energies

MDPI

Article

Intelligent Action Planning for Well Construction Operations Demonstrated for Hole Cleaning Optimization and Automation

Gurtej Singh Saini, Parham Pournazari, Pradeepkumar Ashok and Eric van Oort *

Cockrell School of Engineering, The University of Texas at Austin, Austin, TX 78712, USA
* Correspondence: vanoort@austin.utexas.edu

Abstract: Reactive and biased human decision-making during well construction operations can result in problems ranging from minor inefficiencies to events that can have far-reaching negative consequences for safety, environmental compliance and cost. A system that can automatically generate an optimal action sequence from any given state to meet an operation's objectives is therefore highly desirable. Moreover, an intelligent agent capable of self-learning can offset the computation and memory costs associated with evaluating the action space, which is often vast. This paper details the development of such action planning systems by utilizing reinforcement learning techniques. The concept of self-play used by game AI engines (such as AlphaGo and AlphaZero in Google's DeepMind group) is adapted here for well construction tasks, wherein a drilling agent learns and improves from scenario simulations performed using digital twins. The first step in building such a system necessitates formulating the given well construction task as a Markov Decision Process (MDP). Planning is then accomplished using Monte Carlo tree search (MCTS), a simulation-based search technique. Simulations, based on the MCTS's tree and rollout policies, are performed in an episodic manner using a digital twin of the underlying task(s). The results of these episodic simulations are then used for policy improvement. Domain-specific heuristics are included for further policy enhancement, considered factors such as trade-offs between safety and performance, the distance to the goal state, and the feasibility of taking specific actions from specific states. We demonstrate our proposed action planning system for hole cleaning, a task which to date has proven difficult to optimize and automate. Comparing the action sequences generated by our system to real field data, it is shown that it would have resulted in significantly improved hole cleaning performance compared to the action taken in the field, as quantified by the final state reached and the long-term reward. Such intelligent sequential decision-making systems, which use heuristics and exploration–exploitation trade-offs for optimum results, are novel applications in well construction and may pave the way for the automation of tasks that until now have been exclusively controlled by humans.

Keywords: reinforcement learning; action planning; Monte Carlo tree search; digital twinning; well construction; hole cleaning; heuristic design

Citation: Saini, G.S.; Pournazari, P.; Ashok, P.; van Oort, E. Intelligent Action Planning for Well Construction Operations Demonstrated for Hole Cleaning Optimization and Automation. *Energies* **2022**, *15*, 5749. https://doi.org/10.3390/en15155749

Academic Editor: F. Pacheco Torgal

Received: 22 June 2022
Accepted: 5 August 2022
Published: 8 August 2022

1. Introduction

Well construction is the process of drilling and completing wells to extract subsurface fluids and gases such as water, brine and hydrocarbons, access geothermal energy, and store waste underground. Each well construction task has multiple co-occurring sub-processes and various systems mutually interacting, making it highly complex. Furthermore, drilling deep in the subsurface into variable geological environments adds an inherent irreducible level of complexity to the process. The highly involved and poorly predictable nature of well construction can result in operational inefficiencies or safety and environmental issues, potentially leading to non-productive time (NPT) associated with events such as lost circulation, stuck drillstring and well control incidents. Therefore, careful surveillance of process and equipment data for performance tracking and building intelligent systems for decision-making and action planning are needed.

There are many algorithms and systems in the well construction domain that currently perform process monitoring. To assist with decision-making, approaches such as case-based reasoning [1,2], performance tracking using digital twins [3–5], knowledge graphs and decision-trees [6,7], and type curve matching [8,9] are being utilized. Remote process monitoring and multidisciplinary collaboration among domain experts have been enabled by the use of real-time (RT) data streams, advanced process and system models, and machine learning techniques [10–13]. Final decisions, however, are often still based on an engineer's understanding and interpretation of the data, key performance indicators, and model outputs. This human-centered decision-making not only introduces a degree of bias [14], but can also result in reactive, short-term decisions instead of proactive long-term action planning.

By contrast, intelligent decision-making and action planning systems have resulted in overall performance improvements in various other fields. For instance, areas such as microgrid energy management [15,16], economic dispatch and large-scale power dispatch problems [17,18], electric vehicle systems management [19,20], and smart grid management [21] utilize reinforcement learning (RL) methods successfully. More recently, very high complexity adversarial board games (e.g., chess and Go), and complex RT strategy games (e.g., Alphastar, Dota, and Pac Man) have been solved using a combination of deep neural networks and tree-based search techniques [22–27].

The goal of the research presented here is to examine the applicability of these novel planning techniques in the well construction domain, and to utilize them for the development of intelligent decision-making systems (or decision-engines) for improved safety and performance (Figure 1). The proposed methodology is demonstrated here by structuring a decision-engine specifically for hole cleaning operations. Hole cleaning is the process of removing solids (cuttings, cavings, or metal shavings) from the borehole by circulating a drilling fluid in and out of the well. Removal of solids is critical in ensuring different well construction operations (such as drilling, tripping, running casing and cementing) are performed safely and efficiently. Issues such as downhole tool damage, stuck pipe situations, damage to downhole formations, and difficulty running casing and cementing can result from poor hole cleaning. Annually, hole cleaning issues cause several hundred million dollars in NPT costs [28,29]. Therefore, building a decision-engine for hole cleaning advisory and eventual automation that permits superior proactive action planning and execution is of considerable practical importance and value.

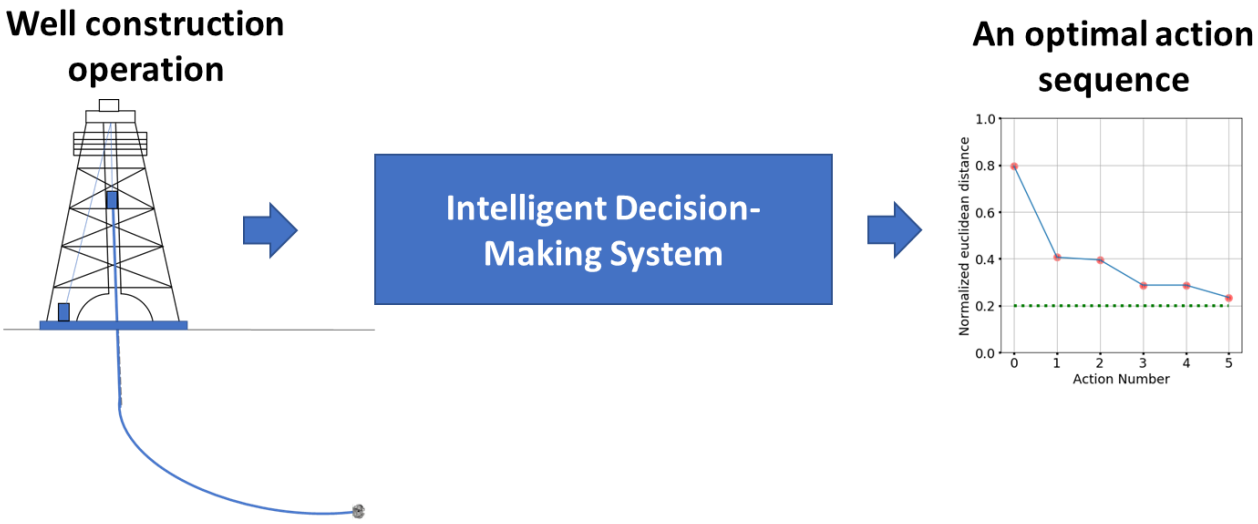


Figure 1. Schematic representation of the objective of this work: development of intelligent decision-making systems for well construction operations. The right figure shows progression of the normalized Euclidean distance of the system states (blue line) to an acceptable goal state (shown by the green dotted line, here taken to be a state at a normalized Euclidean distance of 0.2).

2. Setting Up Decision-Engines for Well Construction Operations

Planning is the process of generating an action sequence from an initial system state to a goal state to satisfy a high-level objective function. Setting up a planning system requires explicitly stating the problem objectives and constraints, defining the state- and action-space, and identifying goal states [30]. The various state-action transitions can either be fully specified in advance or be incrementally discovered (by utilizing models) as the planning proceeds. The solution of a planning problem is a policy or a strategy that suggests the sequence of actions for every successive decision-making step (called a decision epoch). The following sections provide an overview of different planning algorithms (particularly for well construction engineers unfamiliar with such algorithms, which have not yet found widespread adoption in their industry) and discuss their applicability to well construction operations.

2.1. Comparison of Planning Algorithms

Planning algorithms can be compared based on a multitude of factors, such as the search-space traversal methodology, requirement of a heuristic, ability to plan in a limited amount of time, memory requirement for storing intermediate results, computational complexity, etc. [30]. A straightforward approach to solving planning problems, however, is the exhaustive tree search, wherein all possible actions from all states are evaluated until either a goal state is reached, or the available evaluation time is exhausted. Although this method ensures finding an optimal plan, for large systems (with large state and action spaces), the associated time complexity renders this method impractical. Some other classifications of algorithms based on different search-space traversal methodologies are breadth-first search (BFS), depth-first search (DFS), and best-first search (BestFS).

BFS is a first-in, first-out (FIFO) algorithm that searches equally in all directions, i.e., all actions from a given state are evaluated first before the evaluation of the next state's actions can begin. The algorithm stops and outputs the shortest path as soon as a goal state is reached [30,31]. DFS is a more aggressive strategy, wherein, from any state, actions are chosen at random to traverse the system, until either a goal state or a dead-end state is reached. If a dead-end state is reached, the algorithm backtracks one step, then simulates any action not yet explored, i.e., this algorithm works in a last-in, first-out (LIFO) manner. The priority of such algorithms is to search deeper than to expand the search-space. A potential issue with this method is the search becoming stuck in a repetitive loop [31,32]. Neither the DFS nor the BFS utilizes an evaluation function or a heuristic to direct the search, i.e., costs associated with all actions taken from all the states are the same. Moreover, both methods require keeping track of the visited, unvisited, and dead-end states. Therefore, these algorithms run into issues when dealing with large state- or action-space systems due to high computation-time and memory requirements, with BFS more so than DFS due to its more exhaustive nature [30,33].

The BFS algorithm can be enhanced by using some form of evaluation functions or heuristics or both. Dijkstra's algorithm utilizes an evaluation function to quantify the path traversed to reach a given state after starting from the initial state. It, like the BFS, will find an optimal path (if one exists) between the initial and the goal state, but suffers from the same computational and memory issues associated with larger systems [34,35]. If a heuristic is defined to quantify the approximate cost of a state relative to its position from the goal state, this results in a greedy BestFS algorithm. The next action is chosen greedily based on this heuristic. Although greedy BestFS is usually faster than Dijkstra's algorithm, a drawback of this algorithm is the lack of exploration and backtracking, i.e., once an action is selected, it is not re-evaluated based on any new information [30,36]. This search strategy is, therefore, not guaranteed to find an optimal path. Dijkstra's algorithm supplemented by the greedy BestFS heuristic results in the A* search algorithm [30,37]. A* search is a BFS algorithm for which both an evaluation function and an admissible heuristic are defined. A* search will find an optimal path (if one exists) faster than BFS or the Dijkstra's algorithm, but it is memory-intensive since it requires storing all the visited states and keeping track

of all possible and tried actions from all possible states. Another issue with A* search is the necessity for a well-defined admissible heuristic [38].

Iterative deepening search (IDS) is a variant of the DFS, especially useful in systems with high branching factors. For IDS, the DFS algorithm is continually run with increasing depth bounds until a goal state or a dead-end state is reached. During each iteration, every state encountered in the path is expanded in a BFS manner. IDS is faster than the BFS and is guaranteed to find an optimal path if one exists [39]. The iterative deepening A* (IDA*) algorithm utilizes the search technique of the IDS in conjunction with a heuristic to quantify the cost of any state relative to the goal state. IDA* is guaranteed to find the optimal solution if the heuristic is admissible [30,31]. The main advantage of iterative search algorithms (IDA and IDA*) is that they do not require much memory, since only the current path is stored. Evolutionary algorithms such as the genetic algorithm evaluate and rank multiple action sequences simultaneously using an expert-designed fitness function. The quality of the population of these action sequences is successively improved using selection, crossover, and mutation operations, until some stopping criterion is met. The intrinsic randomness in crossover and mutation steps ensures a balance between exploration and exploitation of the search space. These algorithms are the most effective if the search space is sufficiently small, or if time is not a constraint for search. Additionally, the performance of these algorithms is highly dependent on the design of the fitness function, and the crossover and mutation strategies [40,41].

For systems with a vast state- and action-space, lack of an admissible heuristic or minimal dependency on some evaluation function, memory constraints, and limited computation time availability, simulation-based search (SBS) methods are the best suited. SBS, a decision-time planning algorithm, is a type of model-based RL method suited for online action planning, starting from a given state [41]. Here, a goal-directed agent interacts (virtually) with an uncertain environment (models representing the underlying processes) based on specific policies or action plans which evolve with time. These policies are designed to balance exploration and exploitation of the search space, and every state action transition has an associated reward. These algorithms build an 'asymmetric' search tree starting from an initial or root node using a sequential BestFS strategy, as shown in Figure 2. Multiple episodes of experience (starting from the root or the current state, s_t) are simulated until either the goal state or a fixed depth bound is reached. After every episode, one or more nodes (representing a state-action transition) are added to the tree, and the values of the already present nodes are updated. SBS algorithms do not require a heuristic (i.e., they are 'aheuristic'); however, a well-defined heuristic can considerably improve the convergence time. Another useful feature of these algorithms is their 'anytime' property, i.e., the algorithm can be stopped at any time to return the best plan thus far [42].

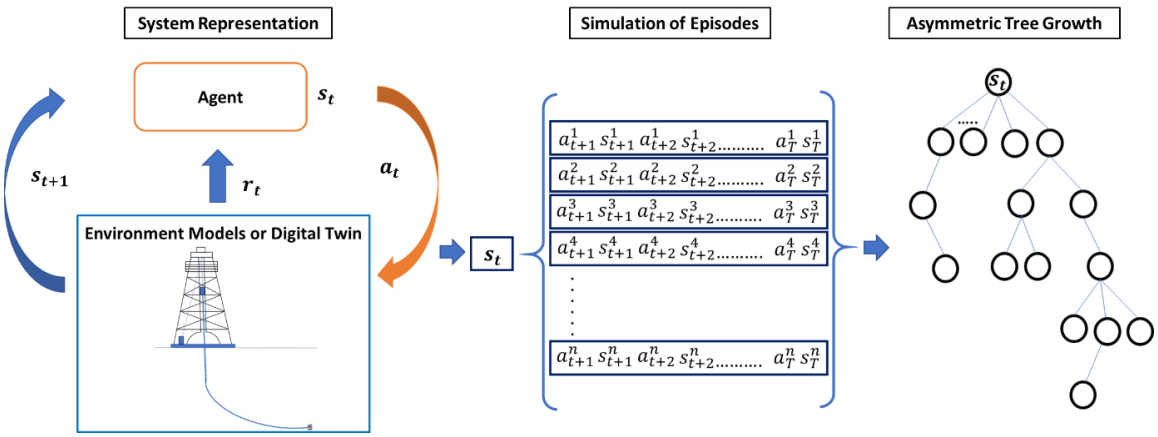


Figure 2. Asymmetric tree growth using simulation-based search algorithms. An action a_t by the agent in the environment (observed by the agent to be in a state s_t) results in an immediate reward r_t and a new observed state s_{t+1}.

