During cutter changing, worn cutters are physically removed from the TBM's cutterhead and replaced with new or refurbished ones. Alternatively, less worn cutters may be relocated to areas on the cutterhead with higher wear. The specific procedures and frequency of cutter changing can vary depending on project requirements, machine specifications, operational constraints, local experience, and common habits.

The decision to change cutters on a TBM is typically based on a combination of factors, including: i) Visual inspection: operators visually assess the wear and condition of the cutter discs by considering parameters such as cutter profile, degree of wear based on profilometers, or obvious defects such as flattened or split cutter rings; ii) Performance monitoring: Real-time monitoring systems track parameters such as thrust, torque, vibration, or energy consumption to detect changes that may indicate reduced cutter effectiveness. Automatic disc cutter wear monitoring is still under development [18]; iii) Maintenance schedules: Planned maintenance intervals or predefined criteria may dictate the frequency of cutter changing.

Cutter wear in TBMs is influenced by various factors, including: i) Geological conditions during excavation significantly affect cutter wear rates due to variations in hardness and abrasiveness, ii) Cutter design and layout affect the wear due to the chosen cutter shape, cutter material, positioning of cutters on the cutterhead and cutter spacing; iii) The way the TBM is operated influences cutter wear due to different factors such as the presence of water or slurry which may be causing adhesive conditions, cutter contact pressure and cutting speed as a result of the total TBM thrust and cutterhead rotations.

Ongoing research aims to optimize these practices through data-driven and machine learning approaches [19], robotics techniques [16], correlations between cutter wear and other geotechnical parameters [20] and factors such as cutter wear patterns and optimization to harsh geological conditions [21]. Understanding the decision basis for cutter changing and the influences on cutter wear is crucial for improving TBM performance, reducing downtime, and enhancing the overall efficiency of tunnelling operations.

3. TunnRL-CC setup

Reinforcement learning (RL) is a machine learning paradigm where an agent learns optimal actions through trial and error by interacting with its environment, aiming to maximize cumulative rewards [22].

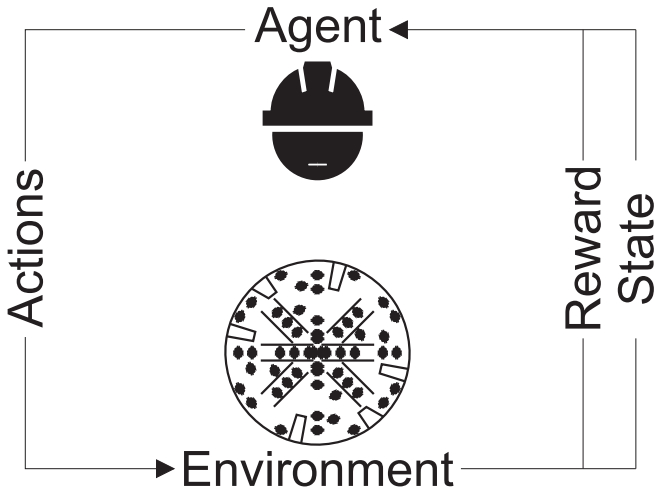


Fig. 1. Reinforcement learning principle contextualized for optimizing a cutter changing policy [11].

This setup is visually depicted for our specific problem in Fig. 1. This approach has evolved from basic tabular methods to sophisticated deep learning algorithms capable of handling complex tasks, ranging from game playing to real-world applications in healthcare and telecommunications [23–26]. RL algorithms are categorized into on-policy, learning directly from current actions, and off-policy, which learn from a different policy's actions, enabling agents to improve without current policy execution. Our study utilizes advanced actor-critic algorithms for on-policy learning, such as Proximal Policy Optimization (PPO) [27] and Advantage Actor Critic (A2C) [28], alongside off-policy methods like Deep Deterministic Policy Gradient (DDPG) [29], Twin Delayed DDPG (TD3) [30], and Soft Actor-Critic (SAC) [31] for continuous control tasks. These methods are chosen for their efficiency in learning and adaptability, featuring mechanisms like entropy addition for enhanced exploration and overestimation bias reduction for stability in training.

At its core, TunnRL-CC consists of an RL setup with an environment that simulates the TBM excavation and the cutter wear and an RL agent that controls the cutter changing process. The framework can be run in three modes of operation:

- Optimization: Due to the problem mentioned above that RL algorithms' performance is highly sensitive towards their internal setup (i.e., algorithm's architecture and hyperparameters) the first step is to choose one basic type of RL agent and optimize its architecture and hyperparameters for the given environment. The optimization process is done automatically, and more information is given in section 3.2, 4.1 and Appendix A.
- Training: This mode enables training of one specific agent architecture and set of hyperparameters that eventually have been found in the optimization step.
- Execution: This mode enables loading a trained agent and subsequently running it through a specific number of episodes to analyze its performance and the methods of cutter changing applied. The episode records are then used to finalize the cutter changing policy.

An overview of the TunnRL-CC framework is given in Fig. 2.

3.1. Excavation and cutter changing simulation

The environment simulates the cutter wear process throughout the TBM excavation and gives the agent both a state- and a reward signal. A list of all symbols is given in Table B.2. in Appendix B.

The state (S) which the agent observes and uses as a base for its decisions represents the current life of each cutter. S is a vector of length n_{c_tot}, where n_{c_tot} is the total number of cutters of the cutterhead $\{s_0, s_1, \ldots s_{n_{c\_tot}}\}$ and each cutter can take values between 1 (new cutter) and 0 (broken/worn-off cutter). The normalized cutter life in the range of 0–1 is based on the theoretical durability of each cutter that is quantified as c_l in "rolling meters per cutter" (e.g., a default cutter life c_l of 40,000 m for each cutter is set based on experience and literature [32]). Due to the simulation of unique geological conditions in every episode (see below), the TBM will achieve a higher or lower penetration rate depending on the encountered geology and consequently the cutters wear down with respect to the geology: high TBM penetration (soft rock mass) → few cutterhead rotations per meter → low cutter wear; low TBM penetration (hard rock mass) → many cutterhead rotations per meter → high cutter wear. The default cutter life of 40,000 m for each cutter serves as an initial reference for this study but the actual cutter life can deviate significantly based on the influences given in section 2. Future versions of the TunnRL-CC framework will integrate more of them to offer a more precise prediction of cutter life under various conditions.

After every stroke (also known as "TBM advance"), c_l for each cutter is decreased based on the "travelled distance" of that cutter which is computed from the cutter's position on the cutterhead, the length of the stroke (l_s) and the penetration rate (p) of that stroke (see below for how p is simulated). A "stroke" refers to one advance length of the TBM corresponding to one full extension of the machines thrust cylinders (usually a distance between 1 and 2 m). Depending on the geological conditions, different numbers of cutterhead rotations can be required to completely excavate one stroke. In the simulation, inner cutters wear down slower than outer cutters, which are in good accordance with real TBM cutter wear [10,32,33]. A basic assumption in the simulation is that the agent can observe S after every stroke of the TBM, which aligns with current developments as stated in section 2.

One episode in the simulation consists of a fixed number of n_s strokes excavated by the TBM. Therefore, for every new episode, a new set of simulated TBM data is generated based on the model for TBM performance prediction by Delisio et al. (2014) [34]. This TBM penetration model was chosen to simulate the excavation process as it is seen to be well suited to synthesize TBM data for hard rock TBM excavations and permits including special cutter failure events like failure due to "blockyness" in the simulation. In the simulation, the TBM penetration model is based on two sets of data that are being generated by random walks within boundaries:

- a vector of length n_s for the volumetric joint count (J_v) [joints / m^3] and
- a vector of length n_s for the intact rock's unconfined compressive strength (UCS) [MPa]

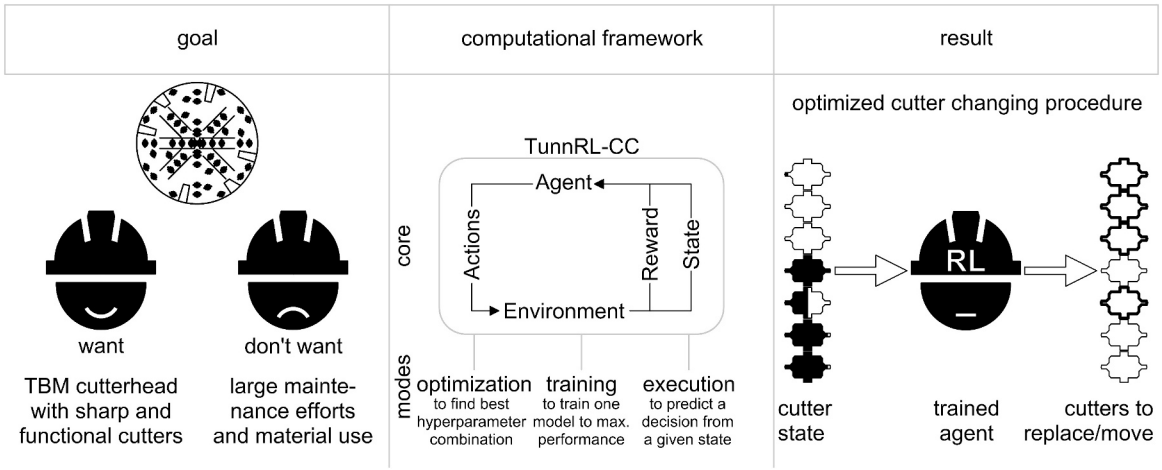


Fig. 2. Overview of the TunnRL-CC framework. Left: goal; middle: computational framework; right: result.

The field penetration index for blocky rock mass conditions (FPI_{blocky}) [kN/m/mm/rot] is then computed with eq. 1 (i.e. eq. 9 in [34]).

$$FPI_{blocky} = e^{6} * J_v^{-0.82} * UCS^{0.17} \quad (1)$$

To simulate the occurrence of hard to predict failures of individual cutters due to "blocky" rock mass conditions, between 1 and 4 cutters have a certain chance to fail in every stroke based on the FPI_{blocky}. Table 1 gives an overview of the failure likelihoods. The chosen ranges are directly based on the proposed ranges of blockyness of Delisio et al. (2014) [34] and the chances of cutters to fail were chosen so that a "sufficient amount of failures" will occur in the simulation. Adapting boundary conditions like this will also influence the policies that the agent learns and thus permit to train agents for specific site conditions. We assume a consistent 1/100 probability of cutter failure for both FPI ranges: <50 and 200–300. Despite differing physical states—disturbed and weak for FPI < 50, and blocky or intact for FPI 200–300—both ranges exhibit homogenous rock mass in their own way, leading to similar failure risks.

The TBM's thrust force (TF) [kN] can then be computed with eq. 2 where D represents the TBM's cutterhead diameter (based on eq. 13 in Delisio et al. (2014) [34]).

$$TF = D * (-523 * ln(J_v) + 2312) \quad (2)$$

An upper limit of 20 MN is set for TF as to avoid generation of unrealistically large thrusts that would not be used in practice to avoid damaging the machine. Finally, the TBM's penetration rate (p) [mm/rot] can be computed with eq. 3 (based on eq. 7 in Delisio et al. (2014) [34] but note that as opposed to the original equation, the friction force is already included in the computed TF of eq. 2).

$$p = \frac{TF}{D * FPI_{blocky}} \quad (3)$$

In Fig. 3, an example is given for one episode of generated data, including cutter failures due to blocky rock mass behavior, as explained above.

