Please fill in the name of the event you are preparing this manuscript for.	2025 SPE Reservoir Simulation Conference
Please fill in your 6-digit SPE manuscript number.	SPE-223917-MS
Please fill in your manuscript title.	Optimal Drilling Scheduling in Field Development Planning by Deep Reinforcement Learning

Please fill in your author name(s) and company affiliation.

Given Name	Surname	Company
Nicolás Javier	Cárdenas Pantoja	Energy Department, Politecnico di Milano, Milano, Italy
Adam	Abdin	Industrial Engineering Research Department, CentraleSupélec Engineering School, University of Paris-Saclay, Gif-sur-Yvette, France
Piero	Baraldi	Energy Department, Politecnico di Milano, Milano, Italy
Luca	Pinciroli	Energy Department, Politecnico di Milano, Milano, Italy
Alice	Forello	Eni S.p.A., via Emilia 1, San Donato Milanese, 20097 Italy
Laura	Dovera	Eni S.p.A., via Emilia 1, San Donato Milanese, 20097 Italy
Enrico	Zio	[1]Centre de Recherche sur les Risques et les Crises (CRC), MINES Paris-PSL, Sophia Antipolis, France [2]Energy Department, Politecnico di Milano, Milano, Italy

Abstract

We consider the problem of optimizing the drilling scheduling in Field Development Planning. The objective is to identify the well drilling sequence and well types that maximize the project Net Present Value (NPV) properly handling uncertainty on our knowledge of the reservoir geological parameters. This Sequential Decision Problem (SDP) is modelled as a Markov Decision Problem (MDP) and solved using Deep Reinforcement Learning. Specifically, we develop an approach based on Deep Q-Networks (DQN), where an online Neural Networks (NN) learns and selects the most suitable drilling action, and a target NN estimates the expected future NPV (Q-value). A soft-update mechanism is adopted to gradually blend the weights of the target NN with those of the online NN with the objective of improving training stability. The proposed method is tested on a synthetic case study that simulates a real drilling case. It is shown to outperform a state-of-the-art DQN implementation, achieving a larger NPV and solutions more robust to the uncertainty on the reservoir geological properties.

1. Introduction

The Oil and Gas (O&G) extraction industry operates in highly dynamic and uncertain environments, where factors such as geological variability, fluctuating commodity prices, and regulatory changes can significantly impact project outcomes. Field Development Planning (FDP) entails the decisions about the placement and design of wells and other critical infrastructures, with the objective of optimizing the process of extraction of subsurface resources (Mirzaei-Paiaman et al., 2022). Specifically, FDP involves selecting well types, determining drilling locations, designing wells, formulating production strategies, and planning infrastructure development, while balancing geological, engineering, economic, safety, and environmental considerations (Muther et al., 2022).

This work proposes a method for optimizing drilling schedules. The objective is to maximize the profit over the entire life cycle of an oil project. Since the decisions on well placement and drilling order influence the performance of the reservoir over time, the optimization problem is formalized as a Sequential Decision Problem

(SDP), where decisions are made in consecutive steps, with each choice affecting future outcomes and subsequent decisions (Ferguson, 1967).

Several approaches have been proposed to address the drilling scheduling problem as a SDP (He et al., 2022; Nasir, 2020; Paola et al., 2020). They typically do not consider the uncertainty on geological properties and provide solutions which are difficult to generalize to other reservoirs. Given the interest of the O&G industry in drilling scheduling, the Olympus challenge was launched in (Fonseca et al., 2020) to provide a basis for assessing the performance of methods for drilling scheduling optimization. This challenge provides 50 possible realizations of the parameters describing the geological properties of a reservoir and requires optimizing the drilling schedule with the objective of maximizing the expected Net Present Value (NPV). Several optimization approaches have been already developed for the Olympus challenge. As discussed in Section 2.1, their performance and applicability are limited by inadequate handling of uncertainty on geological properties and inability to transfer knowledge from one reservoir to another.