Essentially, the anytime, asymmetric, and aheuristic nature of the SBS algorithms makes them a preferred candidate for large systems with memory and computational constraints, as is the case in well construction.

2.2. Utilizing SBS for Well Construction Action Planning

Well construction processes are non-holonomic since the wellbore condition at any time (state) is a function of the well's operational past, i.e., all previous wellbore conditions (past states) and previous operations (actions) [43]. The states themselves, depending on the process, can be represented by combinations of parameters such as equivalent circulation density (ECD), cuttings bed height, cuttings concentration in the flow, friction factor, and drilling dysfunction indicators. To adequately represent a process and distinguish different states, each of these parameters can take on many different values. These unique combinations of values for the many state parameters can result in a vast state-space. Similarly, various combinations of values for the many action control variables result in a vast action-space. Some action variables for drilling operations include drillstring rotation speed (RPM), weight on bit (WOB), flowrate, tripping speeds, mud properties (density and rheology), etc. [44]. Furthermore, the online or near real-time planning requirements impose constraints on computation time.

Decision-time algorithms that can plan in limited time by considering the search-space in the vicinity of the current state are therefore needed to address these requirements. SBS algorithms, such as flat Monte Carlo (MC) and Monte Carlo tree search (MCTS), have been successfully used for planning in systems with large state- and action-space (e.g., game AI engines for chess and Go, as discussed). The game AI engines utilize the MCTS in conjunction with deep neural networks (for policy and value evaluations) to identify, shortlist and sequentially simulate legal actions from a given position (state) for both players. Multiple episodes of such self-play are used to improve the policy incrementally and finally select the most promising line of play (an optimal action sequence). Likewise, the utilization of such SBS techniques for action planning during the various well construction operations can result in improved efficiency and operational performance.

2.2.1. Monte Carlo Tree Search (MCTS) Algorithm

The MCTS is an SBS method that combines MC search with an incremental tree structure. Successive MC simulations (by using a forward model of the process) are utilized for iteratively expanding the search tree and evaluating the various nodes in the tree structure. A node represents a state and contains information about its parent node, possible next actions, the number of current implementations of each action, and the average value associated with each action implementation thus far. Instead of building the entire tree, a few promising lines of play are developed further, resulting in asymmetric tree growth. The selective growth of the search tree is brought about by using two distinct action selection policies: the tree policy and the rollout policy [45,46].

Tree policies can be either greedy (i.e., focusing on exploiting what is already known), or try to balance exploring new or potentially promising paths of the search space with greedy exploitation. This balance is commonly referred to as the exploration-exploitation trade-off. One such policy can be devised by treating every state within the tree as a multi-armed bandit (MAB) problem where the Upper confidence bound (UCB1) algorithm is utilized for action selection within the tree structure [47]. The resulting algorithm is referred to as Upper confidence bound for trees (UCT). The UCT guarantees convergence to an optimal policy (given enough time) since the exploration factor (C_{exp}) not only ensures exploration of unvisited parts of the search space, but the exploration term ($\sqrt{\frac{2 \cdot \ln N(s)}{N(s,a)}}$) also gets less exploratory with increasing number of visits [48]. Equation (1) represents the basic UCT policy, where $N(s)$ is the total number of visits to the state s, $N(s,a)$ represents the number of times action a has been taken from state s. $Q(s,a)$ is the exploitation term representing the average value associated with implementing action a from state s. $Q_{\mathrm{UCT}}(s,a)$ is the upper confidence bound or the urgency term, and the next action within

the search tree is selected based on maximizing this term over the action space A_s from the state s (Equation (2)). The rollout policy, by default, is a uniform-random policy.

$$Q_{\mathrm{UCT}}(s,a) = Q(s,a) + C_{exp}.\sqrt{\frac{2\cdot \ln N(s)}{N(s,a)}} \quad (1)$$

$$a_{n+1} = \max_{a_i \in A_s}(Q_{\mathrm{UCT}}(s,a_i)) \quad (2)$$

Primarily, MC control (a model-free RL technique) is applied to simulated episodes of experiences (model-based RL) from a root node, to grow the search tree iteratively and improve the action plan by backpropagating feedback. A single MCTS iteration consists of the following four phases, as detailed in Figure 3 [47]:

- Selection from nodes already in the search tree using the tree policy;
- Expansion of the tree at the leaf node by adding one (or more) node(s);
- Simulation or rollout of actions, using the rollout policy, until the terminal condition is met (either reaching a terminal state or the end of the planning time horizon, T);
- Backpropagation or backing the rewards up the expanded tree to update the values of different state-action pairs ($Q(s, a)$) encountered during the episode.

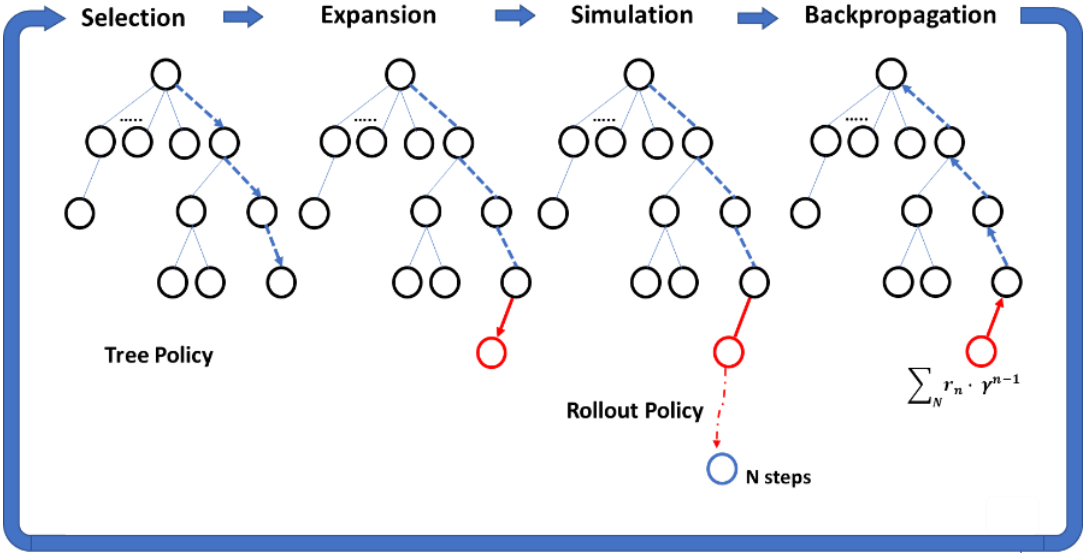


Figure 3. Schematic overview of Monte Carlo tree search algorithm phases (modified from [44]).

As the number of simulations increases over time, the tree expands, and any inherent bias is removed. Every MCTS episode commences from the root node (from which an action plan or action sequence needs to be determined), and traverses the tree based on the most recently acquired knowledge (which is incorporated in the Q values). Thus, MCTS is fundamentally a generalized policy iteration (GPI) algorithm, where the policy or action plan is iteratively evaluated and improved [41,49]. Even with naïve or vanilla policies (such as the standard UCT and uniform random), the MCTS works well and starts to move the results towards optimality.

Although the vanilla MCTS blends the generality of random sampling with the exactness of tree search, its convergence rate can be relatively low. Therefore, in practice, the two MCTS policies have been enhanced by incorporating prior-knowledge or handcrafted strategies. This has resulted in different methods, such as progressive widening, progressive bias, using prior-knowledge, and Rapid Action Value Estimation (RAVE) [47,50–52]. All these strategies require evaluation of a heuristic function, which can either be learned (using deep neural networks) or designed using domain-knowledge or can be a combination of both. The heuristic function devised for most other applications depends on factors such as the proximity of the state to the goal state, values associated with patterns, previous

action values, how dangerous or safe is the state as compared to the adversary's positions, etc. [27,50,51,53].

2.2.2. Structuring Well Construction Operations as Sequential Decision-Making Systems

The goal of well construction is to drill and complete a well safely, quickly, efficiently, and economically in line with a drilling program. As previously stated, accomplishing these objectives requires efficient planning and informed decision-making at each step of every well construction process. Additionally, due to the non-holonomic property of well construction operations, any action implemented at the current time will not only affect the immediate system state but also influence the long-term evolution of the system. This necessitates the development of intelligent decision-making systems, the foundation of which is a Markov Decision Process (MDP) framework. An MDP is composed of the tuple $(\{S, A, P, R\})$ and a policy (π) that address the following elements of finite-horizon long-term sequential decision-making [30,54]:

- Identification of appropriate parameters to quantify the state (s_t) of the system, and defining the desired or goal states (s_{goal}) based on the operation's objective;
- Identification of relevant control variables or actions (a_t) that can affect the system state;
- Building the state-space (S) and the action-space (A) such that $s_t \in S$ and $a_t \in A$ for all s_t and a_t;
- Incorporating problem-specific heuristics ($\pi_{heuristic}$) to identify legitimate and promising actions from every state;
- Building models (digital twins) of the underlying processes to simulate the state transitions ($P^a_{ss'}$) brought about by different actions;
- Defining a method to quantify the various state action transitions, for instance, by using reward functions (R) to calculate long-term value functions (Q);
- Selection of a suitable action-planning technique to formulate a plan (π—the suggested sequence of actions for successive decision epochs).

For a process to be Markovian, it must satisfy the Markovian property, i.e., any transition from a given state depends only on the current state and the immediate action. In other words, the current state completely summarizes the system's operational past. As previously discussed, this is true of well construction operations; therefore, the Markovian property assumption is valid. The MCTS algorithm is particularly well-suited for well construction action planning because of its ability to handle vast search-spaces efficiently. The UCT policy assists with asymmetric tree growth along promising paths while balancing exploration and exploitation of the search space. Although MCTS is inherently aheuristic, it still permits the use of domain-knowledge-derived heuristics for speeding up the search. Moreover, the GPI property of MCTS ensures that over time better plans (action sequences) are found. Figure 4 shows the proposed structure for well construction decision-making systems.

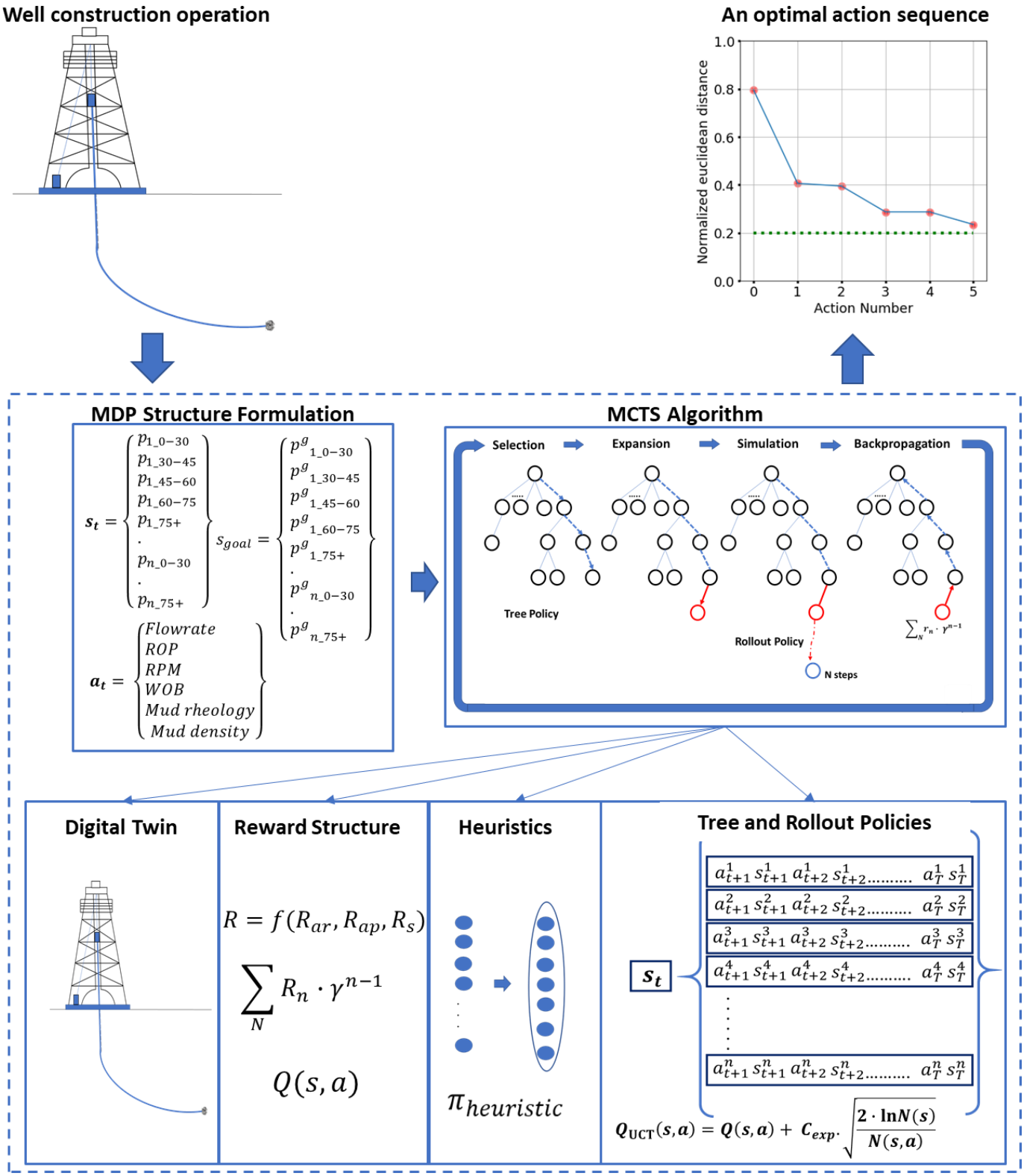


Figure 4. Structure of the proposed decision-engine. Upper right figure shows progression of the normalized Euclidean distance of the system states (blue line) to an acceptable goal state (shown by the green dotted line, here taken to be a state at a normalized Euclidean distance of 0.2).

3. Design of a Planning System for Hole Cleaning Action Planning using MCTS

The objective of hole cleaning operations is to manage:

- Height of the cuttings bed (that settles on the low side of the borehole) at low enough values to prevent issues during any subsequent stage of well construction (Figure 5a);
- Downhole ECD values to remain within a given drilling margin (Figure 5b).