After every stroke, the agent receives a reward (r) and aims to maximize the total cumulative reward over the whole episode (R). Establishing an effective reward function is paramount as it significantly influences the behavior of the agent and permits conveying of domain knowledge, thereby ensuring the agent exhibits the desired behavior. Through extensive testing of the TunnRL-CC framework, the reward function of eq. 4 was developed which is based on four conditions that are being checked in sequential order and r will take the value of the first true condition.

$$r = \begin{cases} -1 \; if \; condition \; 1 \\ 0 \; if \; condition \; 2 \\ eq.5 \; if \; condition \; 3 \\ eq.6 \; if \; condition \; 4 \end{cases} \quad (4)$$

The four conditions are:

- condition 1: if $n_{c\_good} < n_{c\_tot} * t$ with n_{c_good} being the number of cutters with a life >0 and t being a manually set threshold to control the minimum required number of non-broken / worn-down cutters on the cutterhead (e.g. 85%, based on construction site experience).
- condition 2: if a bearing failure has occurred on at least 1 cutter. A bearing failure of a cutter occurs when a cutter was broken due to blocky rock mass behavior but is not immediately changed in the subsequent strokes. If this would happen in reality, the cutter's bearing would fail and thus the whole cutter would require changing.
- condition 3: if no cutter was acted on by the agent (i.e., no replacement or moving of cutters required – see next section), then r is computed with eq. 5.
- condition 4: in all other cases, r is computed with eq. 6.

$$r = \frac{n_{c\_good}}{n_{c\_tot}} \quad (5)$$

$$r = \frac{n_{c_{good}}}{n_{c_{tot}}} - \frac{c_{rw}}{c_w} * \alpha - \frac{c_{mw}}{c_w} * \beta - \frac{d_c}{n_{c_{tot}}} * \gamma - c * \delta \quad (6)$$

The second term in eq. 6 ($\frac{c_{rw}}{c_w}$) is a penalty for replacing cutters where c_{rw} is the weighted sum of all replaced cutters in that stroke and c_w is the weighted sum of all cutters. The weighting is done linearly by giving a higher penalty for replacing outer cutters than for replacing inner cutters, thus representing the work effort associated with cutter replacement. The term $\frac{c_{mw}}{c_w}$ represents a penalty for moving cutters from one position to another on the cutterhead where c_{mw} is again the weighted sum of all moved cutters of that stroke. $\frac{d_c}{n_{c_{tot}}}$ is a penalty for acting on single cutters where d_c represents the total distance between all cutters that are being acted on in that stroke expressed as their index difference (e.g., when cutters on positions 3, 12 and 18 are acted on, d_c would be 18–3 = 15). This aims to motivate the agent to change cutters in series instead of single cutters. c is the last penalty for having to enter the cutterhead in general and should help to motivate the agent to find efficient strategies. α (replacement), β (movement), γ (distance) and δ (enter) are weighting factors which help to control the influence of the four penalties onto r and must all add up to 1. These factors can be tailored to accommodate site-specific requirements. For instance, in a project where labor costs are high and/or material prices are low, prioritizing replacement over movement is recommended and vice versa.

The reward system gives a r for every stroke in the range of −1 to 1 and thus an episode with 1000 strokes can have a minimum and maximum cumulative reward of −1000 and 1000 respectively. Due to the many input parameters, a full visualization of the reward space is not possible, but Fig. 4 gives an overview of six different representative setups of the reward function for an exemplary environment with 41 cutters (n_{c_tot}), a t of 0.85 (i.e. there must be at least 35 cutters with a cutter life >0), and α, β, γ and δ being set to 0.1, 0.65, 0.1 and 0.15 respectively (values determined by trial and error with the goal to achieve a comprehensible agent behavior).

The tunnel simulation environment and reward function were implemented by extending the open AI gym environment API [35].

3.2. Agent

The agent has the task of observing the state of the cutters and, based on that, performing one of three actions for each cutter: i) do nothing, ii) replace the cutter with a new one, or iii) move the current cutter to a new position and replace the original position. Moving of cutters can only be done towards the center of the cutterhead as it is done on construction sites where cutters are reused (Fig. 5).

The implementation of these three possible actions per cutter leads to a large action space of shape $n_{c\_tot} * n_{c\_tot}$ since for every cutter there must be n_{c_tot} possibilities to move to. The action space therefore is implemented as a flattened n_{c_tot} by n_{c_tot} matrix (a vector of size $n_{c\_tot} * n_{c\_tot}$) of indices of cutters. If the i^{th} cutter should be replaced then the matrix at

Table 1
Chances of 1–4 randomly selected cutters to fail within a stroke due to the occurrence of blocky rock mass conditions.

Range of FPI_{blocky}	Rock mass structure designation acc. to Delisio et al. (2014) [35]	Chance of cutters to fail
> 300	massive	0
300–200	blocky	1/100
200–100	blocky/very blocky	1/50
100–50	very blocky	1/10
< 50	blocky/disturbed	1/100

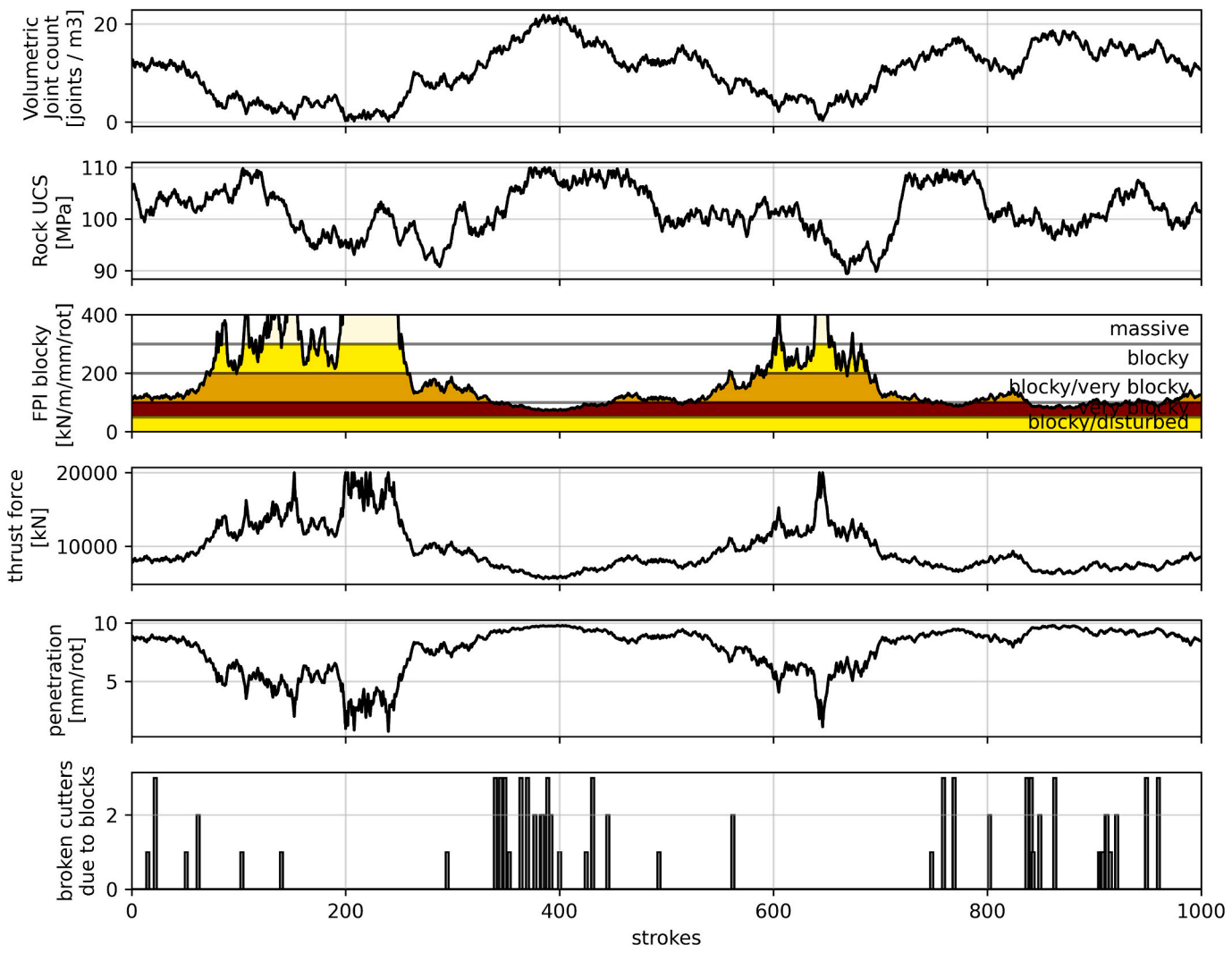


Fig. 3. 1 exemplary episode with 1000 strokes of generated data as the base for the reinforcement learning simulation. Row 1: volumetric joint count generated by a random walk with boundaries; Row 2: intact rock UCS generated by a random walk with boundaries; Row 3: computed FPI_{blocky}; Row 4: computed TBM thrust force; Row 5: computed TBM penetration; Row 6: broken cutters per episode due to the occurrence of blocky rock mass behavior in individual strokes.

position [i, i] must be set to 1 while the other indices remain 0. If the i^{th} cutter is to be moved to position j, then then the matrix at position [i, j] must be set to 1 while the other indices remain 0. If a cutter is not to be replaced or moved, then all indices in the respective cutter's row need to be set to 0. A graphical representation of an exemplary action with only five theoretical cutters for illustrational purposes is given in Fig. 6.

Given this large action space and the requirement that the agent needs to be suited for multi-action-selection problems limits the choice of possible agent architectures. The following RL-agent types were selected for testing in the given environment: Advantage Actor Critic (A2C) [28], Deep Deterministic Policy Gradient (DDPG) [29], Proximal Policy Optimization (PPO) [27], Soft Actor Critic (SAC) [31] and Twin Delayed DDPG (TD3) [30]. The StableBaselines3 [36] implemented version of these algorithms was used as a base for subsequent, extensive architecture tuning and hyperparameter optimization with the Optuna framework [37]. The design of the experimentation framework was inspired by the architecture of the RL Baselines3 Zoo [36]. The problem that RL algorithms' performance is highly dependent on the chosen agent architecture and hyperparameters is also underlined by this study. Only the extensive optimization of these algorithms made it possible to find setups that showed reasonable behavior. In-depth technical information on the agents' architecture, the training, and the parameter optimization can be found in Appendix A.

4. Results

4.1. Reached performance

The decision-making agent for cutter disk maintenance is an RL model developed using extensive optimization of agent-architecture and a wide range of hyperparameters for each algorithm. To ensure the reliability of the agent's decisions, we compared them against a manual process through expert domain reviews. The best-performing agents show reasonable, sometimes innovative, behavior and have learned a cutter-changing policy that could be utilized to automate the cutter-changing process in actual TBM excavations scenarios.