The present work formulates the SDP of optimizing the drilling schedule as a Markov Decision Problem (MDP) and solves it using Deep Reinforcement Learning (DRL). DRL extends traditional reinforcement learning by leveraging neural networks to map states to actions (Yu et al., 2022). Its proven success in handling optimization under uncertainty across various domains, such as game playing (OpenAI et al., 2019), autonomous vehicles (Kiran et al., 2022), finance (Carta et al., 2021) and energy management problems (Q. Zhang et al., 2019), makes DRL a suitable choice for this work. Specifically, we employ Deep Q-Networks (DQN) (Mnih et al., 2013), a DRL algorithm that integrates Q-learning with deep neural networks, enabling the learning of complex policies directly from raw data.

A limitation of DQN is that the agent learning process may become unstable due to abrupt and large updates of the network weights, which, in turn, causes oscillating and diverging evolutions of the reward during training (Halat & Ebadzadeh, 2021). In this context, the main contribution of the proposed method is the integration of a soft-update mechanism (Lillicrap et al., 2015) into DQN. The idea is to ensure a more stable and effective learning process by gradually blending the weights of the target network with those of the online network.

The method is developed considering a synthetic case study that simplifies the complexities of real drilling scheduling problems. In this scenario, oil production from a well is assumed to depend on the capability of maintaining a favorable pressure at the bottom of the well, which, in turn, is influenced by the operation of surrounding wells. Unlike real reservoirs, where pressure propagation follows complex dynamics governed by geological properties, this work assumes that the pressure at any given reservoir location is the sum of contributions from the surrounding wells, each one inversely proportional to the distance between well and location.

Two experiments are designed and conducted to evaluate the proposed method. The first one compares the performance of the proposed method with that of a standard implementation of DQN (Paola et al., 2020), which does not employ the soft-update mechanism to stabilize the learning process. In the second experiment, the effect of the uncertainties affecting the reservoir properties on the performance of the proposed method is investigated.

The remaining sections are organized as follows. Section 2 discusses the scientific literature about FDP Optimization and the current approaches for drilling scheduling by Reinforcement Learning. Section 3.1 states the FDP optimization problem, whereas Sections 3.2 and 3.3 formulate the SDP as a MDP. Section 4 describes the optimization method and the developed DRL technique. Section 5 describes the case study and Section 6 reports the obtained result. Finally, Section 7 identifies potential future steps of the research and concludes the work.

2. Related Works

Related works about FDP optimization and DRL approaches for drilling scheduling are discussed in Sections 2.1 and 2.2, respectively.

2.1 FDP Optimization

The optimization of the drilling sequence was modeled as a SDP in (Wang & Oliver, 2019). The proposed solution employed an A* search algorithm (Hart et al., 1968) guided by an heuristic function previously learned on similar problems. Although this approach provides satisfactory results, it is not able to dynamically adapt to changes in the environment. The Robust Optimization method, which is based on an ensemble of reservoir models, each one predicting a corrective term to be applied to a mean reservoir model, is developed in (Wang & Oliver, 2021). Its main limitation is that it does not directly handle the uncertainty on the reservoir properties. A framework based on Iterative Discrete Latin Hypercube (IDLHC), that uses as starting point a guess based on experience and deterministic sequential optimization was developed in (Mirzaei-Paiaman et al., 2022), the obtained results show that the framework is not able to catch the dynamic feedbacks in the reservoir and lacks of adaptability other reservoirs.

Other methods proposed for the solution of the Olympus challenge are summarized in Table 1 considering the number of realizations of the geological properties utilized and the number of times the optimizer runs the reservoir model.

Reference	Method	Number of utilized realizations of the geological properties	Number of times that the model of the reservoir has been run during the optimization
(Barros et al., 2020)	Leeuwenburgh priority control	50	1300
(Bergey, 2020)	Heuristic + Latin Hypercube Sampling	1	280
(Chang et al., 2020)	Line Search Derivative-Free + Stochastic Simplex Approximate Gradient	50	1750
(Silva et al., 2020)	Genetic Algorithm (GA)	50	105000
(Schulze-Riegert et al., 2020)	Probabilistic well ranking	1	2500

Table 1. Olympus challenge: Optimization approaches proposed for drilling scheduling

2.2 Deep Reinforcement Learning for Drilling Scheduling

A complete review of RL and its use for the solution of a large variety of optimization problems can be found in (Sutton & Barto, 2018). In general, DRL techniques require to define:

- the state space, which comprises static and dynamic information;
- the action space, i.e., the set of possible actions that can be performed in a state;
- the learning agent, i.e., the algorithm that interacts with the environment.