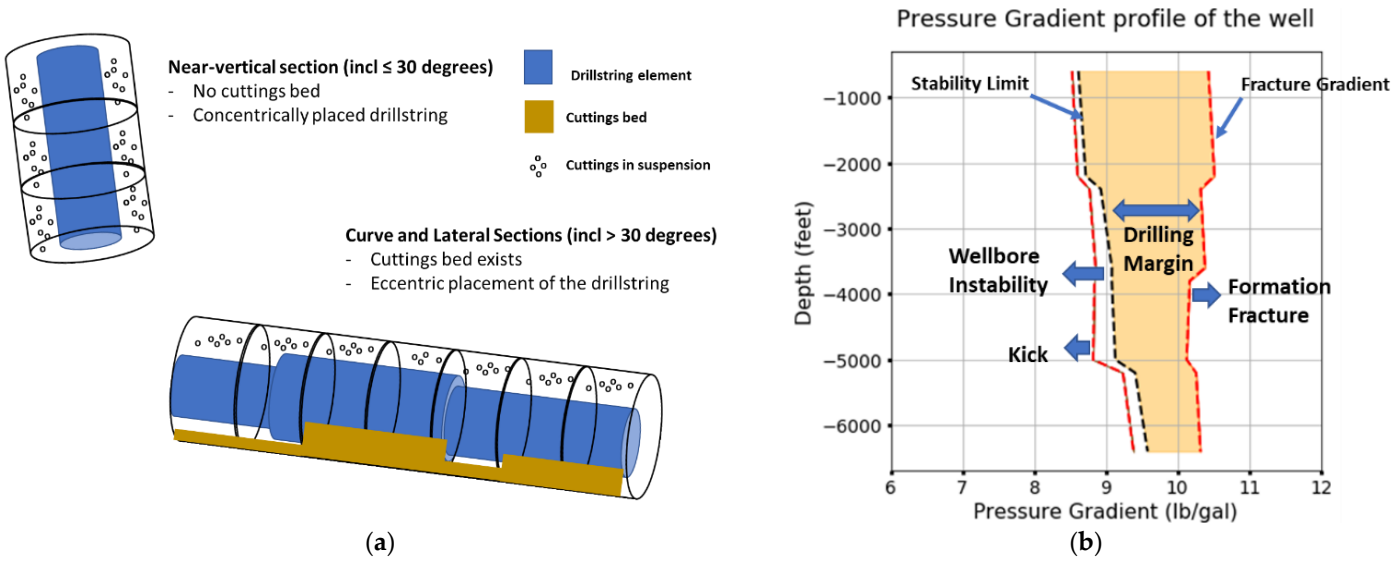


Figure 5. (**a**) Cuttings bed height in different inclination intervals for the well (**b**) Drilling margin.

ECD at any depth is the gradient of the total pressure drop at that depth, and is calculated by Equation (3). The pressure drop is the sum of the hydrostatic pressure exerted by the drilling mud (which is a function of the true vertical depth (TVD) of the well), and the total circulating frictional pressure drop in the annulus between the drillstring and the wellbore (which is a function of the measured depth (MD) along the annular space) [44]:

$$\mathrm{ECD} = \frac{\mathrm{P}_{\mathrm{hydrostatic\_D_{TVD}}} + \mathrm{P}_{\mathrm{frictional\_pressure\_loss\_D_{MD}}}}{\mathrm{D_{TVD}}.\ \mathrm{g}} \quad (3)$$

The stability limit (SL) and the fracture gradient (FG) form the lower and the upper boundaries of the drilling margin. Exceeding the FG can result in fracturing the formation, which can lead to loss of the drilling mud into the formation, referred to as lost circulation events. SL is the higher value between either the pore pressure (PP) or the mud pressure required for wellbore stability. PP is the pressure exerted by the fluids (hydrocarbons or brine) in the pore spaces of formation rocks. Wellbore instability can occur when the ECD or hydrostatic mud pressure falls below the SL, and a kick, i.e., an unwanted influx of formation fluids/gases into the borehole [55] can happen if the ECD or hydrostatic mud pressure falls below the PP.

3.1. Setting Up the MDP

A directional well can be analyzed by dividing it into three distinct sections based on the inclination angles: near-vertical section, build or curve section, and horizontal section. The segment of the well with inclination angles between 0 and 30 degrees is the near-vertical section, while regions of the well with inclination angles greater than 60 degrees constitute the near-lateral horizontal section. The intermediate inclination angle segments comprise the build or the curve section of a well. This method for discretizing the state-space is proposed since the state variables' response to different actions depends to a high degree on the inclination of the well segment. Likewise, for the hole cleaning system, the cuttings transport mechanisms, and therefore the associated hole cleaning requirements,

are different for the three sections. The cuttings are suspended and carried upwards by overcoming the particle slip velocity in the near-vertical section. For the horizontal section, above the angle of repose for the cuttings, a stable cuttings bed forms on the low-side of the borehole. The main requirement is to move this bed up and out of the hole continuously. In the curve section, an unstable cuttings bed forms below the angle of repose, which can avalanche back down the annulus and pack-off around the bottom hole assembly (BHA), causing stuck pipe incidents. The hole cleaning design for this section, therefore, revolves around preventing this cuttings avalanche [56,57].

3.1.1. State-Space and the Goal State

The following parameters are required for quantifying the borehole condition from a hole cleaning standpoint [58,59]:

- Height of the cuttings bed in the curve and the lateral sections of the well;
- ECD along the length of the well.

The state of the system, at any decision epoch, is represented by functional values of these parameters over the inclination intervals $\{[0,\ 30),\ [30,\ 45),\ [45,\ 60),\ [60,\ 75),\ [75+)\}$, as shown in Figure 6, and represented by Equation (4). The functional values for the cuttings bed height ($H_{inc.}$) and the ECD ($ECD_{inc.}$) parameters are whole numbers in ranges $[0,\ 4]$ and $[-3,\ 3]$, respectively. Conversion to functional values helps normalize the absolute values to specific operational thresholds, and assists in reward shaping, which is discussed in later sections. Since the cuttings bed cannot exist in the near-vertical section (inclination interval $[0,\ 30)$), H_{0-30} is not included in the definition of the state.

$$s = \begin{Bmatrix} H_{30-45} \\ H_{45-60} \\ H_{60-75} \\ H_{75+} \\ ECD_{0-30} \\ ECD_{30-45} \\ ECD_{45-60} \\ ECD_{60-75} \\ ECD_{75+} \end{Bmatrix} \tag{4}$$

Defining goal states (or desired states) is necessary to direct a decision-making agent's search of its state and action-space. For the hole cleaning system, the functional values defined for all state components are such that 0 represents the goal state for each. The method for converting the absolute parameter values to their functional values is summarized in Figure 7.

Equation (5) represents the goal state for this system.

$$s_{goal} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{5}$$

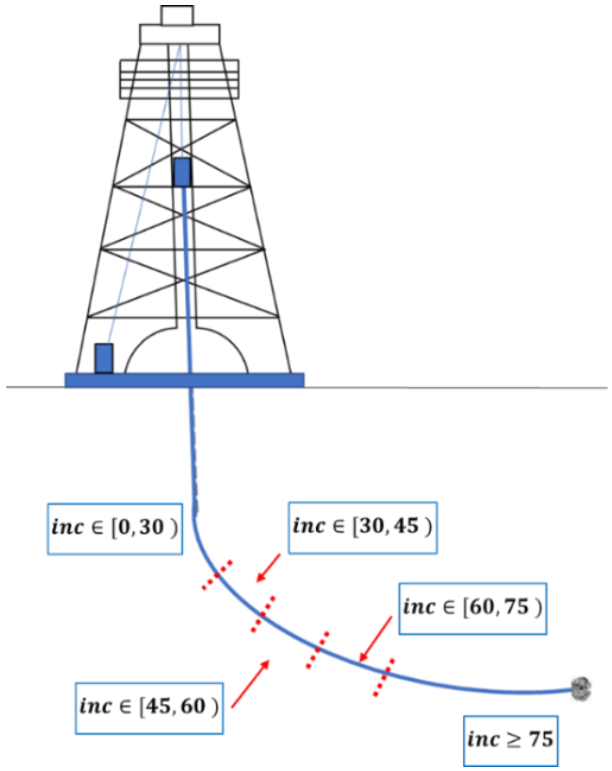


Figure 6. Strategy for defining state vector components based on wellbore inclination angles.

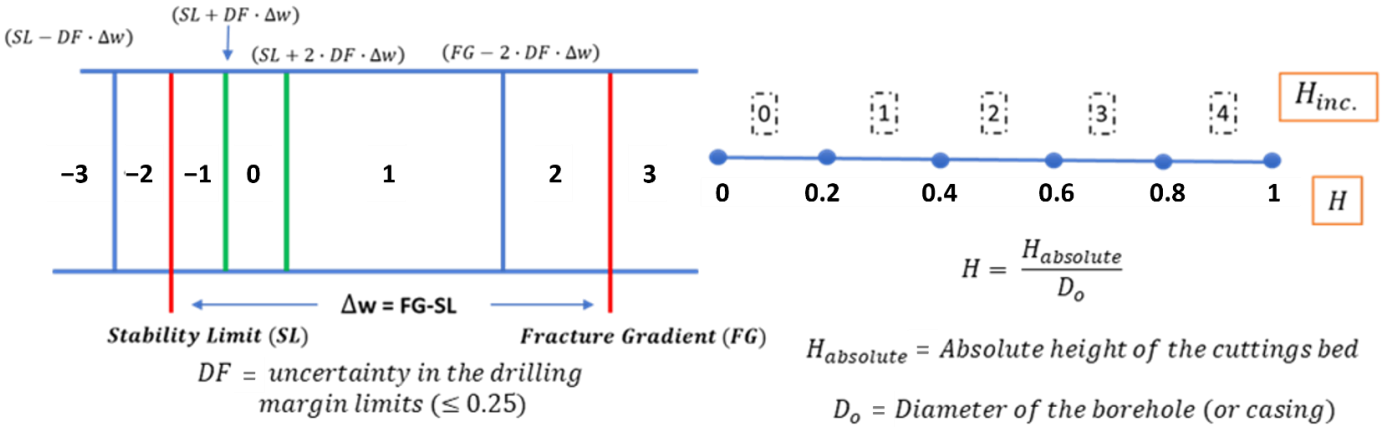


Figure 7. Functional value assignments for the state parameters.

3.1.2. Action-Space

Hole cleaning and ECD management are influenced, to varying degrees, by multiple factors, some of which can be actively controlled in real-time in the field. Factors such as drilling mud properties (particularly density and rheology), drilling parameters (such as drillstring RPM and flowrate), and rate of cuttings generation (which is a function of ROP) significantly influence hole cleaning performance, and can be actively controlled in the field [60,61]. For this research, we assume Bingham Plastic rheological behavior for the drilling mud, in which case its rheology can be characterized by a plastic viscosity (PV) and yield point (YP) [44]. The action set, as shown in Equation (6), is a combination of discrete values of these control variables. Each control variable can take on a finite number of values between some minimum and maximum thresholds that are determined by safety constraints, operational economics, and equipment and process limitations. For instance, if

the flowrate has ten values in the [0, 1800] interval, its equally spaced values, in gallons per minute (GPM), are {0, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800}.

$$a_t = \left\{ \begin{array}{c} \text{Flowrate} \\ \text{ROP} \\ \text{RPM} \\ \text{Mud density} \\ \text{Mud PV} \\ \text{Mud YP} \end{array} \right\} \quad (6)$$

3.1.3. Digital Twin

A digital twin of the hole cleaning operations was built by integrating multiple data streams with analytical implementations of the cuttings transport, hydraulics, and rig-state detection models [62]. The cuttings transport and hydraulics models estimate the height of the cuttings bed and the ECD along the well, respectively. The rig-state detection engine outputs the current operational state (such as rotary or slide drilling, tripping in or out the borehole, etc.) [63]. The purpose of this twin is twofold: first, for performance tracking by simulating different hole cleaning actions, and second, as a forward simulation model for assisting with action planning. Figure 8 illustrates the application of the twin as a forward-simulation model, where an action a_t transitions the system state from s_t, to s_{t+1}, at decision epoch t. The digital twin was designed to plan either every 5-min interval into the future, or whenever there is a change in the well operations.

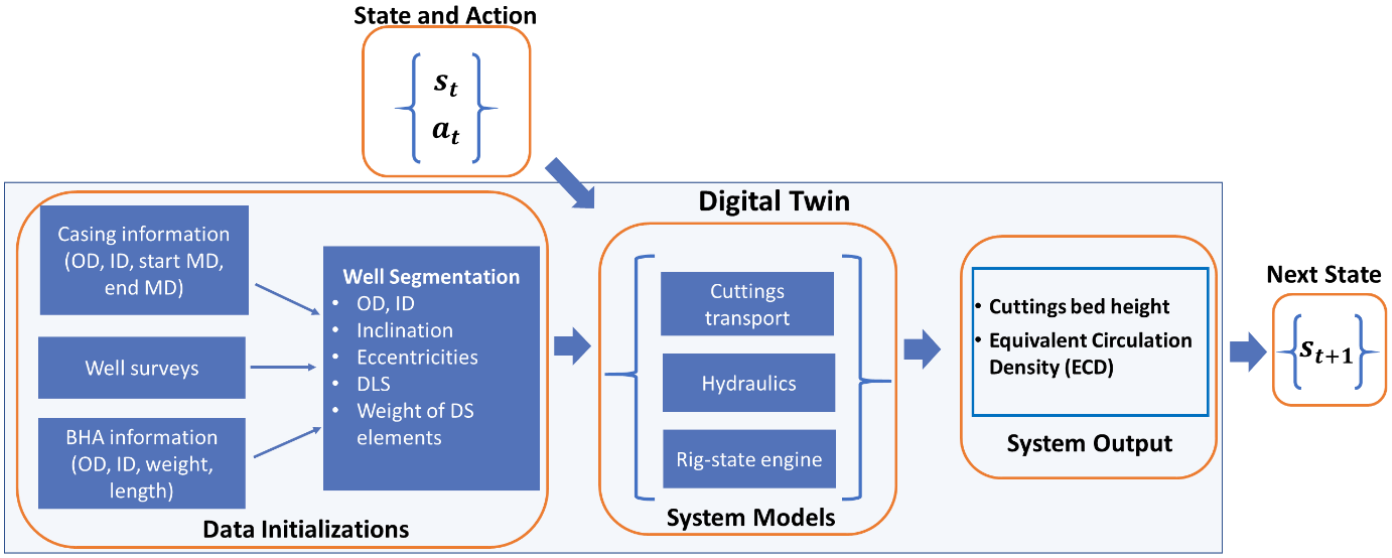


Figure 8. Application of the hole cleaning digital twin for forward-simulation.

3.1.4. Reward Function

A reward function that outputs a normalized feedback (in the [0, 1] interval) for every state-action transition is designed by considering the following:

- the reward set associated with state transition (R_s);
- the penalty set associated with action transition (R_{ap});
- the reward set associated with action values (R_{ar}).

The three reward sets are shown by Equations (7)–(9).

$$R_S = \{R_{H30-45}, R_{H45-60}, R_{H60-75}, R_{H75+}, R_{ECD0-30}, R_{ECD30-45}, R_{ECD45-60}, R_{ECD60-75,} R_{ECD75+}\} \quad (7)$$

$$R_{ap} = \left\{R_{flowrate}, R_{ROP}, R_{RPM}, R_{density}, R_{PV}, R_{YP}\right\} \quad (8)$$

$$R_{ar} = \left\{ R_{a\_flowrate}, R_{a\_ROP}, R_{a\_RPM}, R_{a\_density}, R_{a\_PV}, R_{a\_YP} \right\} \quad (9)$$

Each component of every reward set is assigned different relative weights, which are then combined to give a single number net reward (normalized in the $[0, 1]$ range) for each: R_{s_norm}, R_{ar_norm} and R_{ap_norm}. Finally, these normalized rewards are combined in a weighted fashion, as shown by Equation (10). R_{net} is the final reward output for every state-action transition. This definition of the reward function ensures immediate feedback after every action, instead of the agent having to wait for a few decision epochs (as is the case for sparse reward functions).

$$R_{net} = \frac{W_{s\_norm} R_{s\_norm} + W_{ap\_norm} R_{ap\_norm} + W_{ar\_norm} R_{ar\_norm}}{W_{s\_norm} + W_{ap\_norm} + W_{as\_norm}} \quad (10)$$

Another advantage of this reward structure is that assigning different weights to the various reward set components provides the ability to prioritize different objectives. Prioritizing objectives, for example, would be beneficial while drilling wells that have a high probability of well control issues. In such wells, maintaining ECD within the drilling margin is more important than completely removing the cuttings bed. Similarly, there can be wells where reaching the desired state fast has a higher priority than reducing the penalty associated with taking more aggressive actions. These objectives can be managed by assigning different relative weights to the individual state or action reward components.