In addition to tuning the hyperparameters of each algorithm, the network architecture of the multi-layer perceptron model (MLP) played a crucial role in the optimization process. Unlike in supervised learning setups where hyperparameter tuning typically leads to minor performance increases, here, it made the difference between a model with low rewards and meaningless actions and models that approached the theoretical maximum reward value of 1000. The critical significance of hyperparameter tuning in RL has been well-documented in recent studies [38,39]. Modifying the network design from the default two-layer MLP with 64 nodes to an eight-layer network with 1024 nodes yielded the most significant improvements, particularly in the case of

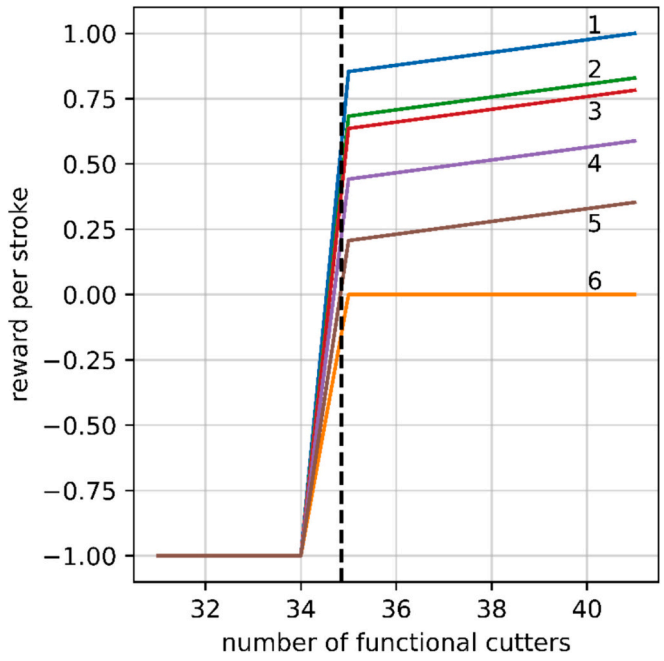


Fig. 4. Exemplary values of the reward function for an environment with 41 cutters. At least 35 functional cutters are required to achieve a reward > -1 (dashed line). Individual rewards are: 1) no replaced cutters, no moved cutters, no bearing failure; 2) 5 replaced cutters, no moved cutters, no bearing failure; 3) no replaced cutters, 5 moved cutters, no bearing failure; 4) 10 replaced cutters, 10 moved cutters, no bearing failure; 5) 5 replaced cutters, 20 moved cutters, no bearing failure (see Fig. 10 for this reward in action); 6) no replaced cutters, no moved cutters, bearing failure.

the TD3 algorithm. For detailed information on the parameters of the best-performing TD3 model, refer to Table B.1. in Appendix B. It outlines the key parameters that contributed to its performance. In Appendix A, we provide a comprehensive description of the hyperparameter optimization process.

The optimization of agents utilized the environment settings from Table 2, tailored for a standard TBM machine in an infrastructure tunnel. The framework, tested across various tunnel diameters and cutter numbers, is applicable to typical infrastructure TBM machines, accommodating a wide range of parameters. Parameters that represent actual values of TBM excavation were set based on the authors' experience in tunnelling. The $\alpha, \beta, \gamma, \delta$ weighting factors in the reward function (see eq. 6), summing to 1.0, were determined through an iterative trial and error process. The aim was to balance the penalties associated with each factor, thereby achieving a confident learning process and an agent performance that closely mimics human behavior. Results are considered to be reasonable if they led to a stable learning process and a comprehensible agent performance, while unreasonable results led to erratic or inconsistent agent behavior, or a non-converging learning process.

Table 3 summarizes the performance of the five RL algorithms employed in training the agents. Clear distinctions emerge, with off-policy algorithms, TD3 and DDPG, outperforming the on-policy counterparts in terms of reward, training time, and practical performance metrics. Among them, TD3 exhibits the best performance, achieving a reward of 942 (theoretical max: 1000), which aligns with its superior results for other metrics. From a computational perspective, TD3 and A2C were the most efficient ones to train in the used framework and SAC was the least efficient (several magnitudes slower than TD3). Given the large number of required training episodes, the computational demand is crucial for successful training. A noticeable discrepancy arises between off-policy and on-policy algorithms when considering practical performance metrics. The TD3 agent demonstrates a lower average cutter replacement rate (0.028 per stroke) and focuses on relocating worn cutters to positions with lower wear in the center of the cutterhead (1.66), resulting in 1.35 broken cutters per stroke. The TD3 agent's ability to manage various degrees of wear in cutters showcases its innovative behavior. Even though DDPG achieves the second highest performance in Table 3, much fewer DDPG trials achieved such a high reward compared to TD3.

On the other hand, the on-policy A2C avoids broken cutters but uses more effort in both replacement (29.8) and movement (11.2) of cutters. These findings are considered while assessing the plausibility of decisions and their practical utility in real-life scenarios. In today's practice, the widespread replacement of cutters per stroke is not economically sustainable. Nonetheless, such results may hold significance in settings where broken cutters are prohibited or in specific cases defined by contractual obligations, machine capabilities, or geological factors.

Fig. 7 visualizes the learning paths of 515 runs using different parameters for the TD3 algorithm, showcasing the substantial variability in outcomes. Many parameter combinations lead to poorly performing models with negative or low rewards. Three main trajectories emerge: i) starting low and gradually reaching the maximum reward, ii) starting similarly but stabilizing to a local optimum around 120 due to convergence to a suboptimal policy, and iii) exhibiting oscillations in the lower region. These trajectories emphasize the significance of parameter selection in achieving optimal performance.

Fig. 8 compares the performance of the different algorithms for the five best performing agents for each RL-algorithm. The reward development depicts the learning path of the agents. It is worth noting the disparity in the number of training episodes required to achieve maximum performance. Notably, off-policy algorithms exhibit faster learning and reach higher levels of performance compared to on-policy algorithms.

Fig. 9 provides further insights into the learning process of the best performing agent, specifically demonstrating how the learning process correlates with improved maintenance outcomes in terms of reduced broken cutters, replaced cutters, and moved cutters during the excavation of a tunnel using a TBM equipped with 41 cutters per episode. The metrics show average values for a single episode consisting of 1000 strokes, each spanning 1.8 m in length, during which a tunnel is excavated. It is important to note that these metrics can vary significantly within the episode. From the summarized metrics presented in Table 3, we observe that the TD3 algorithm achieves a mean value of 1.35 broken cutters per stroke. Fig. 9 shows how the learning process of the TD3 algorithm leads to fewer broken cutters, reduced replacements, and minimized movement of cutters during tunnel excavation.

The learning process in the TD3 algorithm shows a pattern where maximum performance coincides with the lowest loss value for the actor around episode 1200. However, beyond this point, the performance becomes unstable and deteriorates. This pattern emphasizes the importance of selecting a trained agent at the appropriate episode to ensure optimal performance. The observed pattern of deteriorating performance and increasing or chaotic loss after reaching maximum reward is a common phenomenon in training neural networks, indicating overfitting to outliers and fine-grained details in the training data. In TD3, the actor learns to select actions based on a policy that maximizes the expected cumulative reward, while the critic estimates the value that assesses the quality of different actions within a given state. Overfitting in RL refers to the actor becoming excessively specialized to the training environment, thereby struggling to generalize to new states or environments [22]. Increasing or unstable loss indicates a degradation in performance when encountering new, unseen states or environments. Like supervised learning, the training process aims to enhance the generalization capability of the trained agent, enabling it to perform well in unseen environments or states.

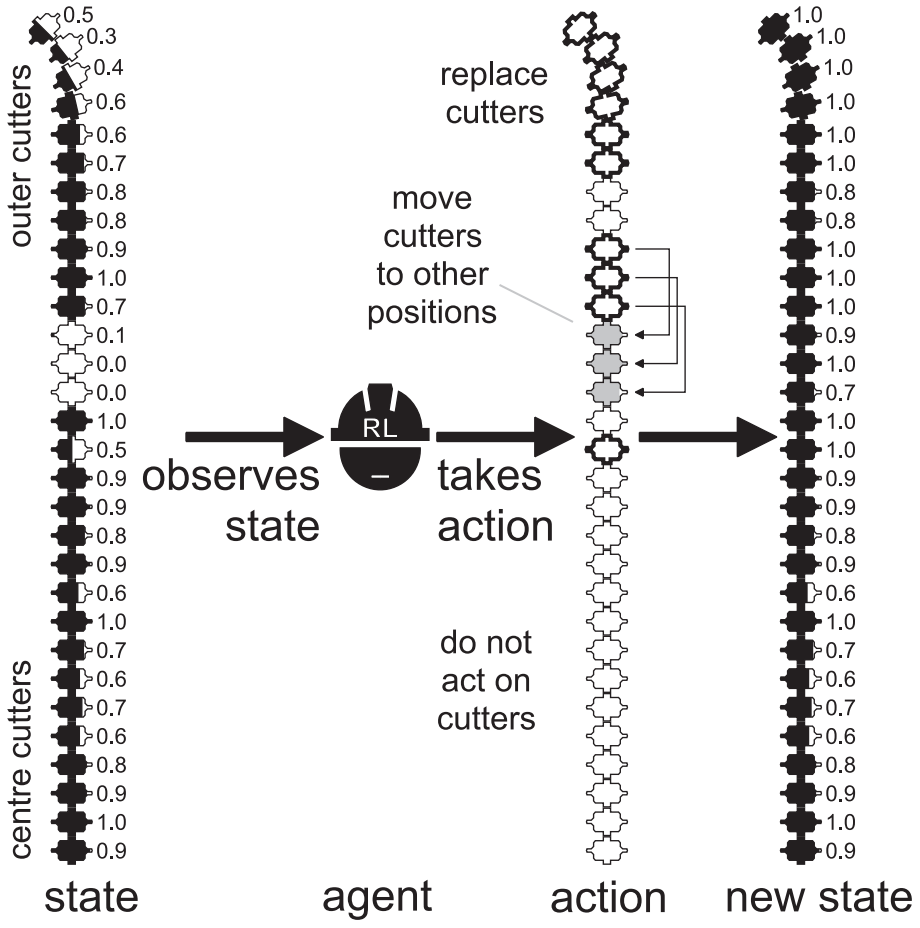


Fig. 5. For every cutter, the agent must choose between three actions: i) do not act on the cutter; ii) replace the cutter; iii) move the cutter to another position towards the center and replace the original position. The numbers besides the graphical representation of the state show the cutter life in the numerical range between 0 and 1.

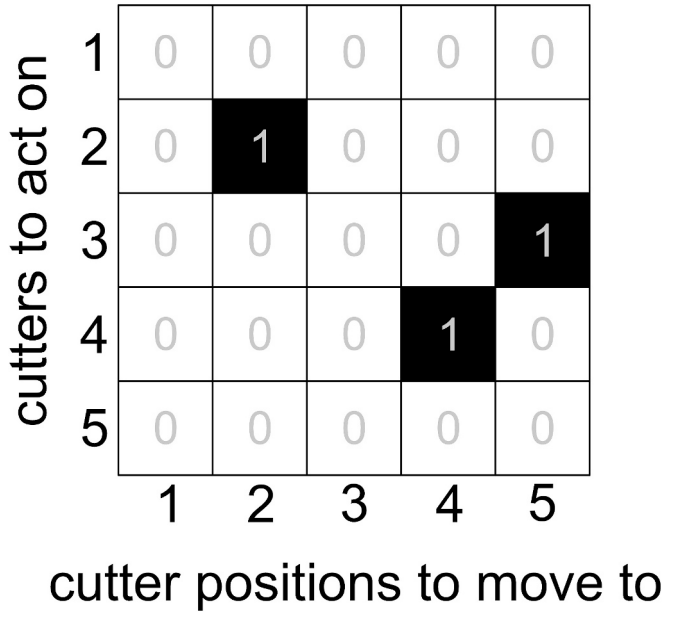


Fig. 6. Exemplary action with 5 cutters only. Cutters 1 and 5 are not acted on. Cutters 2 and 4 have been replaced and cutter 3 is being moved from position 3 to position 5.

Table 2
Environment settings that were used for the optimization of the agents. A description of the parameters is given in section 3.1, and a summary of all symbols in the Appendix.

Parameter	Value
D [m]	8
n_{c_tot} [−]	41
c_l [m]	40,000
n_s [−]	1000
l_s [m]	1.8
t	0.85
$\alpha, \beta, \gamma, \delta$	0.1, 0.65, 0.1, 0.15

4.2. Agent behavior at maximum performance

When the best agent's behavior is analyzed, the general environmental- and especially the settings of the reward function must be considered (see section 3.1) since these are highly influential on how the agent's final policy is. The parameters of Table 2 in combination with the optimized TD3 agent as described in the previous section led to the following behavior. Fig. 10 visualizes the decisions made by a trained TD3 agent in excavating 300 strokes of 1.8 m (540 m of sequential tunnel excavation) for one exemplary rock mass environment.