Table 2 reports the works addressing the problem of optimal drilling scheduling using approaches based on DRL. These works are classified considering state space, action space and learning agent. Dynamic Programming (DP) and DRL were proposed to optimize the drilling policy in (Paola et al., 2020). Specifically, a Deep Recurrent Neural Network (DRNN) is used as a surrogate model of the reservoir simulator to reduce the computational effort of estimating flows in the reservoir. A DRL approach was applied to optimize the drilling policy using Proximal Policy Optimization (PPO) with a convolutional neural network (CNN) in (Nasir, 2020). A reservoir model defined by the parameters of the equations governing the two-phase flow was developed and used to train a CNN within a DRL algorithm in (Nasir et al., 2021). An approach for Field Development Optimization (FDO) based on DRL was developed in (He et al., 2022). Despite these developments, an approach able to satisfactorily handle uncertainty on the reservoir properties has not yet been proposed. Additionally, current approaches were developed considering a specific reservoir and cannot be easily transferred to other reservoirs.

Reference	State Space	Action Space	Agent
(Paola et al., 2020)	Not Reported	· Location · Type of well (INJ/PROD)	DQN
(Nasir, 2020)	Static information: · Permeability Dynamic information: · Pressure · Saturation · Location map · Drilling stage number · Number of producer and injector wells already drilled	· Location · To drill or not · Type of well (INJ/PROD)	PPO
(He et al., 2022)	Static information: · Rock and fluid compressibility · Transmissibility · Impact of depth on fluid potential · Drilling cost · Productivity Index · Producer Bottomhole · Active Cell Indicator Dynamic information: · Pressure · Total compressibility · Time remaining in the planning horizon	· Location · To drill or not · Type of well (INJ/PROD)	PPO
(Nasir et al., 2021)	Static information: · Transmissibility · Productivity Index · Well Cost to Net Oil Price Ratio · Water Production Cost · Maximum Liquid Production Rate · Drilling Time · Water Cut Constraint Dynamic information: · Pressure · Saturation · Oil Accumulation · Water Mobility · Oil Mobility · Producer Drawdown · Well Location Mask · Current Discount Rate · Time remaining in the planning horizon	· Location · To drill or not · Type of well (INJ/PROD)	PPO

Table 2. DRL-based solutions to the drilling scheduling problem

3. Problem Statement and Formulation

The proposed method is based on (Cárdenas Pantoja et al., 2024), where the capacity of DQN to find the optimal policy for drilling scheduling is demonstrated in a reduced synthetic case study that does not consider uncertainty in the reservoir properties.

3.1 Problem Statement

We consider a reservoir whose geological parameters, Φ, are uncertain and distributed according to an unknown distribution. We assume that N_r realizations, ϕ^r, $r = 1, \dots, N_r$, of Φ are provided by reservoir engineers on the basis of the results of a geological study of the field. The life cycle of the O&G recovery project is divided into T

time-steps of duration Δt. The objective is to identify the drilling sequence and the types of well (producer or injector) which maximize the NPV:

$$NPV = \sum_{t=1}^{T} \frac{r_t}{(1+d)^t} \tag{1}$$

where d is the discount factor, and r_t is the difference between revenue and expenses incurred during the time interval t. Producer wells exploit the reservoir, whereas injector wells inject water or gas to sustain reservoir pressure.