3.2. MCTS Setup for Hole Cleaning

Action planning with MCTS requires solving a sub-MDP starting from the current system state (root-node) in a finite amount of time. The hole cleaning digital twin is utilized as the forward-simulation model to simulate the results of different actions on the wellbore condition, and the feedback obtained using the reward function is accumulated and backpropagated. Vanilla MCTS, however, has some limitations:

- The agent is rewarded only when a terminal or a goal state is reached, i.e., sparse rewarding;
- From any state within the tree, all actions in its action-space (A_s) are evaluated by the UCT policy regardless of their practicability for the operation;
- The rollout policy is random uniform, i.e., all actions (irrespective of their feasibility) have an equal probability of being selected.

These limitations result in the requirement to conduct more simulations to expand the search tree to the extent that it can be meaningfully used for trajectory evaluation, thereby slowing down the search. To address these issues this research makes the following changes to the vanilla MCTS:

- Definition of a non-sparse reward function, such that the reward for every state-action transition during the rollout step is utilized for updating all tree nodes' Q values. This is addressed by the reward function shaping strategy, which was discussed in the previous section;
- Using a heuristic function to improve the tree policy;
- Using a heuristic function to reduce the randomness in the rollout policy.

3.2.1. Heuristic Function Development

A domain-knowledge based heuristic $\pi_{heuristic}$ is carefully crafted by balancing the following criteria:

- Safety and performance metric (A_{sp}), to prevent well control issues (such as kicks and lost circulation events) from occurring, as well as to ensure efficient cuttings bed removal and ROP maximization for optimal drilling performance;
- Performance metric, to ensure efficient cuttings bed removal and ROP maximization for optimal drilling performance;
- Sequential metric (A_{sq}), to ensure smoother or sequential changes in values of action control variables;

- Feasibility constraints (A_{fe}), to suggest realistic changes in values of the action control variables;
- Proximity metric (A_{px}), to prioritize actions based on Euclidean distance to the goal state.

Balancing safety and performance is accomplished by incorporating guidelines for well control, hole cleaning, and drilling optimization. Figure 9 is a simplistic representation of some such rules as a decision-tree. The result is a set of feasible actions A_{sp} satisfying the safety and performance requirements.

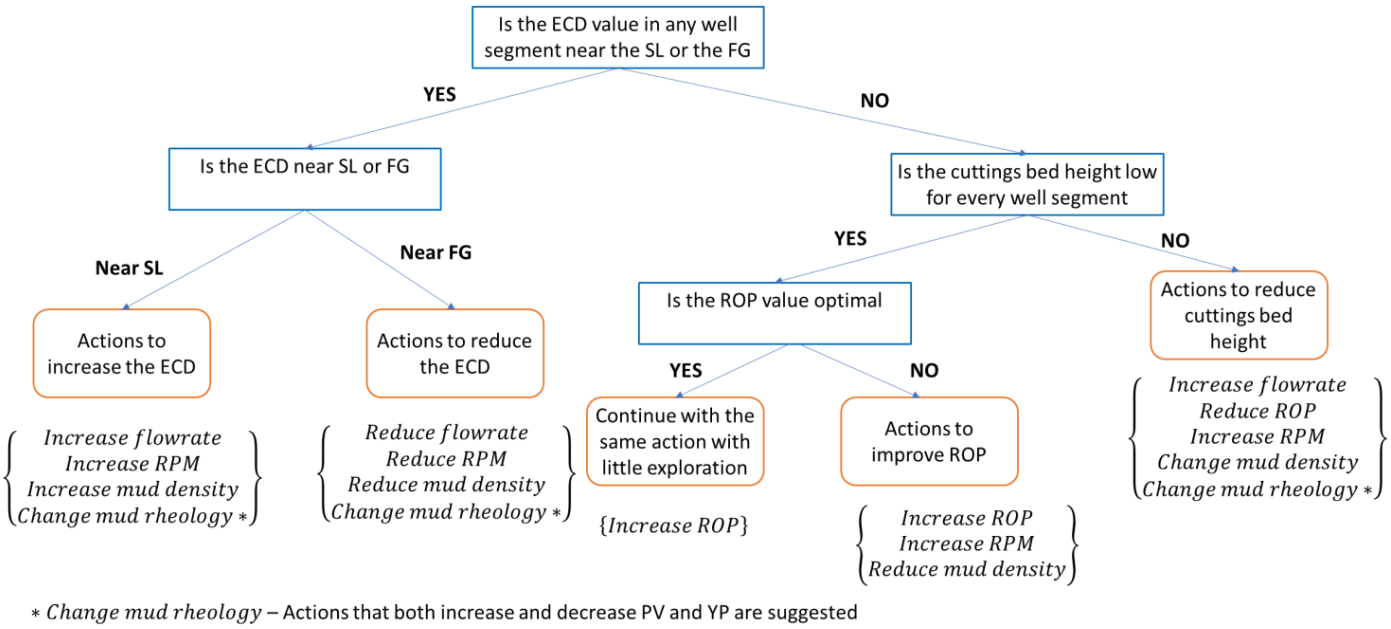


Figure 9. A simplistic representation of an action selection decision-tree for satisfying safety and performance metrics.

Similarly, to satisfy the sequential metric and the feasibility constraints, action sets A_{sq} and A_{fe} are evaluated, respectively. Figure 10 is a simple representation for estimating A_{sq} for a system with an action set containing three control variables (RPM, flowrate, and mud density). A_{sq} consists of actions in the vicinity of the most recent action. The reasoning behind such selection is to dissuade sudden changes in control variables, thereby constraining the rate of change of control variable values to safe limits. Changing mud properties (both density and rheology) is a time-consuming process; it cannot be performed in near real-time during conventional drilling or circulation operations (note that the effective density can be changed quickly in managed pressure drilling operations, where circulation takes place in a closed system where hydrostatic pressure can be quickly raised using a choke system). However, while planning for the circulation operation, adjusting mud properties is a crucial element of efficient hole cleaning. The purpose of defining A_{fe} is to incorporate such information.

To calculate the proximity metric, first, the normalized Euclidean distance for the current state (from s_{goal}), $d_{norm-euc}$, is computed using Equation (11). s_i is the ith element of the state vector s, and s_{i_max} is the maximum magnitude for the ith state vector component. The $d_{norm-euc}$ is then compared with the radius of a 'greed sphere'. This greed sphere is defined based on a normalized Euclidean distance of 0.5 from the goal state, and it is an indication of whether the states are far away from the goal state.

$$d_{norm-euc} = \frac{\sqrt{\sum_{i=i}^{N} s_i^2}}{\sqrt{\sum_{i=i}^{N} s_{i\_max}^2}} \tag{11}$$

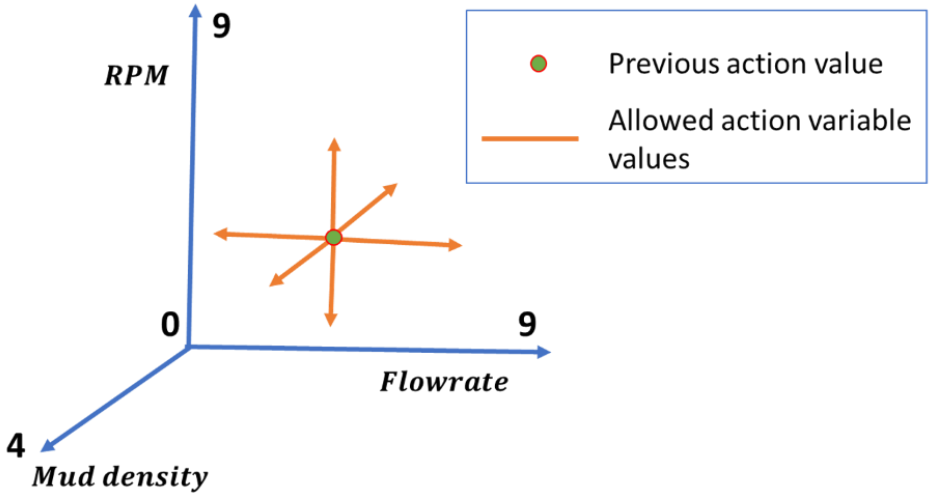


Figure 10. Method for estimating the action set associated with sequential metric. Numbers on axes are in control variable value units.

Figure 11 shows an example of this by considering a three-parameter system state (H_i, H_j and ECD_i). The near-goal states (s''_{goal}) represent the states for which the magnitudes of all state vector components are either 0 or 1.

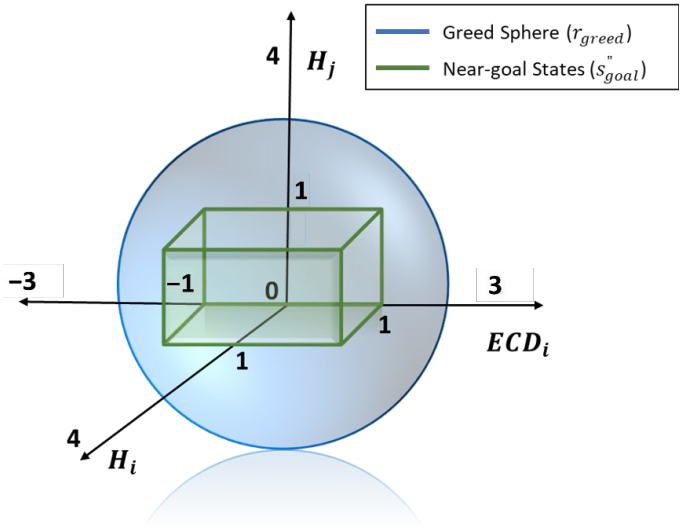


Figure 11. Method for estimating the action set associated with proximity metric.

The action selection strategy is given by Equation (12). In case the current state is at or near the goal state, the same action as the most recent action (a_{-1}) is selected. However, if the current state is further out than the greed sphere, more aggressive actions are included in the action set A_{px}. Aggressive actions represent greater magnitude changes in values of action control variables relative to the most recent action.

$$A_{px} = \begin{cases} a_{-1}\,, & d_{norm-euc} \leq s''_{goal} \\ a_{reg}\,, & s''_{goal} < d_{norm-euc} < r_{greed} \\ a_{agg}\,, & d_{norm-euc} \geq r_{greed} \end{cases} \quad (12)$$

Finally, the different action sets are combined, as shown in Equation (13), to output heuristic values for any action a in the action space A_s.

$$\pi_{heuristic\_a} = \begin{cases} 1\,, & if\ a \in A_{sp} \cap A_{sq} \cap A_{fe} \cap A_{px} \\ 0\,, & otherwise \end{cases} \quad (13)$$

Thus, $\pi_{heuristic}$ assigns a probability of either 0 or 1 to all the available actions in the action-space for a given state.

3.2.2. MCTS Structure

For finite-horizon action planning (T decision epochs into the future) in a limited amount of time t_{max}, the system starting in state s_0 proceeds according to the algorithm shown in Figure 12a. The root node $node_0$ corresponds to the starting state s_0, $node_{leaf}$ is the leaf node reached at the end of the selection phase in a given episode, and $node_{exp}$ is the randomly expanded node from the leaf node. The rollout phase then proceeds to plan until T epochs are reached (with root node as epoch 0), where $node_T$ is the final state (may or may not be the goal state), and G_T is the net discounted return. Equation (14) shows the calculation for G_T, where $\gamma(\leq 1)$ is the discount factor, r_k is the reward associated with the kth state-action transition, and $|exp|$ is the level for $node_{exp}$ in the tree. Finally, the backup function updates the Q value associated with all nodes from the $node_{exp}$ until $node_0$ by averaging this return value.

$$G_T = \sum_{k=|exp|}^{T} r_k.\gamma^{k-1} \tag{14}$$

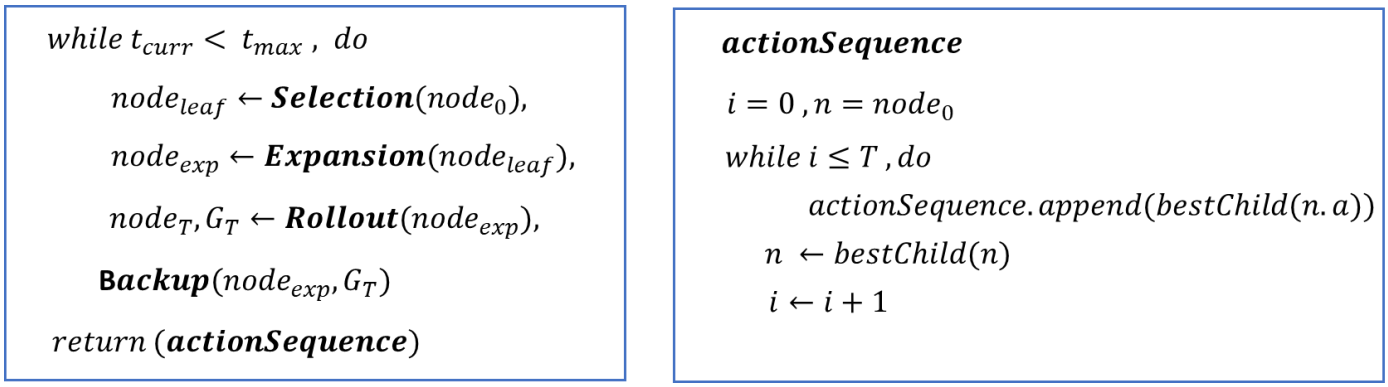


Figure 12. (**a**) MCTS algorithm (**b**) Action sequence selection method.

As previously discussed, MCTS builds an asymmetric search tree in the allocated time (t_{max}), after which an action sequence is given as output, based on the method shown in Figure 12b. The 'best child' from any node is its child-node corresponding to the action with the highest average Q value. Each node in the search tree contains the following information and can be represented by Equation (15):

- The current system state (s_t);
- The most recent action (a_{t-1});
- The action space of the node (A_{s_t});
- The total number of visits to the node so far (N_{s_t});
- All implemented actions and the resulting transitions, i.e., all child nodes ($\{a_i$:$node_i\}$);
- The total value associated with all iterations passing through the current state ($\sum_{children\ i} Q(s_t, a_i)$);
- The parent node ($node_{t-1}$);

$$node_t = \begin{pmatrix} s_t \\ a_{t-1} \\ A_{s_t} \\ N_{s_t} \\ Children: \ \{a_i : node_i\} \\ \sum_{children\ i} Q(s_t, a_i) \\ node_{t-1} \end{pmatrix} \quad (15)$$

The average value $(Q(s_t, a_t))$ associated with a state-action transition is calculated as the mean state-action value over all of its subsequent state's (s_{t+1}) children (a_j).

$$Q(s_t, a_t) = \frac{\sum_{children\ j} Q(s_{t+1}, a_j)}{N(s_t, a_t)} \quad (16)$$

The tree policy is modified by including the $\pi_{heuristic}$ in the exploration term of the UCT formula [23], as shown by:

$$\pi_{tree} = argmax_{a \in A_s} \left[Q(s_t, a) + \pi_{heuristic\_a}.C_{exp}.\sqrt{\frac{2 \cdot \ln N_{s_t}}{N_{(s_t,a)}}} \right] \quad (17)$$

Since the value of the exploitation term $(Q(s_t, a))$ is in the [0, 1] range, C_{exp} is key in determining the number of simulations that would be required to make the values of the exploration $(C_{exp}.\sqrt{\frac{2 \cdot \ln N_{s_t}}{N_{(s_t,a)}}})$ and the exploitation terms comparable. Figure 13 shows the plots for exploration terms calculated for different C_{exp} values as a function of the number of child node visits, given a total of 100 visits to the parent node (N_{s_t}). For the exploitation-heavy cases with C_{exp} values 0.25 and 0.5, the values of the two terms become comparable almost immediately, i.e., there is minimal exploration. On the other hand, for the exploration-heavy cases with C_{exp} values of 2 and 4, it takes a significant number of simulations to start exploiting its knowledge of the system. For the system developed in this research, a C_{exp} value of 1 is thus selected.