The most obvious feature of the learned policy is that to the agent avoids entering the cutterhead excessively often. Typically, there are 15–40 strokes between each maintenance effort. When the reward has lowered for several strokes, due to the wearing of cutters, the agent

Table 3
Summarized metrics for experiments of the top performing agents of each of the five algorithm types. Off = off-policy algorithm, on = on-policy algorithm.

Algorithm	Maximum Reward	Episode num max reward	Number of trials for the algorithm	Avg. replaced cutters	Avg. moved cutters	Avg. broken cutters
TD3 (off)	945	1264	515	0.028	1.66	1.348
DDPG (off)	879	400	393	0.002	2.13	1.346
A2C (on)	650	3448	324	29.8	11.2	0
PPO (on)	637	5296	293	35.04	1.23	0.775
SAC (on)	205	468	41	0.862	34.3	0.14

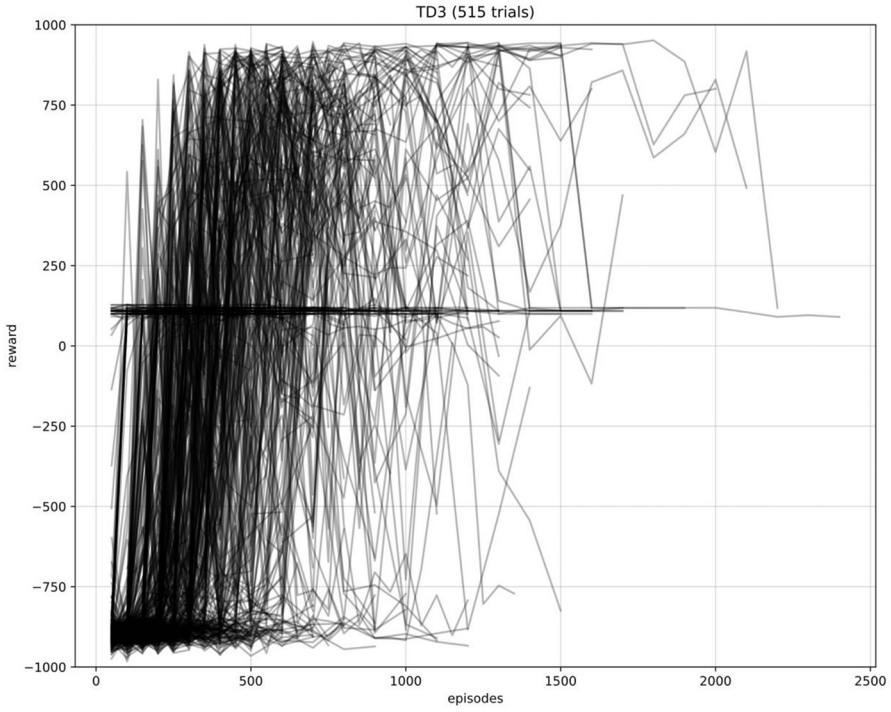


Fig. 7. Learning paths for 515 trials of training a TD3 agent.

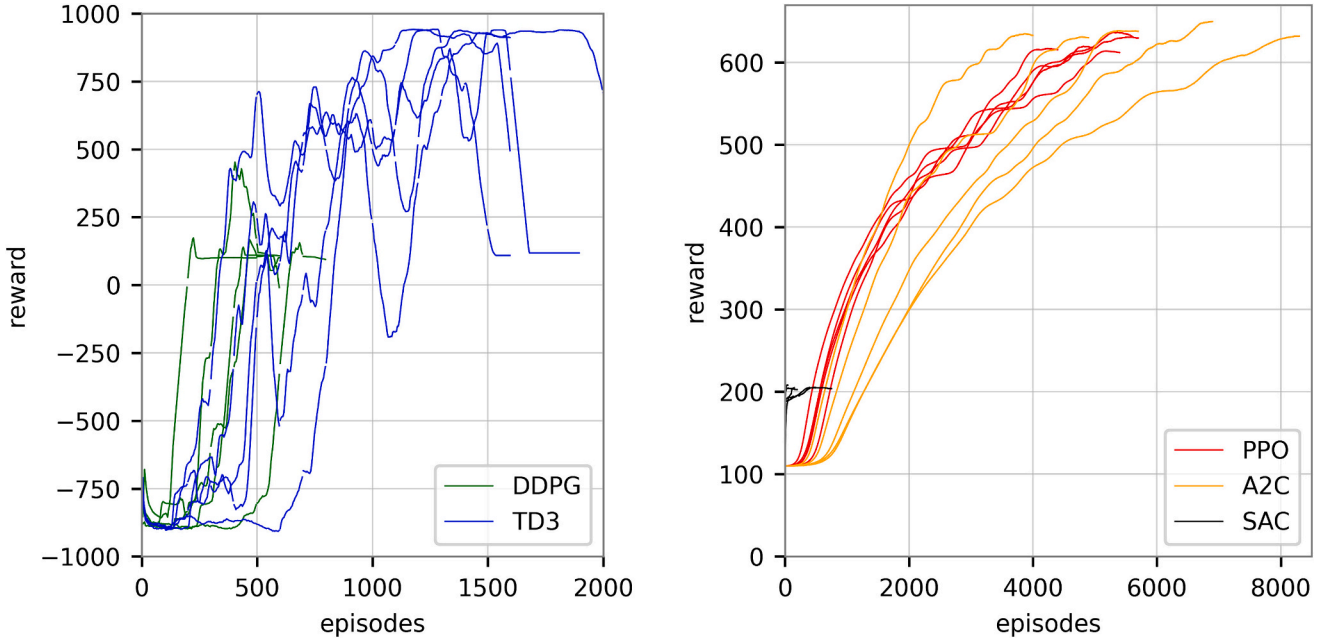


Fig. 8. Best five performing agents for each of five algorithms, off-policy algorithms to the left and on-policy algorithms to the right (note the different x- and y-scales of the left and right figure).

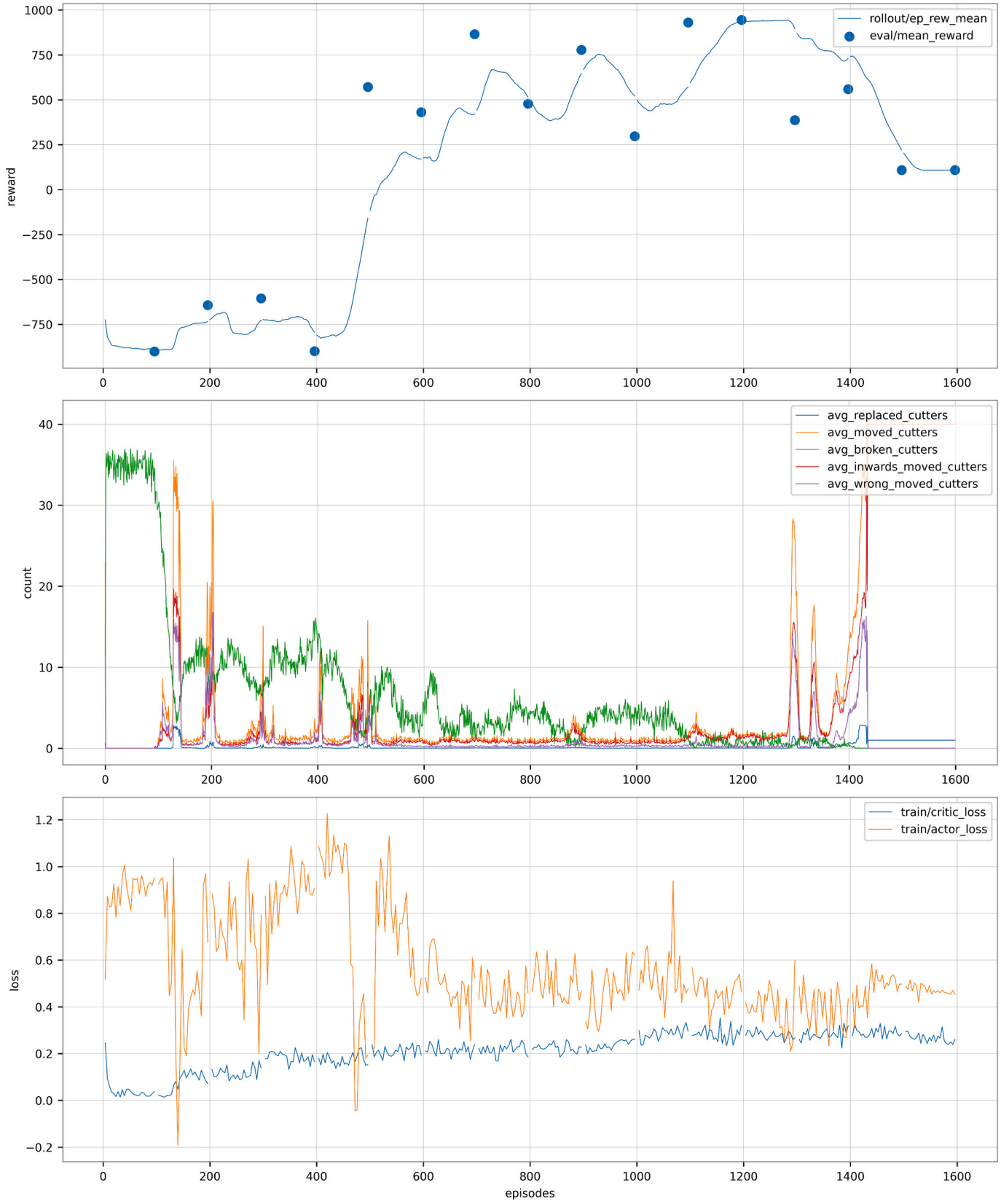


Fig. 9. Exemplary training process for best performing agent, using the TD3 algorithm. Top row shows the reward development, middle row the development of maintain status indicators for TBM, lower describes development in algorithm loss value.

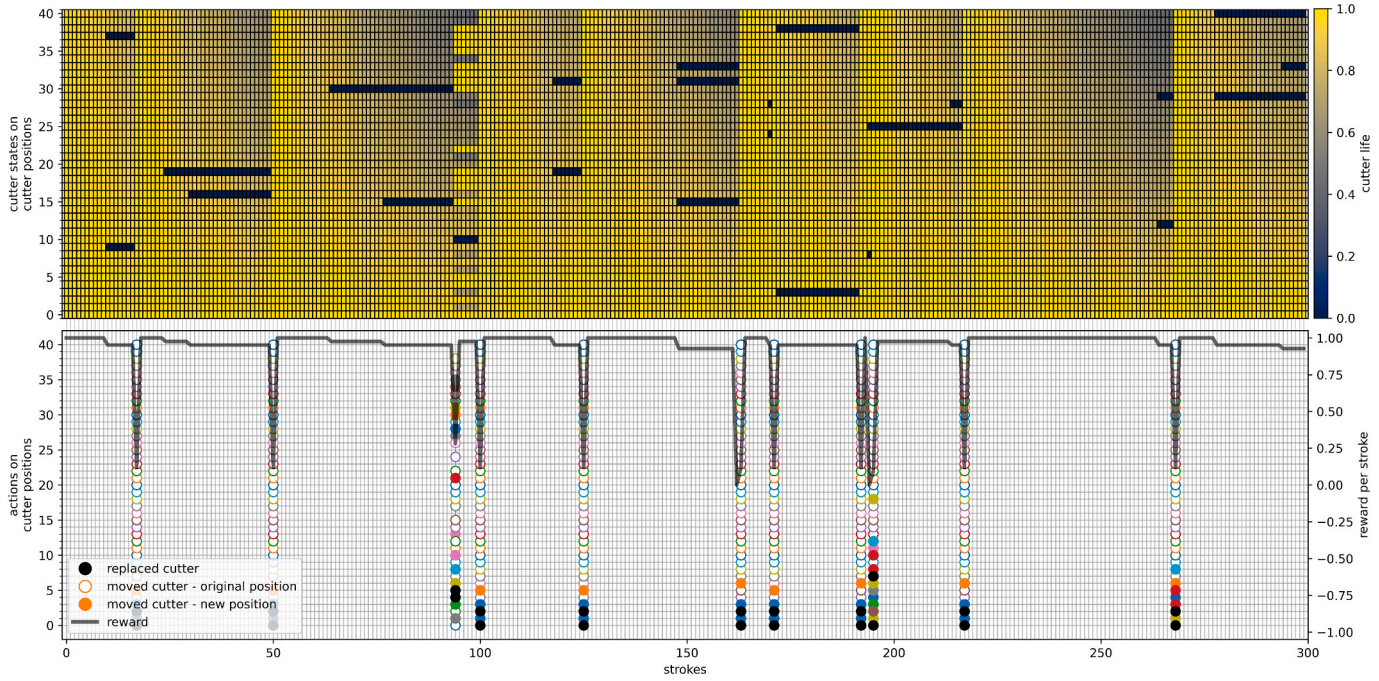


Fig. 10. Decisions made by a trained TD3 agent excavating 300 strokes of 1.8 m (540 m sequential tunnel excavation). In every stroke (visualized by a vertical line) we see the status of the reward, the wearing of each of the 41 cutters, and an eventual action made on each of the cutters. Note the pattern of replacement of center cutters (lower numbers) and movement of outer cutters.