The operational expenses are the platform investment and the cost of drilling wells and injecting water. The revenue is directly proportional to the flow of oil extracted by the producer wells. Specifically, the oil production of the i^{th} well in the time interval t, $o_{t,i}$, is assumed to be a function, f, of the Bottom Hole Pressure (BHP), $p_{t,i}$, in correspondence of the i^{th} well and the reservoir geological parameters Φ

$$o_{t,i} = f(p_{t,i}, \Phi)$$

Similarly to the Olympus challenge (Fonseca et al., 2020), we further assume that:

- only single wellbores, with no sidetracks, can be drilled;
- wells cannot be converted from producer to injector or vice versa after they start operating;
- the number, N_w, of wells to drill is fixed;
- well trajectories and well locations are known.

3.2 Problem Formulation

Due to the sequential nature of the decisions involved, the problem of optimizing the drilling sequence is a SDP, which, in turn, is formulated as a MDP (Bellman, 1957) defined by the sets $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$, γ, where:

- $\mathcal{S}$ is the state space;
- $\mathcal{A}$ is the action space;
- $\mathcal{P}$ is the transition probability space. Specifically, the generic element of the matrix: $p(s'|s,a)$ is the probability to transit from state s to state s' by performing action a;
- $\mathcal{R}$ is the set of possible rewards, i.e., $r(s,a,s')$ is the expected immediate reward of a transition from s to s' as a consequence of action a;
- $\gamma \in [0,1]$ is the discount factor used to compute the present value of future rewards.

Sections 3.3.1, 3.3.2 and 3.3.3 define the state space, $\mathcal{S}$, the action space, $\mathcal{A}$, and the reward function, $\mathcal{R}$, respectively. Note that the learning agent directly interacts with a simulation representing the environment, hence the explicit definition of the transition function $\mathcal{P}$ is not required since transitions are managed by the environment (simulation) itself.

3.3 Definition of State, Action and Reward Spaces

1. State Space

The state s_t of the system in time-step t is the concatenation of the states, $l_{t,i}$, $i = 1, \dots, N_w$, of the well locations in the same time-step:

$$s_t = \left[l_{t,i}\right]_{i=1,\dots,N_w}, \quad t = 0, \dots, T \tag{3}$$

The state, $l_{t,i}$, of the i^{th} well location at time-step t is defined as (Cárdenas Pantoja et al., 2024):

Loc_i : two-dimensional vector of the Cartesian coordinates of the well location;

$Type_{t,i}$: ternary variable set equal to 1 if the well is a producer, -1 if the well is an injector or 0 if the well has not yet been drilled;

$p_{t,i}$: scalar equal to the BHP in correspondence of well i at the beginning of time-step t;

$ow_{t,i}$: scalar equal to the fluid produced or injected by well i in time-step t. It is positive if the well is a producer and negative if it is an injector.

2. Action Space

$\mathcal{A}_t$ is the set containing well locations and type (producer or injector) possible at the beginning of time-step t. It is dynamically defined as:

$$\mathcal{A}_{t+1} = \mathcal{A}_t \setminus \{a_t\} \tag{6}$$

where $a_t \in \mathcal{A}_t$ is a tuple containing the well drilled during time-step t and its type:

$$a_t = [w_i, Type_i] \tag{7}$$

For example, $a_t = (w_i,\ 1)$ indicates that the well drilled during time-step t is w_i and its type is producer.

3. Reward

The reward, $\mathrm{r}(s,\ a,\ s')$ of a transition from state s to state s' as a consequence of action a in time-step t is:

$$r(s,\ a,\ s') \doteq r_t = \frac{\left(\sum_{i \in N_w} o_{t,i}\right) v_o - \left(\sum_{i \in N_w} w_{t,i}\right) v_w}{(1+d)^t} \tag{8}$$

where,

$o_{t,i}$: is the oil produced by well i in time-step t;

$w_{t,i}$: is the water injected by well i in time-step t;

v_o : is the oil price per unit sold;

v_w : is the cost of injecting water per unit.