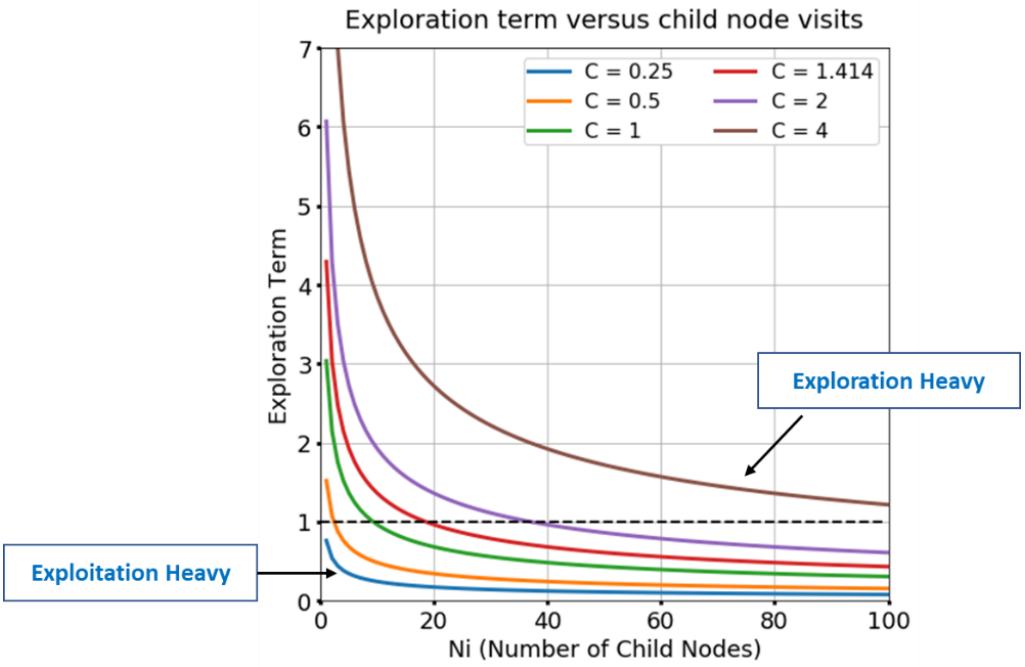


Figure 13. Calculation of exploration terms for different C_{exp} values as a function of the number of child node visits, given a total of 100 visits to the parent node.

During the rollout phase from any node, a uniform random selection policy is followed on a reduced action space (which is constructed based on Equation (18)).

$$A_{rollout} = \bigcup_{a \in A_s} \pi_{heuristic\_a}.a \tag{18}$$

4. Application of the System

The developed action planning system for hole cleaning was evaluated by performing post-mortem analysis on real-world oil well cases that exhibited hole cleaning issues. The analyses were performed by suggesting action sequences from some critical points during well construction. The final states associated with these plans were then compared with the actual well performance. A detailed investigation for one such well that exhibited issues due to insufficient hole cleaning during multiple well construction processes is presented here.

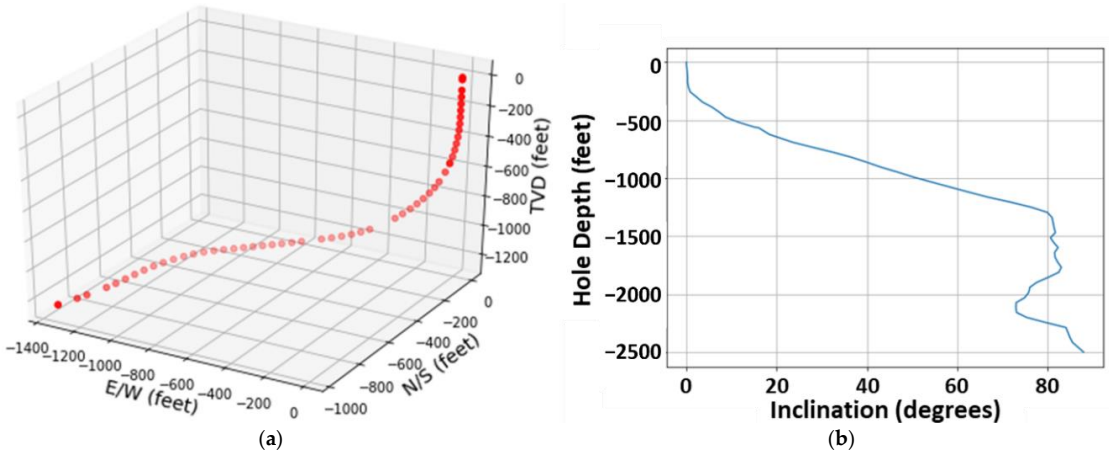
The dataset for the well included information such as:

- well trajectory, represented by directional surveys (inclination and azimuth versus the hole depth);
- well profile, represented by the BHA, casing and bit details;
- one-second surface sensor data, for directly and indirectly measuring the drilling parameters;
- mud-checks, to determine mud density and rheology among other drilling mud properties.

A hole cleaning digital twin, as discussed in Section 3.1.3 and detailed in Figure 8, was developed by integrating the various models with the relevant data sources.

4.1. Well Profile

The well had a short vertical section and a shallow kick-off point (where the well starts building inclination from the vertical) around 300 feet MD. After kick-off, the inclination angle was continuously built until the well became horizontal at about 1250 feet MD. After this, the well stayed near-horizontal until it reached its total depth (TD) of 2500 feet MD. The well profile is shown in Figure 14. This well was completed in two 'BHA runs' where each run comprised of drilling to a predetermined hole depth and subsequently, casing and cementing the hole. After the first BHA run, a surface casing of internal diameter 13.375-inch was set at a depth of 623 feet MD. Following this, in the second BHA run, a 12.25-inch hole section was drilled to well TD. Upon reaching TD, a 50-min on-bottom circulation cycle (at a flowrate of around 950 GPM and 60 RPM) was performed for hole cleaning purposes. The drillstring was then tripped out of the hole with intermittent back-reaming (at 910 GPM and 60 RPM), and finally, a 9.625-inch casing was run to TD and cemented. Poor hole cleaning negatively affected the last trip out of the hole with the drilling BHA, the run into the hole with the casing, and, ultimately, the casing cementing operation.



(**a**) (**b**)

Figure 14. (**a**) Well trajectory (**b**) Well inclination profile (negative sign indicates downward depth into the sub-surface).

Calculations show that for running a 9.625-inch casing in a 12.25-inch borehole, the maximum theoretical cuttings bed height (on pulling the drillstring out of hole) should not exceed approximately 5-inches (Figure 15). This 5-inch bed height corresponds to 45.1 in^2 of cuttings in the cross-section.

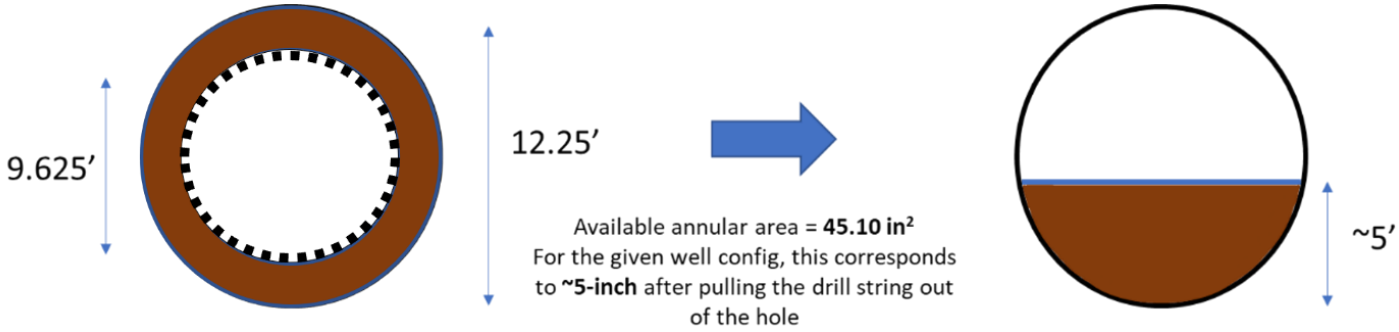


Figure 15. Theoretical maximum allowed height of the cuttings bed for the given well profile.

This cross-section of cuttings was then translated to an equivalent bed height by assuming the drillstring to be in the hole with the drill bit at TD. For the given well profile and trajectory (based on changes in outer diameters and eccentric placements of different drillstring components), this limit is represented by the red line in Figure 16a. The red shaded region corresponds to unsafe levels of cuttings bed height, while the green zone represents the goal state. Similarly, Figure 16b depicts the ECD profile (drilling window) for the well, bounded by SL and FG, with ten percent uncertainty in their values. As in Figure 16a, the green shaded zone corresponds to the goal state, while the red zones represent regions with the potential for well control issues. The zones in orange represent safe but suboptimal states.

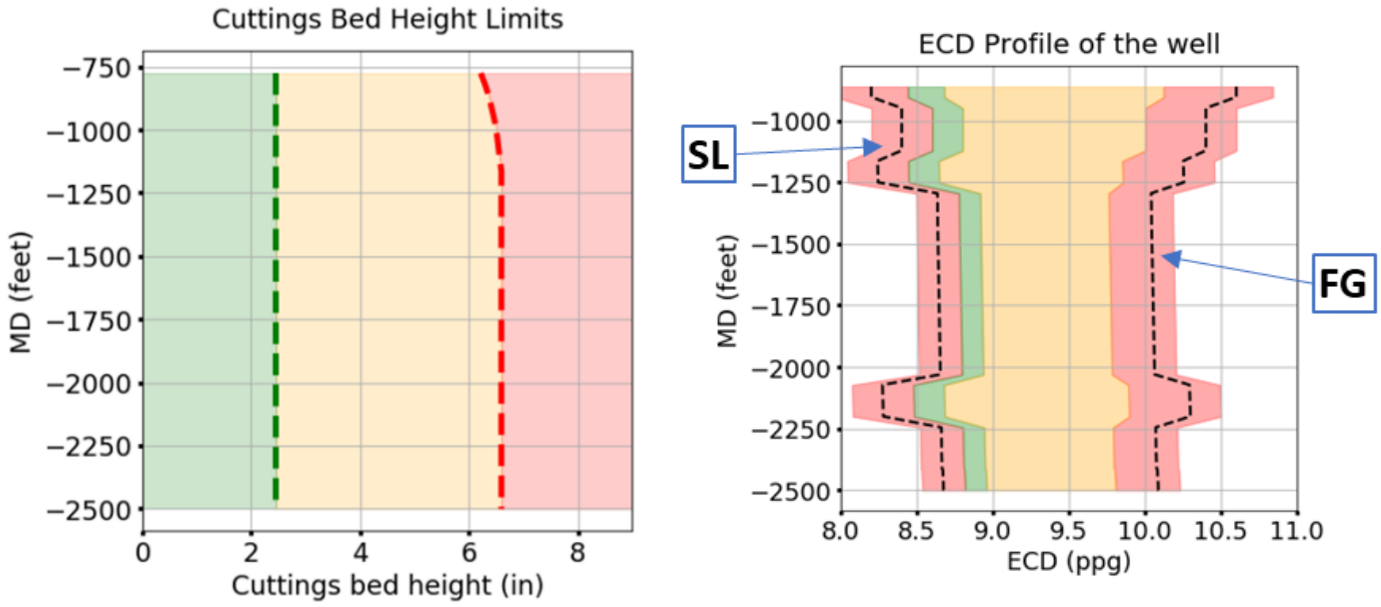


Figure 16. (**a**) Equivalent bed height limits for the given well profile (**b**) ECD limits for the well considering a ten percent uncertainty in the SL and FG values (negative sign indicates downward depth into the sub-surface).

Table 1 shows the SL and FG values to define drilling margins for different sections of the well. For the near-vertical section, the SL and FG were assigned non-limiting values of 6 ppg and 18 ppg, respectively, because this interval was entirely cased while drilling the 12.25-inch section during the second BHA run.

Table 1. The SL and FG values for the different inclination intervals.

Inclination Interval	Stability Limit (ppg)	Fracture Gradient (ppg)
[0, 30)—in casing	6	18
[30, 45)	8.2	10.6
[45, 60)	8.4	10.4
[60, 75)	8.2	10.2
[75+)	8.6	10.0

4.2. Performance Tracking and Action Planning

The system was set up in the MDP framework by using the procedure discussed in the previous sections. Table 2 shows the number and ranges of values associated with the different control variables utilized for defining the action-space.

Table 2. Number and ranges of values for the different control variables to define the action-space.

Control Variable	Number of Discrete Values	Range of Values
Flowrate (GPM)	10	[0, 1800]
Drilling ROP (ft/h)	10	[0, 900]
Drillstring RPM (rev/min)	10	[0, 180]
Mud Density (ppg)	5	[8.5, 9.7]
Mud Plastic Viscosity (cP)	5	[7, 42]
Mud Yield Point (lbf/100 ft^2)	5	[7, 42]

The weights assigned to the different reward function components are shown in Table 3. Equal weights for all the state transitions components (W_s) assumes equal relative importance of the different state components. Similarly, the relative penalties associated with changing the different action components (W_{ap}) are also assumed to be the same. As discussed in the previous section, these weights can be tuned by the rigsite engineer to prioritize different objectives. However, for the following example, the weights for combining the normalized reward components (weights associated with R_{s_norm}, R_{ap_norm} and R_{ar_norm}) are assigned such that being in or near the goal states is prioritized over the penalty associated with changes in action variables, or the rewards due to an increased ROP. Similarly, while drilling, the weight of the reward associated with a higher ROP is more than the weight of the penalty associated with the changes in action variables. The weight of the normalized action reward component (W_{ar_norm}) has two values depending on the operation being tracked. A weight of zero is assigned to circulation operations because no new hole is being drilled (i.e., ROP is zero).

Table 3. Weight assignments for different reward function components.