decides to act, perform maintenance (with a cost illustrated in the drop of the reward), thus making sure the cutterhead is in good shape and a high reward is re-established. All actions on cutters (i.e., replacing and moving of cutters) are bundled together and the agent tends to act on almost all cutters simultaneously. This is a strong improvement over the behavior of the agents in the preliminary study which were much more random in [11]. Another improvement over underperforming agents is that these sometimes focus on specific individual cutters only, while this behavior is not given in the best-performing TD3 agents. With the given set of environmental parameters, another behavior that can be observed is that the agent has learned to favor moving cutters instead of replacing them except for the centermost cutters which are rather replaced than moved. To a certain extent the agent accepts up to three simultaneous broken cutters for some strokes but hardly ever more than one single broken cutter on the cutterhead (visualized as black horizontal lines for cutter life in Fig. 10). A strange behavior that can be occasionally observed is that the agent sometimes acts in two consecutive strokes and performs multiple moves and replacements in both.

Fig. 11 shows a visualization of different recordings for running a trained TD3 agent for a full exemplary episode, enriched with more metrics and alternative presentations than in Fig. 10. The above-described agent behavior is well visible.

4.3. Overall cutter changing policy analyses

To analyze the action space, 30,000 individual actions from 30 episodes where the best performing agent was executed were collected. The individual action-vectors have the size $n_{c\_tot}{}^{*}n_{c\_tot}$, which in the case of the environment setup of Table 2 equals to 1681-dimensional action vectors. To analyze these high dimensional actions, the dimensionality reduction algorithm "t-distributed stochastic neighborhood embedding" (t-SNE) [40] was used to project the 30,000, 1681 dimensional actions down to a 2D map. The "Uniform Manifold Approximation and Projection for Dimension Reduction" (UMAP) [41] algorithm was also tested but produced very similar results as t-SNE and thus the t-SNE representation of the data shall be shown here as it is the more widely used algorithm. The most influential parameter for the outcome of t-SNE is the "perplexity" which controls to which extent the algorithm balances the attention between the local and the global aspects of the data [42]. The best results – in terms of clearness of the structure of the t-SNE map – were found with a high perplexity of 1000 which is related to the large number of datapoints and also to the goal to find out more about the global relationship and patterns within the actions.

t-SNE often produces representations of the data in the form of clear clusters in cases of inherently categorical data (e.g., Fig. 2 in van der Maaten and Hinton (2008) [40]) or "clouds" of datapoints with different densities when there is no clear grouping within the original data (e.g., extended data Fig. 1 in [43]). Multiple t-SNE embeddings of the 30 k actions show a recurring and consistent pattern, which can be described as two clusters that each have a pronounced linear geometry that indicates an action space with a linear continuous topology (Fig. 12). Where the smaller cluster represents actions where cutters have been acted on, the larger cluster represents actions without any actual agent activity. Within the small cluster of actions that contain cutter changes and replacements, these actions represent a continuous transition from "low average cutter life → many cutter movings and replacements" to "high average cutter life → few cutter movings and replacements". Different color coding in Fig. 12 b)-d) show this clearly and illustrate how the reward is high for actions that do not require any replacement of moving of cutters and lower for actions that do.

5. Discussion

The results of the previous section show that while reasonable performance can be achieved, the learning process is unstable and sensitive to the chosen input parameters. The environment setting of Table 2 was chosen based on the authors' experience. However, future studies are encouraged to investigate other parameters and especially weight combinations of α, β, γ, δ. The inherent instability in the learning process (which can be attributed to the exploration setup of the algorithm, necessary for discovering new policies) is the necessity to train multiple agents using the best parameters and subsequently select the best performing one. Retraining an agent using optimized parameters does not guarantee good performance. In our experiments, it was common to

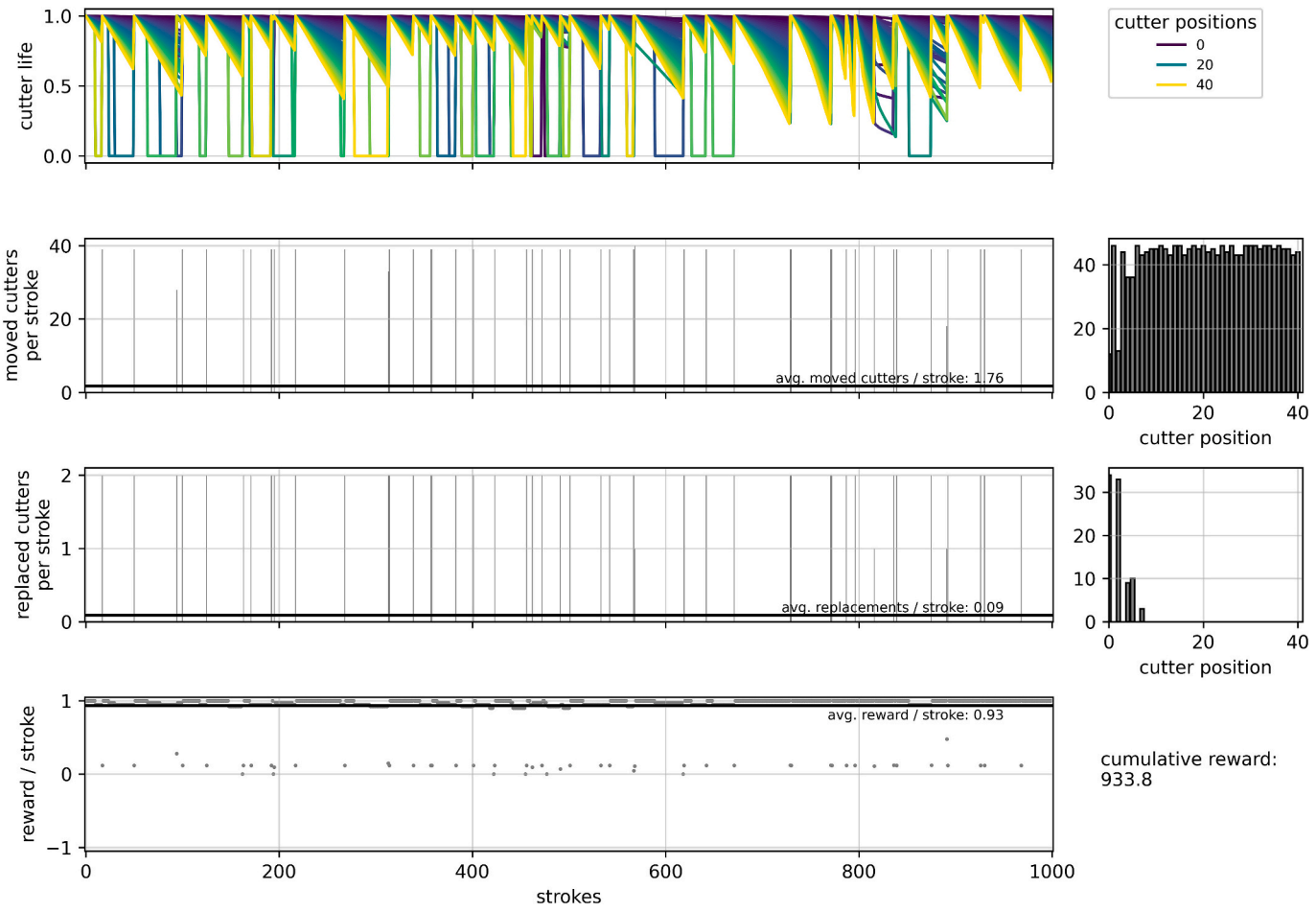


Fig. 11. Visualization of the recordings of one exemplary episode. The top row shows the cutters' life with a color coding acc. to cutter positions. The second and third rows show the number of moved and replaced cutters (with a vertical line at a certain stroke) respectively and the histograms to the right show the number of moves / replacements per cutter position. The fourth row shows the received reward per stroke and the total cumulative reward of the episode to the right.

observe that 9 out of 10 trials resulted in poorly performing agents, despite utilizing the best parameters. However, this situation is not necessarily problematic since some of the trials yield well-performing agents that can be saved and utilized for execution.

By training multiple agents and selecting the best-performing one, we increase the likelihood of obtaining an agent that demonstrates satisfactory performance. This approach accounts for the instability and unpredictability inherent in the learning process and ultimately enables the identification of agents with high performance.

It is observed that the highest-performing agents are bundling their cutter maintenance efforts together as far as possible. The reason for bundling of maintenance efforts is seen in the reward function's (eq. 6) incentives to change cutters in series and the general penalty of entering the cutterhead (fourth and fifth term of eq. 6).

The reason for the agents' affinity towards moving cutters instead of replacing them is seen in the fact that the reward function raises an incentive for moving and the fact that moving a cutter also replaces the original position with a new cutter. Thus, the agent might perceive this as "cheaper" than just replacing the cutter. The observation that center cutters are replaced rather than moved can be interpreted with the setting of the reward function, which favors moving cutters from the outside to the inside of the cutterhead where the wear is lower.

The observed behavior that agents sometimes act on two consecutive strokes is presumably related to the current setting for bearing failure (see reward function condition 2 in section 3.1) since this behavior can often be observed close to an occurred failure due to blocky rock mass behavior. As given in section 3.1, a failure of a cutter due to blockyness should be responded to with an immediate change of this cutter to avoid a bearing failure. Since cutter failure due to blockyness only affects single cutters, which would need immediate changing, this conflicts with the overall incentive not to change individual cutters and thus creates a hard-to-solve situation for the agent that would require a special behavior for single "exotic" events.

6. Conclusions

The study examined the use of reinforcement learning (RL) algorithms to optimize cutter-changing policies in Tunnel Boring Machine (TBM) excavation, presenting a computational framework adaptable to various TBM operations. Extensive experimentation and analysis provided insights into the performance and effectiveness of different RL algorithms in this context. The findings highlight the importance of hyperparameter optimization in achieving successful RL models. Due to the low reproducibility of trained agents, a strategy involving training 10–100 agents and selecting the best-performing agent was successfully employed. Notably, among the algorithms studied, the off-policy TD3 model showed high performance, achieving substantial rewards and plausible actions.

The trained agent adopted a cutter maintenance approach around every 20 strokes, prioritizing the relocation of worn cutters to areas with less wear on the cutterhead rather than immediately replacing them. This strategy resulted in an average of 1.35 broken cutters per stroke, considering a TBM equipped with 41 cutters. Note that the simulated rockmass can be seen as "extremely abrasive" due to these numbers, but

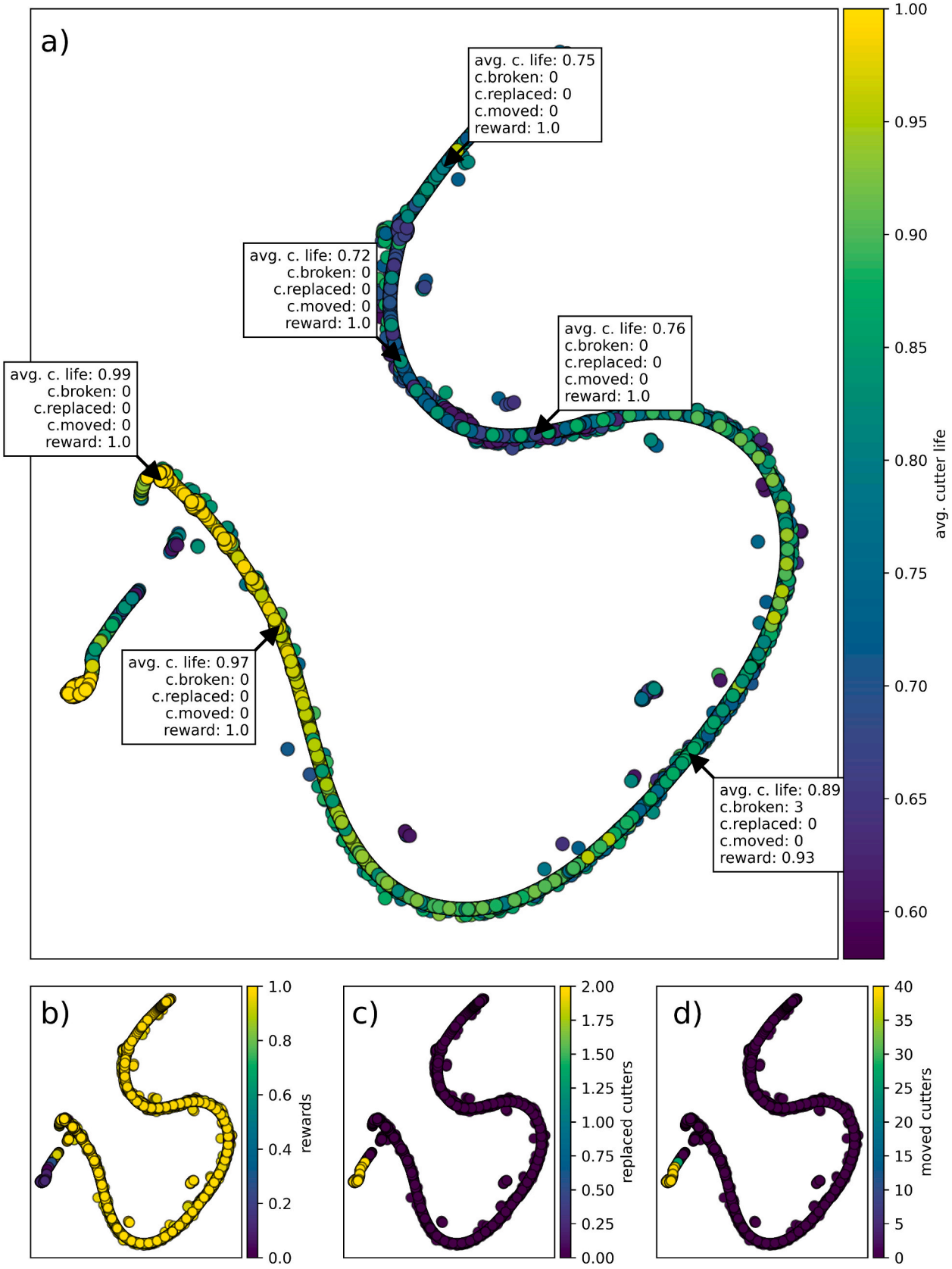


Fig. 12. t-SNE dimensionality reduction of 30 k actions of the best-performing agent. a) color coding acc. to average cutter life of the state that corresponds to the action; b) reward received after the action; c) number of replaced cutters as a result of the action; d) number of moved cutters as a result of the action.

the simulation can also be adjusted to account for less abrasive rockmass conditions. Changing the size of the cutterhead / the number of cutters is possible, but it must be considered that an increasing number of cutters results in an exponentially growing action space (see section 3.2) and thus vastly increasing computational demand and complexity of the problem for the agent to solve. The reward function's weightings can be adjusted to adapt the policy to specific site conditions by creating higher incentives for e.g. moving cutters instead of replacing them with the weighting factors α, β, γ and δ (see section 3.1). The development and implementation of an RL-based recommender system for cutter-changing policies in TBMs offer benefits for various stakeholders, including: i) TBM site operators: The use of RL algorithms, such as the high-performing TD3 model, can optimize cutter-changing decisions, resulting in enhanced TBM performance, increased productivity, and reduced downtime. Operators can benefit from improved maintenance strategies, cost savings, and more efficient tunnel excavation processes leading to better resource allocation, improved maintenance scheduling, and reduced manual inspection effort. ii) Project owners and contractors: Optimized cutter changing policies, driven by RL algorithms, can contribute to project cost reductions, improved project timelines, and increased overall project efficiency. This benefits project owners and contractors by ensuring smoother operations and successful project completion.

A practical implementation of the RL-based recommender system for cutter changing policies for TBM tunnelling could involve the following components: i) Real-time data collection: Sensors and monitoring systems installed in the TBM gather data on various parameters such as cutter wear, bearing condition, thrust, torque, vibration, and energy consumption. ii) Data processing and analysis: The collected data is processed and fed as state information to the RL-based recommender system, such as a trained TD3 agent, which evaluates the current cutter status and recommends appropriate cutter changing actions before the next stroke. iii) Decision support interface: The system provides recommendations to TBM operators or maintenance crews through an intuitive user interface, assisting them in making well-informed decisions regarding cutter changing. iv) Continuous learning and improvement: Leveraging human feedback on the agents' decisions can be used to update the agent, and further enhance its recommendations over time.

7. Outlook

While the TunnRL-CC framework can be used as a support for decision-making in real tunnelling today, there are several points that still can be improved in future studies:

i) The reward function contains penalties for excessive work effort for cutter maintenance which is simply based on the cutters' position on the cutterhead (i.e., outer cutters need more effort than inner cutters). The subsequent advancement entails incorporating a sophisticated model that considers the intricate geometry of the cutterhead, including its inner layout featuring multiple bucket chambers. By integrating this enhanced model, the system can incentivize the replacement of cutters within a single bucket chamber while penalizing the effort involved in moving cutters between chambers. (i.e., include a shortest path problem).
ii) The current penetration model after Delisio et al. (2014) [34] is seen as well suited to simulate geological conditions for hard rock excavation in blocky rock mass conditions. Different penetration models could be considered and/or established to get representative simulations in other geological conditions.
iii) The reward function controls the agent's behavior and is the lever that allows adjustment of the learned policy towards construction site specific geological, logistical and economic conditions. Further investigations about the extent to which the reward function's parameters α, β, γ and δ control the learned policy should be conducted in order to achieve site specific tailored policies.
iv) The current study has done extensive optimization of advanced RL models to find an effective setup for the given environment. However, optimizing RL agents is challenging due to the low repeatability of single model runs and the resulting uniqueness of the optimized agents and thus optimization schemes that work for supervised learning are only of limited help for RL. Further investigations about efficient large-scale parametric studies and optimizations of RL algorithms are therefore required.
v) The presented study shows a simulation-driven approach towards agent-based optimization of TBM cutter changing. Simulations were necessary to ensure the computational feasibility of the RL approach but a theoretical proof of concept is given. Including real world data from construction sites is the necessary next step to validate the learned policies and demonstrate that this approach also works under operational conditions.

Moving forward, refining TunnRL-CC with real-world data and tailored optimization will be essential for practical application. The presented cutter changing policy optimization systems TunnRL-CC fits well into recent developments in TBM tunnelling and can be the missing link between automated cutter state monitoring [18] and robotic replacement of cutters in the cutterhead [16]. These enhancements will solidify the role of RL in advancing TBM maintenance and set a pragmatic course towards smarter and more efficient tunnelling operations.

Research data

The full TunnRL-CC software can be found as a locked release version with Python code under the following Github repository: https://github.com/TunnRL/TunnRL_TBM_maintenance/releases/tag/v2.0.0

CRediT authorship contribution statement

Tom F. Hansen: Writing – original draft, Software, Methodology, Investigation, Data curation, Conceptualization. **Georg H. Erharter:** Writing – original draft, Software, Project administration, Methodology, Investigation, Data curation, Conceptualization. **Thomas Marcher:** Writing – review & editing, Conceptualization.

Declaration of competing interest

Declaration of Generative AI and AI-assisted technologies in the writing process.

During the preparation of this work the authors used "Bard: Google's AI research chatbot" (Retrieved from https://bard.google.com/) in order to improve the readability and language of some paragraphs in the abstract and first section. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Data availability

The study is fully based on the developed TunnRL-CC software package that is given in the manuscript's Research Data section. No other data sources were used that must be stated.

Acknowledgments

Prof. Amund Bruland (NTNU) and Mr. Gamal Heikal are thanked for their input during the conceptualization phase of this study. Code review and code design advice from Ida Norderhaug Drøsdal is also highly appreciated.

This research received no specific grant from funding agencies in the

public, commercial, or not-for-profit sectors. Open Access funding was provided by the University of Oslo.

Appendix A. Extended technical description of parameter optimization for different agents

Hyperparameter optimization plays a crucial role in training RL agents, as acknowledged in prior research [38,39]. Taking inspiration from the experimental setup for hyperparameter orchestration and following the guidelines provided in RL baselines3 Zoo [36], we established a hyperparameter tuning framework based on Optuna [37]. This framework incorporated the latest practices, including defining reasonable intervals for hyperparameters and tailoring specific configurations for each algorithm under investigation. Additionally, we developed specialized functions to ensure scalability in network design and learning rates, catering to the unique requirements of the task at hand.

In our experiments, we evaluated the performance of the trained agent at regular intervals, typically every 50–100 episodes (varying for different algorithms). To assess performance, we loaded the best performing agents obtained up until that point and executed their decisions in the environment for the last 10 episodes. Subsequently, we calculated the mean performance as an indicator of the agent's effectiveness. This process allowed us to monitor the agent's progress over time and make informed decisions regarding parameter optimization.

The parallel coordinate plots in Figs. A.1-A.4 provide insights into the sensitivity of various hyperparameters to the final reward. It is evident that TD3 demonstrates robustness in handling a wide range of values for most parameters. In contrast, the other algorithms exhibit optimal results within narrower ranges of parameter values. Despite its generality, TD3 has some parameters crucial for performance that we want to point out:

- The use of more complex networks with additional layers and significantly more nodes compared to the default setup.
- A longer policy delay update period, above 5 steps (default is 2). This addresses a problem in DDPG, which TD3 builds upon.
- Learning starts after around 300 steps (default is 100). In this first period the agent samples valid actions from a random uniform distribution. More steps enhance its exploratory capabilities.
- The absence of noise injection regulates the exploratory capabilities of the agent.

In the case of the best on-policy model, PPO, a distinct band of best performing parameters can be observed. Compared to TD3, the core network architecture for PPO is notably smaller. A characteristic of models that are less dependent on hyperparameters was the lower degree of reproducibility when using the best performing parameters. For example, running TD3 ten times with these parameters resulted in a well-performing model, achieving a reward above 900, only once out of ten attempts. On the other hand, a comparable run using PPO consistently yielded high rewards, over 600, in all ten repeated runs. This suggests that PPO exhibits greater stability and reproducibility when utilizing the best performing parameters compared to TD3, still TD3 performs notably better when it is successful.