$W_s = [1, 1, 1, 1, 1, 1, 1, 1, 1]$	
$W_{ap} = [1, 1, 1, 1, 1, 1]$	
$W_{ar} = [0, 1, 0, 0, 0, 0]$	
$W_{s\_norm} = 0.50$	
$W_{ap\_norm} = 0.20$	
Drilling	Circulation
$W_{ar\_norm} = 0.30$	$W_{ar\_norm} = 0.00$

4.2.1. Performance Tracking

State transitions were monitored, and associated rewards were calculated to track the performance of the system. The state of the system at the end of the drilling operations (during second BHA run) is represented by Equation (19), and can be visualized by Figure 17. The mud properties for drilling the last section of the well were: mud density of 9.06 ppg, PV of 11 cP, and YP of 36.5 lbf/100 ft^2.

$$s_{TD} = \begin{Bmatrix} H_{30-45} \\ H_{45-60} \\ H_{60-75} \\ H_{75+} \\ ECD_{0-30} \\ ECD_{30-45} \\ ECD_{45-60} \\ ECD_{60-75} \\ ECD_{75+} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 3 \\ 4 \\ 4 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix} \quad (19)$$

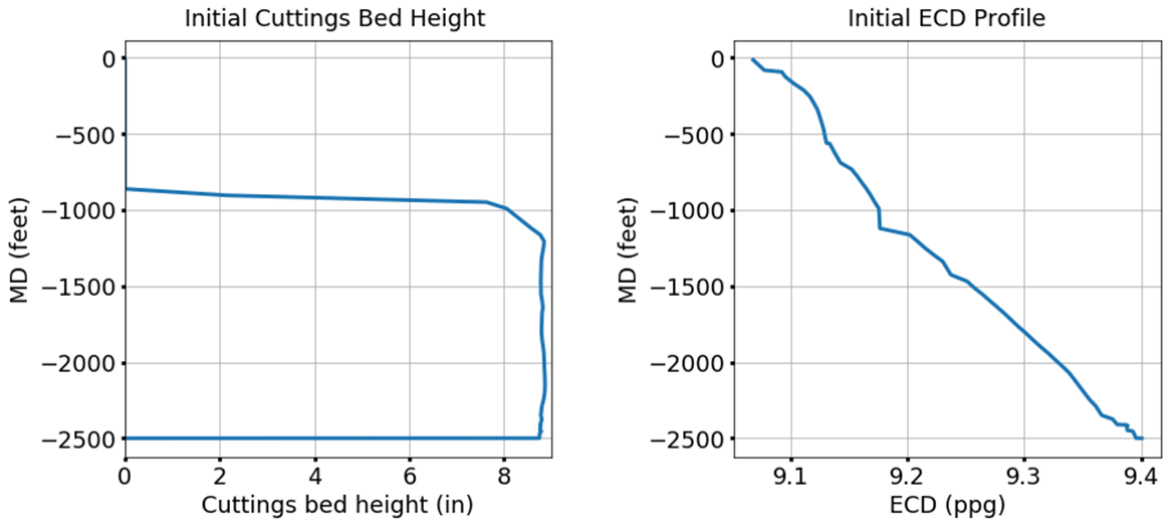


Figure 17. The state of the borehole at the end of the drilling operation during the second BHA run.

The final cuttings bed was around 9-inch, which is significantly higher than the limits specified in Figure 16a. Therefore, to ensure safe tripping operations without getting stuck, and to prepare the well for casing and cementing operations, these cuttings needed to be removed. A 50-min circulation cycle was then performed, the resulting state of which is represented by Equation (20), and Figure 18. The final cuttings bed height was still close to the allowed limit, and therefore non-optimal, explaining the issues encountered during tripping, casing, and cementing.

$$s_{TD\_circ} = \begin{Bmatrix} 0 \\ 2 \\ 3 \\ 3 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix} \quad (20)$$

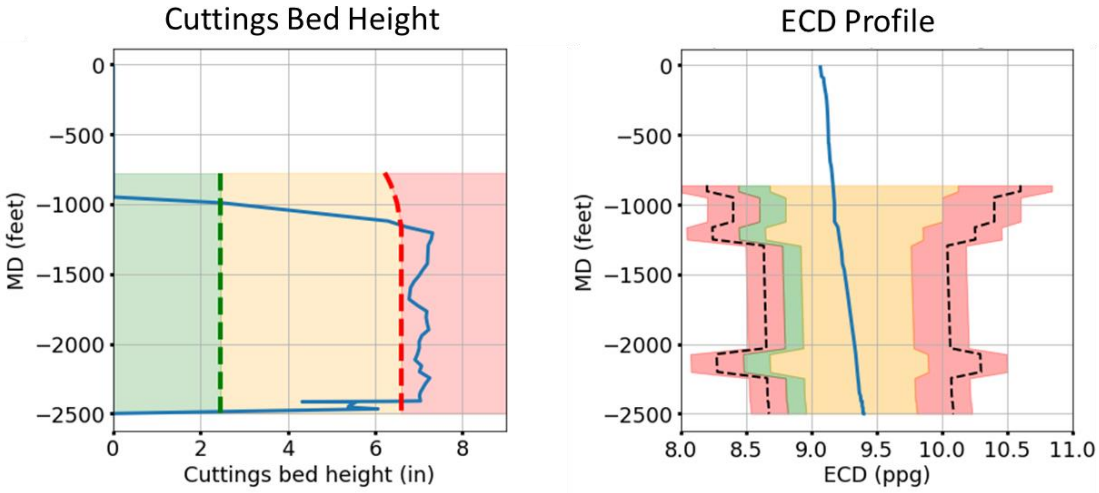


Figure 18. State of the borehole at the end of the circulation cycle.

4.2.2. Action Planning

The developed decision-making system was then used for planning the circulation cycle. The goal of the system, as previously discussed, is to reduce the cuttings bed height to safe limits while maintaining the ECD within the drilling margin (i.e., bed height and ECD need to be managed to in or around the green-shaded zones defined in Figure 16).

The following metrics are used to evaluate the different action sequences:

- The final system state;
- The average return value of the action sequence (V), which is the mean of total accumulated reward over the given trajectory that results from following the action sequence.

$$V = \frac{\sum_{t=1}^{T} R_t}{T} \tag{21}$$

For the well's actual circulation operation, the final state represented by Equation (20) was considerably far from the goal state. The V value for this action sequence was evaluated to be 0.74. The progression of the normalized Euclidean distance of the system state is shown in Figure 19. The green line at 0.2 corresponds to those states which have an absolute Euclidean distance of around two. With the definition of the state vector for this system, the theoretical maximum Euclidean distance is evaluated as 10.44, as detailed in Equation (22). For the ratio in Equation (11) to be 0.2, the maximum value for the current state's Euclidean distance can be two. For such states, the values of either four of their components have magnitude one, or one of their components has a magnitude of two, while the rest of the components are zero.

$$\sqrt{\sum_{i=i}^{9} s_{i\_max}^2} = \sqrt{4(4^2) + 5(3^2)} = 10.44 \tag{22}$$

The purpose of this 0.2 line is purely to serve as a visual aid, such that states closer to the line represent states closer to the goal state.

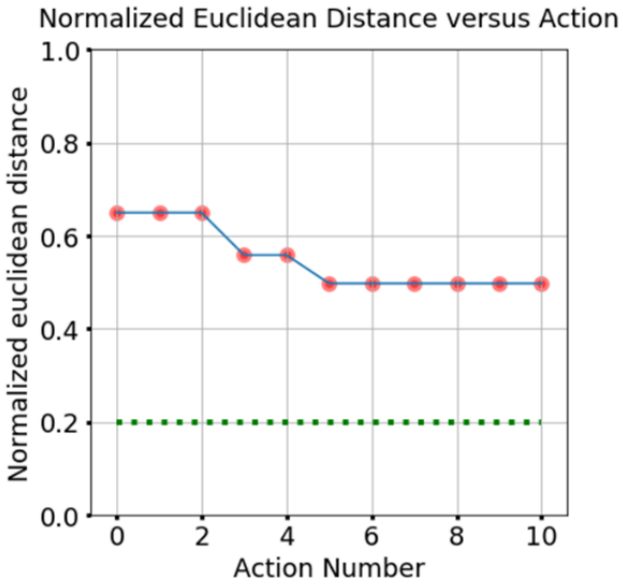


Figure 19. Progression of the normalized Euclidean distance (blue line) of the system states during the actual well circulation operation (shown by the green dotted line, here taken to be a state at a normalized Euclidean distance of 0.2).

4.2.3. Plan 1

Using the initializations defined in Tables 2 and 3, and with a C_{exp} value 1, the decision-engine was used to plan ahead for eight decision epochs (40-min) starting from the state s_{TD} (Equation (19)). Equation (23) represents the action sequence (a_{seq1}) recommended by the system. Selecting the right mud properties at the beginning of the circulation cycle is essential, since changing them is a time-consuming process. It is, therefore, highly impractical to adjust them in the middle of the circulation cycle. To this effect, the system suggested changing the mud properties (at the beginning of the circulation cycle) to mud density of 8.9 ppg, PV of 17.5 cP, and YP of 10.5 lbf/100 ft^2 from mud density 9.06 ppg, PV 11 cP, and YP 36.5 lbf/100 ft^2 initially.

$$a_{seq1} = \left\{\begin{matrix}1000\\0\\80\\8.9\\17.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1200\\0\\100\\8.9\\17.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1400\\0\\120\\8.9\\17.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1600\\0\\160\\8.9\\17.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1800\\0\\160\\8.9\\17.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1600\\0\\160\\8.9\\17.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1400\\0\\160\\8.9\\17.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1400\\0\\160\\8.9\\17.5\\10.5\end{matrix}\right\} \quad (23)$$

The predicted output state of the system is shown in Figure 20. The cuttings bed is almost entirely removed (as represented by the blue line), and the ECD values are very close to the desired regions throughout the well. Equation (24) represents the final state of the system.

$$s_{TD\_circ1} = \left\{\begin{matrix}0\\0\\0\\0\\1\\1\\1\\1\\1\end{matrix}\right\} \quad (24)$$

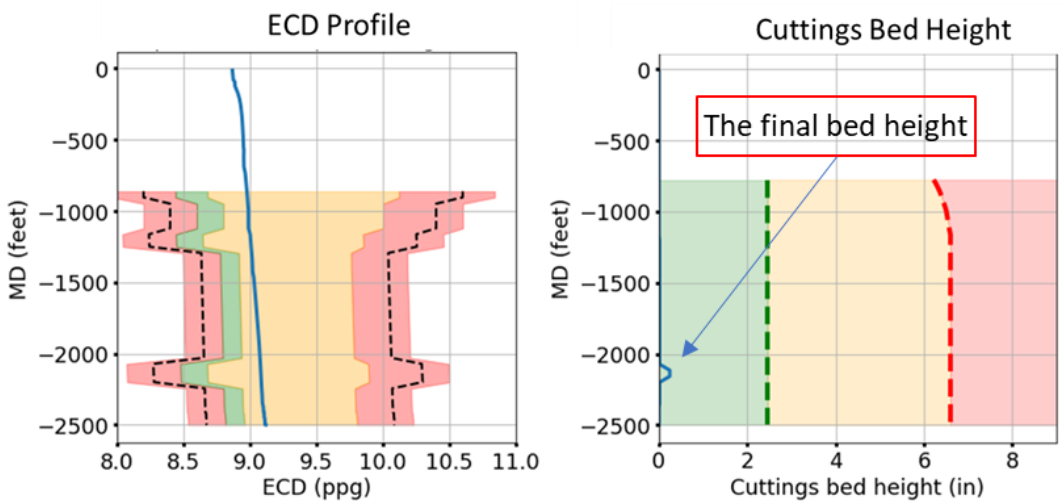


Figure 20. (**a**) ECD profile (**b**) Cuttings bed height (output state) for a_{seq1}.

Figure 21 illustrates the progression of the system with actions for the sequence a_{seq1}. Figure 21a shows the reduction in the cuttings bed height with different actions, and Figure 21b represents the progression of the normalized Euclidean distance with actions.

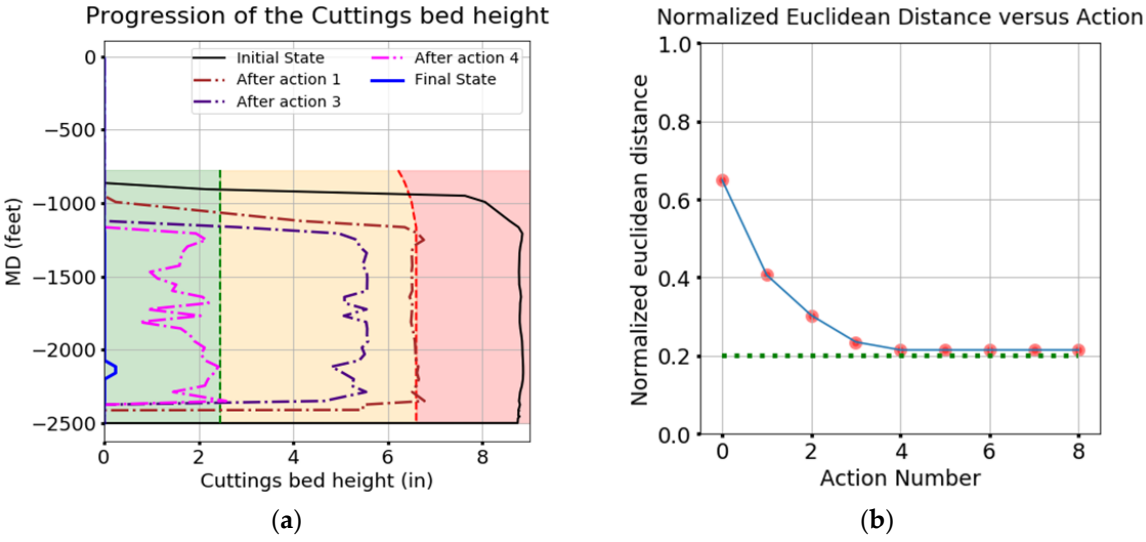


Figure 21. (**a**) Progression of the cuttings bed (**b**) Progression of the normalized Euclidean distance (blue line) of the system states for a_{seq1} to an acceptable goal state (shown by the green dotted line, here taken to be a state at a normalized Euclidean distance of 0.2).

By the fourth action, the system has already moved close to the 0.2 line (by cuttings bed being reduced to the goal state), where it stays until the end. Figure 22 shows the progression of the rewards associated with this action sequence. Thus, the V value calculated for a_{seq1} is 0.82.

Figure 22. Progression of the rewards associated with a_{seq1}.

An interesting observation in the action sequence is the system actively trying to manage the ECD by reducing the flowrate, after the cuttings bed has been removed. Initially, the system suggests increasing the flowrate and the RPM, which helps with cuttings bed removal (as can be seen by the reduced Euclidean distance), and then later tries to reduce the ECD.

4.2.4. Plan 2

Another planning operation was performed by changing the flowrate and RPM thresholds, as shown in Table 4. No changes were made to the mud density and rheology thresholds, and the weights associated with the different reward components were also unaltered.

Table 4. Modified flowrate and RPM thresholds for varying the action-space.