During the experiments, it was observed that several trials reached a dead-end trajectory with a local optimum, even when using the exact same parameters as earlier successful runs. Running into local optima is a well-known problem of DDPG and TD3 [44] and was particularly noticeable in the plots of experiment trials of these algorithms, as depicted in Fig. A-5. This phenomenon, along with the necessity of conducting numerous hyperparameter experiments, posed challenges in effectively managing computational resources for numerous experiments necessary. To address this issue, we implemented functionality for parallel runs of Optuna trials using the Joblib library [45]. This allowed multiple trials to be executed concurrently, utilizing different computer nodes. To facilitate reproducibility and streamline the experimentation process on High-Performance Computing (HPC) machines, we containerized the training code using Docker containers. By containerizing the code, we ensured consistent execution environments and simplified the deployment of the training process.

To simplify parameter configuration and ensure effective management of parameters across different experiments, we leveraged the Hydra framework [46]. Hydra allowed us to easily configure and keep track of parameters in various experimental setups. Additionally, we employed MLflow [47] to track metrics for each experiment, enabling us to compare the performance of different runs systematically.

During the execution of each experiment, which could span several hours, we utilized Tensorboard to monitor the development of metrics. This real-time evaluation facilitated the early termination of certain experiments when deemed necessary. To enhance the training process and gain deeper insights, we implemented a range of callbacks. These callbacks interacted with the agent during training, captured the state of the environment, and logged and visualized the training process. By incorporating these callbacks, we were able to gain a comprehensive understanding of the agent's progress and performance throughout the training process.

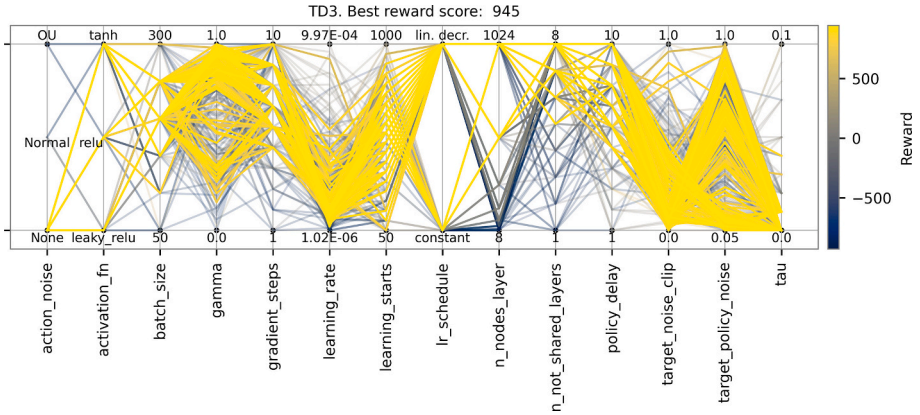


Fig. A.1. Parallel coordinate plot that shows parameter combinations during the optimization of the TD3 algorithm.

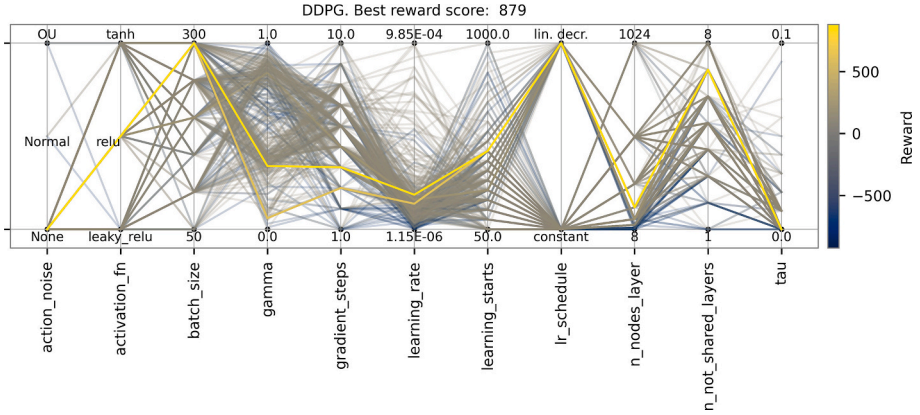


Fig. A.2. Parallel coordinate plot that shows parameter combinations during the optimization of the DDPG algorithm.

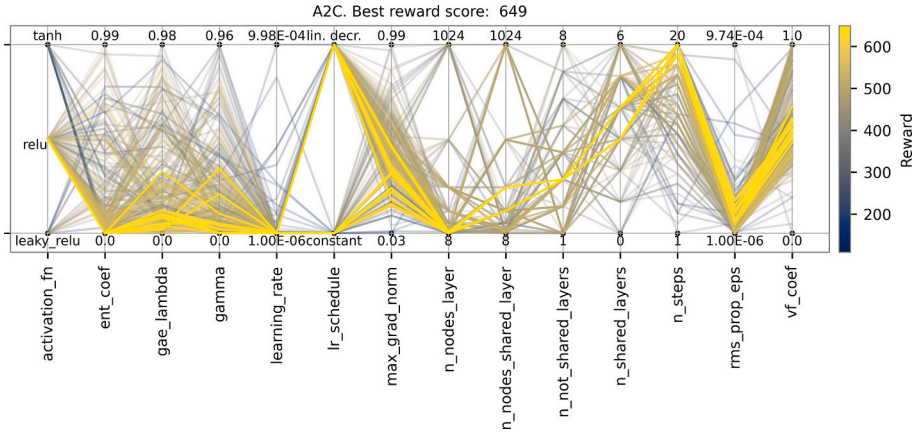


Fig. A.3. Parallel coordinate plot that shows parameter combinations during the optimization of the A2C algorithm.

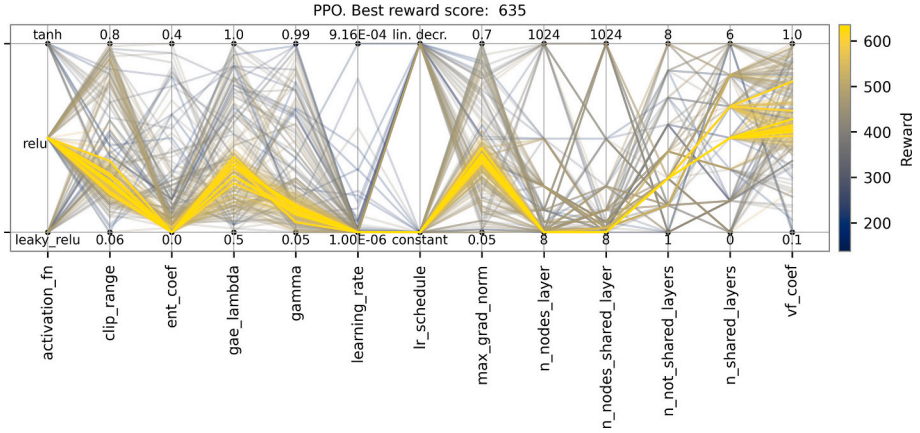


Fig. A.4. Parallel coordinate plot that shows parameter combinations during the optimization of the PPO algorithm.

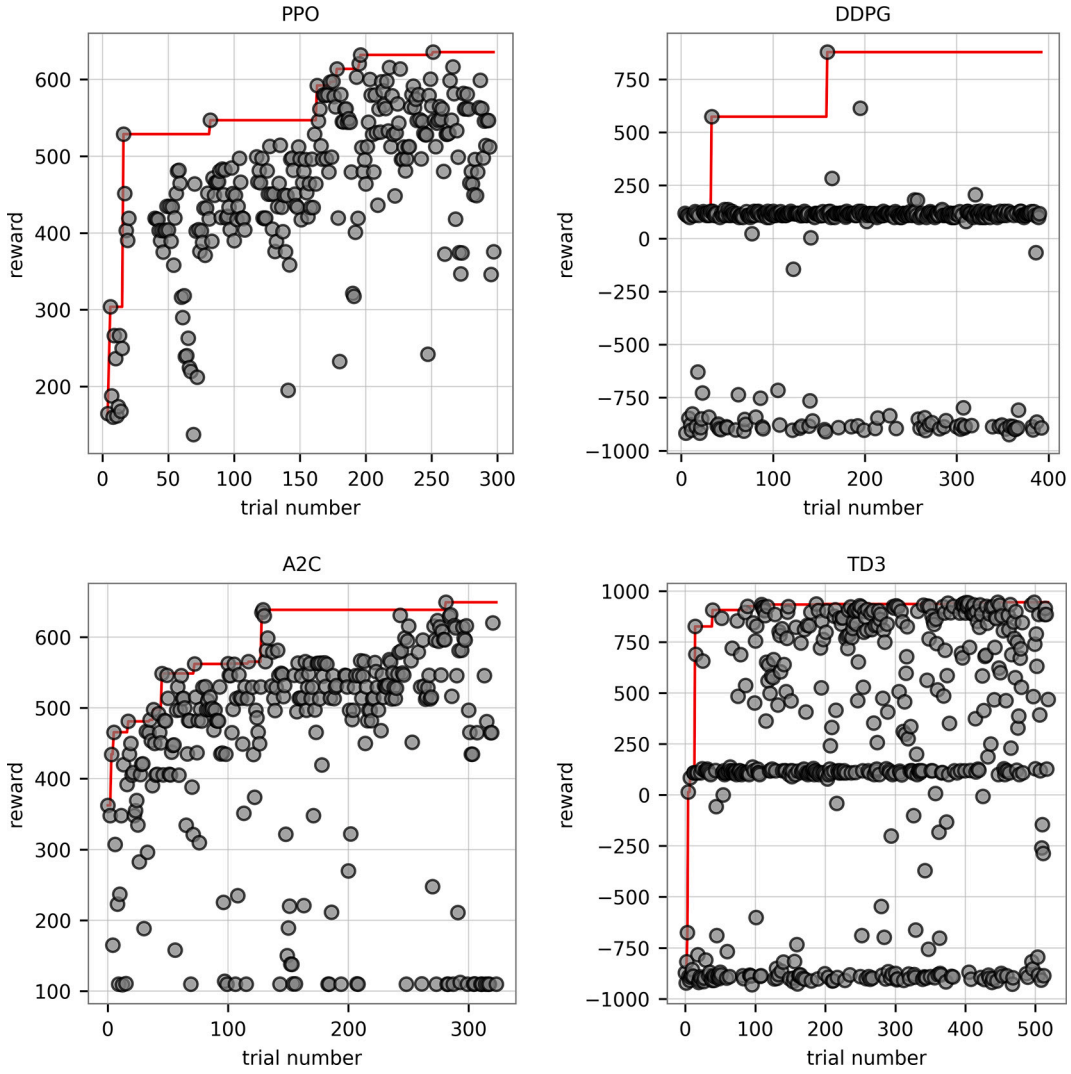


Fig. A.5. Optimization process visualized in trial number versus reward for the same algorithms.

Appendix B. Tables with additional information

Table B1
Optimized hyperparameters for best performing TD3-agent.

Parameter	Value
Action noise	None
Activation function	tanh
Batch size	200
Gamma	0.9766519262714228
Gradient steps	8
Learning rate	8.86666407593999e-05
Learning starts	600
Lr schedule	Linear decrease
Number of nodes in layer	1024
Number of layers	8
Policy delay	10
Target noise clip	0.42580299347844197
Target policy noise	0.09564547952520154
Tau	0.0028384818651399457

Table B2
Overview table of all used symbols.