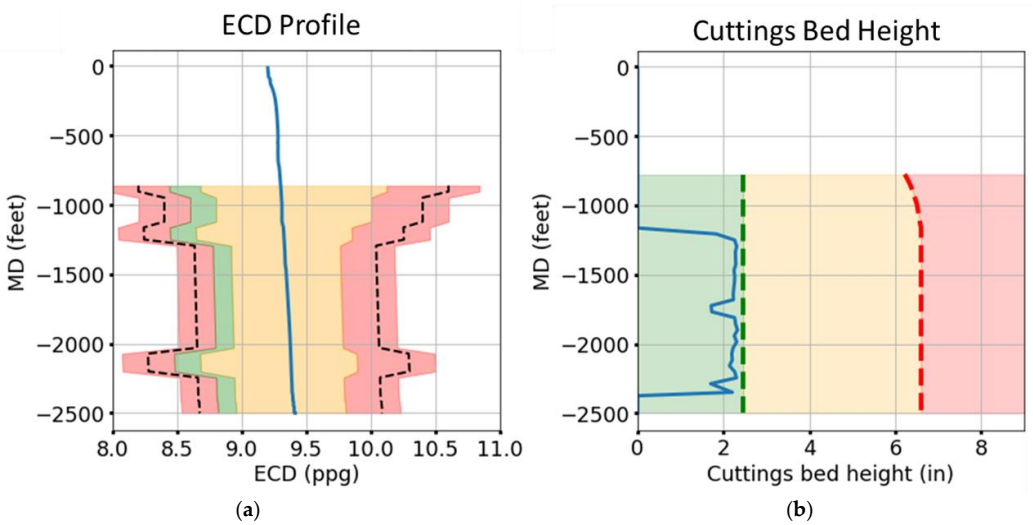
Control Variable	Number of Discrete Values	Range of Values
Flowrate (GPM)	10	[0, 1500]
Drillstring RPM (rev/min)	10	[0, 150]

Equation (25) represents the action sequence (a_{seq2}) output by the planning system, over a 40-min (eight decision epochs) interval. Due to the truncated flowrate and RPM thresholds, the system suggests different mud density and rheology than for a_{seq1} to help with the cuttings removal.

$$a_{seq2} = \left\{\begin{matrix}833\\0\\67\\9.24\\10.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1000\\0\\83\\9.24\\10.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1167\\0\\83\\9.24\\10.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1333\\0\\100\\9.24\\10.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1333\\0\\117\\9.24\\10.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1500\\0\\133\\9.24\\10.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1500\\0\\150\\9.24\\10.5\\10.5\end{matrix}\right\}, \left\{\begin{matrix}1333\\0\\150\\9.24\\10.5\\10.5\end{matrix}\right\} \quad (25)$$

The final state of the system (s_{TD_circ2}), represented by Equation (26), is shown in Figure 23.

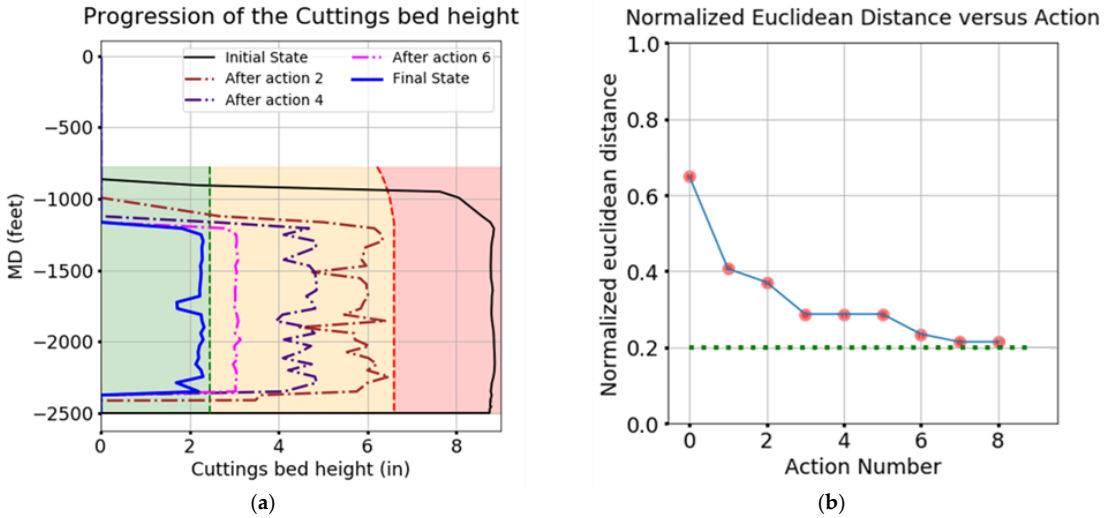
$$s_{TD\_circ2} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix} \quad (26)$$



(**a**) (**b**)

Figure 23. (**a**) ECD profile (**b**) Cuttings bed height.

The normalized Euclidean distance does decrease with time, but it only reaches the 0.2 line on the seventh action (Figure 24b). An interesting observation in Figure 24a is that the cuttings bed is not entirely removed but is reduced to the goal state (green-shaded zone) values. The ECD is also in the safe-but-suboptimal region, but it is much higher than for the first case due to a higher suggested mud density.



(**a**) (**b**)

Figure 24. (**a**) Progression of the cuttings bed (**b**) Progression of the normalized Euclidean distance (blue line) of the system states for a_{seq2} to an acceptable goal state (shown by the green dotted line, here taken to be a state at a normalized Euclidean distance of 0.2).

Although, the final state representations for both the plans (a_{seq1} and a_{seq2}) are the same, the average V value calculated for a_{seq2} is 0.79, which is lower than for a_{seq1}. The progression of rewards for the sequence a_{seq2} is shown in Figure 25.

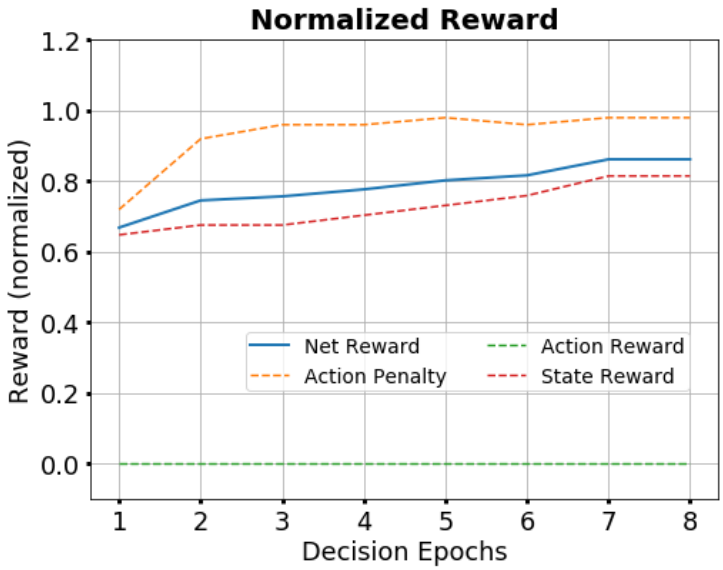


Figure 25. Progression of the rewards associated with a_{seq2}.

4.3. Discussion

For both plans a_{seq1} and a_{seq2}, the decision-engine was able to self-learn by simulating multiple episodes of experience and output action sequences that would have helped move the system towards the goal states. To summarize:

1. The non-holonomic nature of drilling operations combined with the decision-engine's long-term planning capabilities allow for more robust planning. An example of this is the system selecting appropriate mud properties at the beginning of the circulation cycle by evaluating multiple steps into the future. Both plans that were generated result in a better outcome than the actual plan generated and implemented by a human decision-maker.
2. Tuning the weights associated with the different reward components allows for prioritizing different objectives or different sections of the well over others.
3. Utilizing the domain-knowledge enriched MCTS allows for faster and more efficient planning, and well-defined heuristic functions make such planning systems implementable in the field. Exhaustively evaluating all nodes in the search tree to the eighth level (eight decision epochs or 40-min) would require over a hundred million simulations, as well as require storing the results of each state-action transition. This would be highly computationally and memory inefficient. MCTS, on the other hand, requires a number of simulations that are many orders of magnitude lower (only a few thousand in total), and all state-action transition results do not need to be stored. For the cases discussed in this paper, the planning algorithm, without any parallel processing or multi-threading on a standard laptop using a non-streamlined Python code, was able to generate these plans in under an hour.
4. Planning with higher values does result in the convergence of the state's Euclidean distance towards the 0.2-line, but it requires many more simulations. On the other hand, lower C_{exp} values introduce an element of bias depending on the order in which the nodes are added to the tree, which itself depends on the random rollout policy for MC simulations.

5. Conclusions

This paper proposes a method for the development of intelligent decision engines for well construction operations by utilizing an MDP formalism with the MCTS for action planning. This method is demonstrated by implementing a hole cleaning action planning system and comparing its performance against a human decision maker's performance. To the best of our knowledge, there are no published models or systems that address the issue of intelligent long-term sequential decision-making and action planning for the well construction domain. Such systems, however, have resulted in significant performance improvement across other domains such as game AI (such as in AlphaGo, AlphaZero), electric power distribution, dynamic resource allocation, etc. To summarize:

1. MCTS planning systems allow for a hybrid approach to managing conflicting objectives by combining the advantages of the exploration-exploitation trade-off offered by the MCTS, with domain-knowledge derived heuristics, thereby helping make better decisions.
2. A combination of the digital twin and a non-sparse reward function, with backpropagation of the episodic returns, allows the system to learn from simulated experience. A non-sparse reward structure ensures that the feedback received by the agent is frequent and meaningful, thereby speeding up the policy improvement process.
3. The underlying tree and rollout policies of the MCTS algorithm can be enhanced by using well-defined process-specific heuristics. This assists in improving the convergence rate of the system towards an optimal action sequence. For the hole cleaning system, the heuristic was designed to balance safety, performance, feasibility, and proximity constraints.
4. Utilizing such systems can aid in overall performance improvement by eliminating the need to wait on decisions, as well as suggesting optimal drilling parameters for the given wellbore condition.

Furthermore, decision engines can be developed for a multitude of other well construction applications, such as well control, drilling parameter optimization, tripping automation, cementing, etc. These decision engines can then be integrated into a rig's control system to automate monitoring, planning, and control of action variables such as drilling rate, drillstring rotation speed, tripping speed, flowrate, and mud properties. The ultimate goal is to automate important well construction tasks and optimize them by removing the (non-optimum) variability associated with subjective human-based decision-making.

Author Contributions: Conceptualization, G.S.S., P.A. and E.v.O.; methodology, G.S.S., P.P., P.A. and E.v.O.; software, G.S.S. and P.P.; validation, G.S.S., P.P. and P.A.; formal analysis, G.S.S.; investigation, G.S.S. and P.P.; resources, P.A. and E.v.O.; data curation, G.S.S. and P.A.; writing—original draft preparation, G.S.S.; writing—review and editing, P.P., P.A. and E.v.O.; visualization, G.S.S.; supervision, P.A. and E.v.O.; project administration, E.v.O.; funding acquisition, P.A. and E.v.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the RAPID Industry Affiliate Program (IAP) at the University of Texas at Austin.

Acknowledgments: The authors would like to thank the members of the Rig Automation and Performance Improvement in Drilling (RAPID) consortium at the University of Texas at Austin for their support of this research. We would also like to acknowledge our colleagues in the RAPID research group for valuable brainstorming sessions.

Conflicts of Interest: The authors declare no conflict of interest.

Unit Conversion

1 meter (m)	3.28 feet (ft)
1 meter/second (m/s)	11,811 feet/hour (ft/h)
1 psi	6894.76 Pa
1 ppg	119.83 kg/m^3
1 radian	57.2958 degrees
1 ft^3	0.02832 m^3
1 GPM	0.0000631 m^3/s
1 cP	0.001 Pa·s
1 lbf/100 ft^2	0.4788 Pa
1 lbs.	0.4536 Kg

Nomenclature

A	Action-space
A_{fe}	Action set associated with feasibility constraints (for hole cleaning heuristic)
A_{px}	Action set associated with proximity metric (for hole cleaning heuristic)
$A_{rollout}$	Reduced action space for a state during the rollout phase of the MCTS
A_s	Action space from state s
A_{s_t}	The action space of the node associated with state s_t
A_{sp}	Action set associated with safety and performance metrics (for hole cleaning heuristic)
A_{sq}	Action set associated with sequential metric (for hole cleaning heuristic)
a_{-1}	The most recently executed action
a_{agg}	Aggressive actions representing greater magnitude changes in values of action control variables relative to the most recent action
a_{reg}	Regular actions representing small changes in action control variables with respect to the most recent action
$a_{seq}1$	action sequence recommended by the system for the first planning case
$a_{seq}2$	action sequence recommended by the system for the second planning case
a_t	Action executed by the agent at time or decision epoch t
C_{exp}	The exploration factor in the UCT formula
D_{TVD}	True vertical depth for the measured depth D_{MD}
$d_{norm-euc}$	Normalized Euclidean distance for the current state
ECD	Equivalent circulation density (pounds per gallon or ppg)
$ECD_{inc.}$	Functional value of ECD in the inclination interval segment inc.
$\|exp\|$	The level at $node_{exp}$ is in the tree
flowrate	Rate of flow of the drilling mud through the drillstring controlled by a positive displacement reciprocating mud pump on the surface (GPM)
G_T	the net discounted return to T decision epochs
g	Acceleration due to gravity (9.81 m/s^2)
$H_{inc.}$	Functional value of the cuttings bed height in the inclination interval segment $incl$.
$incl$	Inclination angle range (degrees)
$N(s)$	Total number of visits to the state s
$N(s,a)$	Number of times action a has been taken from state s
N_{s_t}	Total number of visits to the node associated with state s_t
$node_0$	Root node corresponding to the starting state s_0
$node_{exp}$	Randomly expanded node from the leaf node during the expansion phase of the MCTS
$node_{leaf}$	Leaf node reached at the end of the selection phase of the MCTS
$node_t$	Node at decision epoch t
$P_{frictional_{pressure\_loss\_D_{MD}}}$	Frictional pressure drop in the annulus (Pa) at a measured depth D_{MD}
$P_{hydrostatic\_D_{TVD}}$	Hydrostatic Pressure (Pa) at a vertical depth of D_{TVD}
$P^a_{ss'}$	Transition probability of a system in the state s to the state $s\prime$ when an agent executes action a
PV	Plastic viscosity (cP)

$Q(s,a)$	Average value associated with implementing action a from state s
$Q_{\mathrm{UCT}}(s,a)$	The upper confidence bound or the urgency term in the UCT formula
R	Reward set
$R_{a\_density}$	Action reward component associated with the density value
$R_{a\_flowrate}$	Action reward component associated with the flowrate value
R_{a_PV}	Action reward component associated with the PV value
R_{a_ROP}	Action reward component associated with the ROP value
R_{a_RPM}	Action reward component associated with the RPM value
R_{a_YP}	Action reward component associated with the YP value
R_{ap}	Action transition-based penalty set
R_{ap_norm}	Normalized action penalty for the hole cleaning system
R_{ar}	Action value-based reward set
R_{ar_norm}	Normalized action reward for the hole cleaning system
$R_{ECDincl.}$	State reward component associated with ECD in the inclination interval *incl*.
$R_{density}$	Action Penalty component associated with changing mud density
$R_{flowrate}$	Action Penalty component associated with changing flowrate
$R_{Hincl.}$	State reward component associated with cuttings bed height in the inclination interval *incl*.
R_{net}	Net normalized reward function for the hole cleaning system
R_{PV}	Action Penalty component associated with changing mud PV
R_{ROP}	Action Penalty component associated with changing ROP
R_{RPM}	Action Penalty component associated with changing RPM
R_S	State transition-based reward set
R_{s_norm}	Normalized state reward for the hole cleaning system
R_{YP}	Action Penalty component associated with changing mud YP
ROP	Rate of penetration or drilling rate (ft/h)
RPM	Drillstring rotation speed (revs./min)
r_{greed}	Radius of the greed sphere for A_{px} evaluation
r_k	Reward associated with the kth state-action transition
S	State-space
SL	Stability limit (ppg)
s_{TD}	State of the hole cleaning system at the well TD
s_{TD_circ}	State of the wellbore after performing a circulation cycle at TD
s_{TD_circ1}	State of the wellbore after performing a circulation cycle at TD following a_{seq1}
s_{TD_circ2}	State of the wellbore after performing a circulation cycle at TD following a_{seq2}
s_{goal}	Goal or desired state for the MDP
s''_{goal}	The states near the goal state (for evaluating the A_{px} set)
s_t	State of the system at time or decision epoch t
T	Number of decision epochs to evaluate till in the future
t	Time step or decision epoch t
t_{max}	Time available for MCTS algorithm to plan
V	The average return value of an action sequence
W_{ap}	Weight set associated with the action transition penalty
W_{ap_norm}	Weight value associated with the normalized action penalty
W_{ar}	Weight set associated with the action value reward
W_{ar_norm}	Weight value associated with the normalized action reward
W_s	Weight set associated with state transition reward
W_{s_norm}	Weight value associated with the normalized state reward
WOB	Weight on bit (klbs.)
YP	Yield point (l bf/100 ft^2)
π	Policy or plan
$\pi_{heuristic\_a}$	Problem specific heuristic (probability of selecting action a)
π_{tree}	Tree policy—the action selected from a given node in the search tree during the selection phase of the MCTS
γ	Discount factor for return calculation

References

1. Sadlier, A.; Says, I.; Hanson, R. Automated Decision Support to Enhance While-Drilling Decision Making: Where Does it fit Within Drilling Automation? In Proceedings of the SPE/IADC Drilling Conference, Amsterdam, The Netherlands, 5–7 March 2013. [CrossRef]
2. Shokouhi, S.V.; Skalle, P. Enhancing decision making in critical drilling operations. In Proceedings of the SPE Middle East Oil and Gas Show and Conference, Manama, Bahrain, 15–18 March 2009; pp. 809–816. [CrossRef]
3. Mayani, M.G.; Baybolov, T.; Rommetveit, R.; Ødegaard, S.I.; Koryabkin, V.; Lakhtionov, S. Optimizing drilling wells and increasing the operation efficiency using digital twin technology. In Proceedings of the IADC/SPE International Drilling Conference and Exhibition, Galveston, TX, USA, 3–5 March 2020. [CrossRef]
4. Nadhan, D.; Mayani, M.G.; Rommetveit, R. Drilling with digital twins. In Proceedings of the IADC/SPE Asia Pacific Drilling Technology Conference and Exhibition, Bangkok, Thailand, 27–29 August 2018. [CrossRef]
5. Saini, G.S.; Ashok, P.; van Oort, E.; Isbell, M.R. Accelerating well construction using a digital twin demonstrated on unconventional well data in North America. In Proceedings of the Unconventional Resources Technology Conference, Houston, TX, USA, 23–25 July 2018. [CrossRef]
6. Danner, G.E. Using knowledge graphs to enhance drilling operations. In Proceedings of the Offshore Technology Conference, Houston, TX, USA, 4–7 May 2020. [CrossRef]
7. Miller, P.; Gouveia, J. Applying decision trees to improve decision quality in unconventional resource development. In Proceedings of the SPE Annual Technical Conference and Exhibition, Calgary, AB, Canada, 30 September–2 October 2019. [CrossRef]
8. Cordoso, J.V.; Maidla, E.E.; Idagawa, L.S. Problem detection during tripping operations in horizontal and directional wells. *SPE Drill. Complet.* **1995**, *10*, 77–83. [CrossRef]
9. Zhang, F.; Miska, S.; Yu, M.; Ozbayoglu, E.; Takach, N.; Osgouei, R.E. Is Well Clean Enough? A Fast Approach to Estimate Hole Cleaning for Directional Drilling. In Proceedings of the SPE/ICoTA Coiled Tubing & Well Intervention Conference & Exhibition, The Woodlands, TX, USA, 24–25 March 2015. [CrossRef]
10. Abbas, A.K.; Almubarak, H.; Abbas, H.; Dawood, J. Application of machine learning approach for intelligent prediction of pipe sticking. In Proceedings of the Society of Petroleum Engineers—Abu Dhabi International Petroleum Exhibition and Conference 2019 (ADIP 2019), Abu Dhabi, UAE, 11–14 November 2019. [CrossRef]
11. Da Cruz Mazzi, V.A.; Dumlao, V.C.Q.; Mourthé, A.C.L.; Elshahawi, H.; Kim, I.; Lumens, P. Machine Learning-Enabled Digital Decision Assistant for Remote Operations. In Proceedings of the Offshore Technology Conference, Houston, TX, USA, 4–7 May 2020. [CrossRef]
12. Fjellheim, R. Smart collaboration for decision making in drilling. In Proceedings of the SPE Middle East Intelligent Energy Conference and Exhibition, Manama, Bahrain, 28–30 October 2013. [CrossRef]
13. Saini, G.S.; Chan, H.C.; Ashok, P.; van Oort, E.; Behounek, M.; Thetford, T.; Shahri, M. Spider bots: Database enhancing and indexing scripts to efficiently convert raw well data into valuable knowledge. In Proceedings of the Unconventional Resources Technology Conference, Houston, TX, USA, 23–25 July 2018. [CrossRef]
14. Chan, H.C.; Lee, M.M.; Saini, G.S.; Pryor, M.; van Oort, E. Development and Validation of a Scenario-Based Drilling Simulator for Training and Evaluating Human Factors. *IEEE Trans. Hum.-Mach. Syst.* **2020**, *50*, 327–336. [CrossRef]
15. Ji, Y.; Wang, J.; Xu, J.; Fang, X.; Zhang, H. Real-Time Energy Management of a Microgrid Using Deep Reinforcement Learning. *Energies* **2019**, *12*, 2291. [CrossRef]
16. Kuznetsova, E.; Li, Y.F.; Ruiz, C.; Zio, E.; Ault, G.; Bell, K. Reinforcement learning for microgrid energy management. *Energy* **2013**, *59*, 133–146. [CrossRef]
17. Guan, J.; Tang, H.; Wang, K.; Yao, J.; Yang, S. A parallel multi-scenario learning method for near-real-time power dispatch optimization. *Energy* **2020**, *202*, 117708. [CrossRef]
18. Jasmin, E.A.; Imthias Ahamed, T.P.; Jagathy Raj, V.P. Reinforcement Learning approaches to Economic Dispatch problem. *Int. J. Electr. Power Energy Syst.* **2011**, *33*, 836–845. [CrossRef]
19. Qi, X.; Luo, Y.; Wu, G.; Boriboonsomsin, K.; Barth, M. Deep reinforcement learning enabled self-learning control for energy efficient driving. *Transp. Res. Part C Emerg. Technol.* **2019**, *99*, 67–81. [CrossRef]
20. Vandael, S.; Claessens, B.; Ernst, D.; Holvoet, T.; Deconinck, G. Reinforcement Learning of Heuristic EV Fleet Charging in a Day-Ahead Electricity Market. *IEEE Trans. Smart Grid* **2015**, *6*, 1795–1805. [CrossRef]
21. Kang, D.-J.; Park, J.H.; Yeo, S.-S. Intelligent Decision-Making System with Green Pervasive Computing for Renewable Energy Business in Electricity Markets on Smart Grid. *EURASIP J. Wirel. Commun. Netw.* **2009**, *2009*, 247483. [CrossRef]
22. Arulkumaran, K.; Cully, A.; Togelius, J. AlphaStar: An Evolutionary Computation Perspective. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, 13–17 July 2019; pp. 314–315. [CrossRef]
23. Moerland, T.M.; Broekens, J.; Plaat, A.; Jonker, C.M. A0C: Alpha Zero in Continuous Action Space. *arXiv* **2018**, arXiv:1805.09613.
24. Ontanon, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; Preuss, M. A survey of real-time strategy game AI research and competition in starcraft. *IEEE Trans. Comput. Intell. AI Games* **2013**, *5*, 293–311. [CrossRef]
25. Silver, D.; Sutton, R.S.; Müller, M. Temporal-difference search in computer Go. *Mach. Learn.* **2012**, *87*, 183–219. [CrossRef]
26. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of Go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef] [PubMed]

27. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **2018**, *362*, 1140–1144. [CrossRef] [PubMed]
28. Ahmed, O.S.; Aman, B.M.; Zahrani, M.A.; Ajikobi, F.I.; Aramco, S. Stuck Pipe Early Warning System Utilizing Moving Window Machine Learning Approach. In Proceedings of the Abu Dhabi International Petroleum Exhibition & Conference, Abu Dhabi, UAE, 11–14 November 2019.
29. Forshaw, M.; Becker, G.; Jena, S.; Linke, C.; Hummes, O. Automated hole cleaning monitoring: A modern holistic approach for NPT reduction. In Proceedings of the International Petroleum Technology Conference, Dhahran, Saudi Arabia, 13–15 January 2020. [CrossRef]
30. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006. [CrossRef]
31. Poole, D.L.; Mackworth, A.K. *Artificial Intelligence: Foundations of Computational Agents*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2017.
32. Bennett, B. Graphs: Breadth First Search vs. Depth First Search [WWW Document]. 2019. Available online: https://medium.com/javascript-in-plain-english/graphs-breadth-first-search-vs-depth-first-search-d9908c560642 (accessed on 1 July 2020).
33. Mussmann, S.; See, A. Graph Search Algorithms. 2017. Available online: http://cs.stanford.edu/people/abisee/gs.pdf (accessed on 1 June 2020).
34. Alija, A.S. Analysis of Dijkstra's and A* Algorithm to Find the Shortest Path. Ph.D. Thesis, Universiti Tun Hussein Onn Malaysia, Batu Pahat, Malaysia, 2015.
35. Reddy, H. Path Finding—Dijkstra's and A* Algorithm's. *Int. J. IT Eng.* **2013**, 1–15.
36. Bhaumik, D.; Khalifa, A.; Green, M.C.; Togelius, J. Tree Search vs Optimization Approaches for Map Generation. *arXiv* **2019**, arXiv:1903.11678.
37. Cui, X.; Shi, H. A*-based Pathfinding in Modern Computer Games. *Int. J. Comput. Sci. Netw. Secur.* **2011**, *11*, 125–130.
38. Klein, S. Attacking SameGame using Monte-Carlo Tree Search: Using Randomness as Guidance in Puzzles. Master's Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2015.
39. Li, L.K.; Seng, K.P.; Yeong, L.S.; Ch'ng, S.I.; Li-Minn, A. The Boundary Iterative-Deepening Depth-First Search Algorithm. *Int. J. Adv. Comput. Sci. Its Appl.* **2014**, *4*, 45–50.
40. Korkmaz, M.; Durdu, A. Comparison of optimal path planning algorithms. In Proceedings of the 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET), Slavske, Ukraine, 20–24 January 2018; pp. 255–258. [CrossRef]
41. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018.
42. Silver, D. *Reinforcement Learning and Simulation-Based Search in Computer Go*; University of Alberta: Alberta, AB, Canada, 2009.
43. Cayeux, E.; Mihai, R.; Carlsen, L.; Stokka, S. An approach to autonomous drilling. In Proceedings of the IADC/SPE International Drilling Conference and Exhibition, Galveston, TX, USA, 3–5 March 2020. [CrossRef]
44. Mitchell, R.F.; Miska, S.Z. *Fundamentals of Drilling Engineering*; Society of Petroleum Engineers: Richardson, TX, USA, 2011.
45. James, S.; Konidaris, G.; Rosman, B. An analysis of Monte Carlo tree search. In Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 3576–3582.
46. Vodopivec, T.; Samothrakis, S.; Ŝter, B. On Monte Carlo Tree Search and Reinforcement Learning. *J. Artif. Intell. Res.* **2017**, *60*, 881–936. [CrossRef]
47. Browne, C.B.; Powley, E.; Whitehouse, D.; Lucas, S.M.; Cowling, P.I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; Colton, S. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comput. Intell. AI Games* **2012**, *4*, 1–43. [CrossRef]
48. Kocsis, L.; Szepesvári, C.; Willemson, J. *Improved Monte-Carlo Search*; Technical. Repotrt 1; University of Tartu: Tartu, Estonia, 2006.
49. Vodopivec, T. *Monte Carlo Tree Search Strategies*; Maastricht University: Maastricht, The Netherlands, 2018.
50. Chaslot, G.; Bakkes, S.; Szitai, I.; Spronck, P. Monte-carlo tree search: A New Framework for Game AI. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Stanford, CA, USA, 22–24 October 2008; pp. 389–390.
51. Chaslot, G.M.J.-B.; Winands, M.H.M.; van den Herik, H.J.; Uiterwijk, J.W.H.M.; Bouzy, B. Progressive Strategies for Monte-Carlo Tree Search. *New Math. Nat. Comput.* **2008**, *4*, 343–357. [CrossRef]
52. Gelly, S.; Silver, D. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artif. Intell.* **2011**, *175*, 1856–1875. [CrossRef]
53. Efroni, Y.; Dalal, G.; Scherrer, B.; Mannor, S. How to Combine Tree-Search Methods in Reinforcement Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 3494–3501. [CrossRef]
54. Puterman, M.L. *Markov Decision Processes, Wiley Series in Probability and Statistics*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1994. [CrossRef]
55. Bourgoyne, A.T.; Millheim, K.K.; Chenevert, M.E.; Young, F.S. *Applied Drilling Engineering*; Society of Petroleum Engineers: Richardson, TX, USA, 1986.
56. Ozbayoglu, M.E.; Saasen, A.; Sorgun, M.; Svanes, K. Effect of Pipe Rotation on Hole Cleaning for Water-Based Drilling Fluids in Horizontal and Deviated Wells. In Proceedings of the IADC/SPE Asia Pacific Drilling Technology Conference and Exhibition, Jakarta, Indonesia, 25–27 August 2008. [CrossRef]

57. Sanchez, R.A.; Azar, J.J.; Bassal, A.A.; Martins, A.L. The Effect of Drillpipe Rotation on Hole Cleaning During Directional Well Drilling. In Proceedings of the SPE/IADC Drilling Conference, Amsterdam, The Netherlands, 4–6 March 1997. [CrossRef]
58. Baldino, S.; Osgouei, R.E.; Ozbayoglu, E.; Miska, S.; Takach, N.; May, R.; Clapper, D. Cuttings settling and slip velocity evaluation in synthetic drilling fluids. In Proceedings of the Offshore Mediterranean Conference and Exhibition, Ravenna, Italy, 25–27 March 2015.
59. Cayeux, E.; Mesagan, T.; Tanripada, S.; Zidan, M.; Fjelde, K.K. Real-time evaluation of hole-cleaning conditions with a transient cuttings-transport model. *SPE Drill. Complet.* **2014**, *29*, 5–21. [CrossRef]
60. Gul, S.; van Oort, E.; Mullin, C.; Ladendorf, D. Automated Surface Measurements of Drilling Fluid Properties: Field Application in the Permian Basin. *SPE Drill. Complet.* **2020**, *35*, 525–534. [CrossRef]
61. Nazari, T.; Hareland, G.; Azar, J.J. Review of Cuttings Transport in Directional Well Drilling: Systematic Approach. In Proceedings of the SPE Western Regional Meeting, Anaheim, CA, USA, 27–29 May 2010. [CrossRef]
62. Saini, G.S.; Ashok, P.; van Oort, E. Predictive Action Planning for Hole Cleaning Optimization and Stuck Pipe Prevention Using Digital Twinning and Reinforcement Learning. In Proceedings of the IADC/SPE International Drilling Conference and Exhibition, Galveston, TX, USA, 3–5 March 2020. [CrossRef]
63. De Oliveira, G.L.; Zank, C.A.C.; De Costa, A.F.S.; Mendes, H.M.; Henriques, L.F.; Mocelin, M.R.; De Oliveira Neto, J. Offshore drilling improvement through automating the rig state detection process—Implementation process history and proven success. In Proceedings of the IADC/SPE Drilling Conference and Exhibition, Fort Worth, TX, USA, 1–3 March 2016. [CrossRef]