Symbol	Unit	Explanation
S	–	State vector
n_{c_tot}	–	Total number of cutters on the cutterhead
c_l	[m]	Cutter life expressed as the total rolling distance of one cutter
l_s	[m]	Length of one stroke of the TBM
p	[mm/rot]	Penetration rate of the TBM
n_s	–	Number of strokes in one episode
J_v	[joints / m^3]	Vector of length n_s with values for the volumetric joint count
UCS	[MPa]	Vector of length n_s with values for the intact rock unconfined compressive strength
FPI_{blocky}	[kN/m/mm/rot]	Field penetration index for blocky rock mass conditions acc. to [34]
TF	[kN]	Thrust force of the TBM
D	[m]	Diameter of the TBM cutterhead
r	–	Reward that the agent receives after every stroke
R	–	Cumulative reward of one whole episode
n_{c_good}	–	For reward: number of cutters with a $c_l > 0$
t	[%]	For reward: threshold of minimum required number of cutters with a $c_l > 0$
c_w	–	For reward: linearly weighted sum of all cutters where outer cutters have a higher weight than inner cutters
c_{rw}	–	For reward: linearly weighted sum of all replaced cutters where outer cutters have a higher weight than inner cutters
c_{mw}	–	For reward: linearly weighted sum of all moved cutters where outer cutters have a higher weight than inner cutters
d_c	–	For reward: total distance between all cutters that have been acted on expressed as index difference
c	–	For reward: penalty for effort of having to enter the cutterhead
$\alpha, \beta, \gamma, \delta$	–	For reward: weighting factors to adjust penalties to different construction economic conditions.

References

[1] K. Gehring, Leistungs- und Verschleißprognosen im maschinellen Tunnelbau, Felsbau (1995) 439–448.
[2] K. Thuro, Geologisch-felsmechanische Grundlagen der Gebirgslösung im Tunnelbau: Habilitationsschrift, in: Münchner Geologische Hefte, Reihe B, 2002, pp. 1–160.
[3] L. Wang, Y. Kang, Z. Cai, Q. Zhang, Y. Zhao, H. Zhao, P. Su, The energy method to predict disc cutter wear extent for hard rock TBMs, Tunn. Undergr. Space Technol. 28 (2012) 183–191, https://doi.org/10.1016/j.tust.2011.11.001.
[4] J. Hassanpour, J. Rostami, S. Tarigh Azali, J. Zhao, Introduction of an empirical TBM cutter wear prediction model for pyroclastic and mafic igneous rocks; a case history of Karaj water conveyance tunnel, Iran, Tunn. Undergr. Space Technol. 43 (2014) 222–231, https://doi.org/10.1016/j.tust.2014.05.007.
[5] R.J. Plinninger, M. Alber, J. Düllmann, Rock mass-scale factors with an influence on tool wear in the mechanised tunnelling process in hard rock, Geomechanik und Tunnelbau 11 (2018) 157–168, https://doi.org/10.1002/geot.201700068.
[6] L. She, S. Zhang, C. Wang, Y. Li, M. Du, A new method for wear estimation of TBM disc cutter based on energy analysis, Tunn. Undergr. Space Technol. 131 (2023) 104840, https://doi.org/10.1016/j.tust.2022.104840.
[7] N. Zhang, S.-L. Shen, A. Zhou, A new index for cutter life evaluation and ensemble model for prediction of cutter wear, Tunn. Undergr. Space Technol. 131 (2023) 104830, https://doi.org/10.1016/j.tust.2022.104830.
[8] Y. Liu, S. Huang, G. Di Wang, D. Zhang Zhu, Prediction model of tunnel boring machine disc cutter replacement using kernel support vector machine, Appl. Sci. 12 (2022) 2267, https://doi.org/10.3390/app12052267.
[9] X. Shen, X. Chen, Y. Fu, C. Cao, D. Yuan, X. Li, Y. Xiao, Prediction and analysis of slurry shield TBM disc cutter wear and its application in cutter change time, Wear 498-499 (2022) 204314, https://doi.org/10.1016/j.wear.2022.204314.
[10] E. Farrokh, Cutter change time and cutter consumption for rock TBMs, Tunn. Undergr. Space Technol. 114 (2021) 104000, https://doi.org/10.1016/j.tust.2021.104000.
[11] G.H. Erharter, T.F. Hansen, Towards optimized TBM cutter changing policies with reinforcement learning, Geomechanics and Tunnelling 15 (2022) 665–670, https://doi.org/10.1002/geot.202200032.
[12] G.H. Erharter, T.F. Hansen, Z. Liu, T. Marcher, Reinforcement learning based process optimization and strategy development in conventional tunneling, Autom. Constr. 127 (2021), https://doi.org/10.1016/j.autcon.2021.103701.
[13] N.S. Kedir, S. Somi, A.R. Fayek, P.H. Nguyen, Hybridization of reinforcement learning and agent-based modeling to optimize construction planning and scheduling, Autom. Constr. 142 (2022) 104498, https://doi.org/10.1016/j.autcon.2022.104498.
[14] A. Biniyaz, B. Azmoon, Z. Liu, Intelligent control of groundwater in slopes with deep reinforcement learning, Sensors (Basel) 22 (2022), https://doi.org/10.3390/s22218503.
[15] E. Soranzo, C. Guardiani, W. Wu, The application of reinforcement learning to NATM tunnel design, Underground Space (2022), https://doi.org/10.1016/j.undsp.2022.01.005.
[16] L. Du, J. Yuan, S. Bao, R. Guan, W. Wan, Robotic replacement for disc cutters in tunnel boring machines, Autom. Constr. 140 (2022) 104369, https://doi.org/10.1016/j.autcon.2022.104369.
[17] J. Yuan, R. Guan, J. Du, Design and implementation of disc cutter changing robot for tunnel boring machine (TBM), in: 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2019, pp. 2402–2407. doi:https://doi.org/10.1109/ROBIO49542.2019.8961494.
[18] H. Lan, Y. Xia, Z. Ji, J. Fu, B. Miao, Online monitoring device of disc cutter wear – design and field test, Tunn. Undergr. Space Technol. 89 (2019) 284–294, https://doi.org/10.1016/j.tust.2019.04.010.
[19] A. Mahmoodzadeh, M. Mohammadi, H. Hashim Ibrahim, S. Nariman Abdulhamid, H. Farid Hama Ali, A. Mohammed Hasan, M. Khishe, H. Mahmud, Machine learning forecasting models of disc cutters life of tunnel boring machine, Autom. Constr. 128 (2021) 103779, https://doi.org/10.1016/j.autcon.2021.103779.
[20] M.-H. Rajati, J. Rostami, H. Memarian, M.-T. Hamzaban, A study on predicting the wear of TBM disc cutters using Cerchar testing, Tunn. Undergr. Space Technol. 140 (2023) 105290, https://doi.org/10.1016/j.tust.2023.105290.
[21] L. Brackmann, A. Röttger, M. Treppmann, S. Weber, The behavior of cutting discs for mechanized tunneling under cyclic loading conditions, Tunn. Undergr. Space Technol. 137 (2023) 105151, https://doi.org/10.1016/j.tust.2023.105151.
[22] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, Second edition, The MIT Press, Cambridge Massachusetts, 2018. ISBN: 9780262039246.
[23] X. Zhao, C. Gu, H. Zhang, X. Yang, X. Liu, J. Tang, H. Liu, DEAR: Deep Reinforcement Learning for Online Advertising Impression in Recommender Systems, arXiv preprint, arXiv:1909.03602, 2019, https://doi.org/10.48550/arXiv.1909.03602.
[24] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M.T. Ribeiro, Y. Zhang, Sparks of Artificial General Intelligence: Early Experiments with GPT-4, arXiv, https://arxiv.org/pdf/2303.12712.pdf?utm_source=webtekno, 2023. (Accessed 20 December 2023).
[25] A. Raghu, M. Komorowski, I. Ahmed, L. Celi, P. Szolovits, M. Ghassemi, Deep Reinforcement Learning for Sepsis Treatment, arXiv preprint, arXiv:1711.09602, 2017, https://doi.org/10.48550/arXiv.1711.09602.
[26] R. Li, Z. Zhao, Q. Sun, C.L.I.C. Yang, X. Chen, M. Zhao, H. Zhang, Deep reinforcement learning for resource Management in Network Slicing 6, IEEE Access, 2018, pp. 74429–74441, https://doi.org/10.1109/ACCESS.2018.2881964.
[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms. http://arxiv.org/pdf/1707.06347v2, 2017.
[28] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T.P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous Methods for Deep Reinforcement Learning, 2016, https://doi.org/10.48550/arXiv.1602.01783.
[29] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: ICLR, 2016, https://doi.org/10.48550/arXiv.1509.02971.
[30] S. Fujimoto, H. van Hoof, D. Meger, Addressing Function Approximation Error in Actor-Critic Methods, in: Proceedings of the 35th International Conference on Machine Learning, 2018, https://doi.org/10.48550/arXiv.1802.09477. Stockholm, Sweden.
[31] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, 2018, https://doi.org/10.48550/arXiv.1801.01290.
[32] B. Maidl, L. Schmid, W. Ritz, M. Herrenknecht, Hardrock Tunnel Boring Machines, Ernst, Berlin, 2008. ISBN: 978–3–433-01676-3.

[33] 1, Hard Rock Tunnel Boring, Fakultet for ingeniørvitenskap og teknologi, Trondheim. https://folk.ntnu.no/pdj/bruland%201998/, 1998 (accessed 20 December 2023).
[34] A. Delisio, J. Zhao, A new model for TBM performance prediction in blocky rock conditions, Tunn. Undergr. Space Technol. 43 (2014) 440–452, https://doi.org/10.1016/j.tust.2014.06.004.
[35] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, OpenAI Gym, 2016, https://doi.org/10.48550/arXiv.1606.01540.
[36] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Stable-Baselines3: reliable reinforcement learning implementations, J. Mach. Learn. Res. 22 (2021) 12348–12355. http://jmlr.org/papers/v22/20-1364.html (accessed 20 December 2023).
[37] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage AK USA, ACM, 2019, pp. 2623–2631. New York, NY, USA. ISBN: 9781450362016, https://doi.org/10.1145/3292500.3330701.
[38] T. Eimer, M. Lindauer, R. Raileanu, Hyperparameters in Reinforcement Learning and How To Tune Them, arXiv preprint, arXiv:2306.01324, 2023, https://doi.org/10.48550/arXiv.2306.01324.
[39] J.K.H. Franke, G. Köhler, A. Biedenkapp, F. Hutter, Sample-Efficient Automated Deep Reinforcement Learning, 2021, https://doi.org/10.48550/arXiv.2009.01555.
[40] L. van der Maaten, G. Hinton, Visualizing Data using t-SNE, J. Mach. Learn. Res. 9 (2008), pp. 2579–2605. https://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf (accessed 20 December 2023).
[41] L. McInnes, J. Healy, J. Melville, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. doi: 10.48550/arXiv.1802.03426.
[42] M. Wattenberg, F. Viégas, I. Johnson, How to use t-SNE effectively, Distill (2016), https://doi.org/10.23915/distill.00002. http://distill.pub/2016/misread-tsne.
[43] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533, https://doi.org/10.1038/nature14236.
[44] Achiam Joshua, Spinning Up in Deep Reinforcement Learning. https://spinningup.openai.com/en/latest/user/introduction.html, 2018 (accessed 13 July 2023).
[45] Joblib Development Team, Joblib: running Python functions as pipeline jobs. https://joblib.readthedocs.io/, 2020.
[46] Omry Yadan, Hydra - a Framework for Elegantly Configuring Complex Applications. https://github.com/facebookresearch/hydra, 2019.
[47] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S.A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, Others, accelerating the machine learning lifecycle with MLflow, IEEE Data Eng. Bull. 41 (2018) 39–45.