The total reward of a complete episode is the NPV obtained in the time horizon T:

$$NPV = \sum_{t=1}^{T} r_t \tag{9}$$

4. Proposed Solution Method

Deep Reinforcement Learning

DRL is a machine learning paradigm where an agent learns to make sequential decisions by interacting with an environment. In DRL, the agent aims to maximize a cumulative reward by taking actions that influence future states of the environment. Unlike traditional reinforcement learning (RL), DRL leverages deep neural networks to approximate the mapping between states and actions in complex, high-dimensional spaces. At each time-step,

the agent observes the current state, selects an action, and receives feedback in the form of a reward, which quantifies the immediate benefit or cost of the chosen action. The ultimate goal of DRL is to find an optimal policy, i.e., a mapping from states to actions, which maximizes the expected long-term cumulative reward.

4.1 Q – Learning

Q-Learning (Watkins & Dayan, 1992) is a RL algorithm used for solving sequential decision problems. It focuses on learning an optimal action-value function, often denoted as $Q(s,\ a)$ which represents the expected cumulative reward an agent can achieve by taking action a in state s and following an optimal policy. Q-Learning is described by the Q-value update equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_{a'} Q\,(s', a') - Q(s_t, a_t) \right] \tag{10}$$

where,

$Q(s_t, a_t)$: The Q-value for state s and action a. It represents the expected cumulative reward.

α : The learning rate that controls the magnitude of the updates. Typically, $\alpha \in [0,1]$

γ : The discount factor (from the SDP).

$\max_a Q\,(s', a')$: The Q-value for the next state s after taking the action that maximizes Q-value.

The Q-value update equation reflects the iterative process of Q-Learning. The agent updates its estimate of the Q-value for a state-action pair based on the observed reward, r_{t+1}, and the maximum expected future reward, $\gamma \max_a Q\,(s', a')$, from the next state. The learning rate, α, controls the step size of these updates, allowing the Q-values to gradually converge to their optimal values. Q-Learning is well-suited for FDP as it can learn to make decisions about well drilling, resource allocation, and infrastructure management to maximize long-term rewards. However, its applicability can be limited by significant memory requirements for storing Q-values when dealing with large state and action spaces.

4.2 Deep Q – Networks

DQN is a value-based reinforcement learning algorithm that leverages deep learning to approximate the optimal action-value function $Q^*(s, a)$, enabling the application of value-based approaches to learning optimal policies in complex, high-dimensional environments characterized by a large number of state-action pairs. To deal with the instability associated with function approximation in Q-Learning, DQN incorporates two key components: the *replay buffer* and the *target network*.

Replay Buffer: The replay buffer stores experiences as transitions $(s_t, a_t, r_{t+1}, s_{t+1})$. During training, mini-batches of these transitions are randomly sampled, breaking the correlation between consecutive experiences. This random sampling stabilizes training by reducing the variance of updates and preventing overfitting to recent experiences.

Target Network: DQN uses two networks: the *online network* and the *target network*. The online network, which is responsible for learning and updating the Q-values during training, is updated at each step to improve the action-value estimates. In contrast, the target network, which provides stable target Q-values for the learning process by holding a fixed copy of the online network's weights for a set number of iterations, is updated less frequently. This mechanism provides more stability by preventing the action-value estimates from oscillating or diverging.

The loss function used to train the DQN is derived from the Bellman equation and is defined as:

$$\mathrm{L}(\theta) = E\left[\left(r_{(t+1)} + \gamma \max_{a'} Q\left(s_{(t+1)}, a'; \theta^{-}\right) - Q(s_t, a_t; \theta)\right)^2\right] \quad (11)$$

where γ is the discount factor, θ are the parameters of the online network, and θ^{-} are the parameters of the target network. The online network updates its parameters θ to minimize this loss function. The DQN process, as illustrated in Figure 1, involves the following steps:

1. The agent observes the current state of the environment.
2. The *online network* predicts the Q-values for all possible actions, selecting the action with the highest value or, with probability ε, choosing an exploratory action.
3. After the action is executed, the resulting reward and the next state are observed, and the transition $(s_t, a_t, r_{t+1}, s_{t+1})$ is stored in the replay buffer.
4. A mini-batch of transitions is sampled from the replay buffer to update the *online network* based on the loss function.
5. Periodically, the weights of the *target network* are updated to match the *online network*.

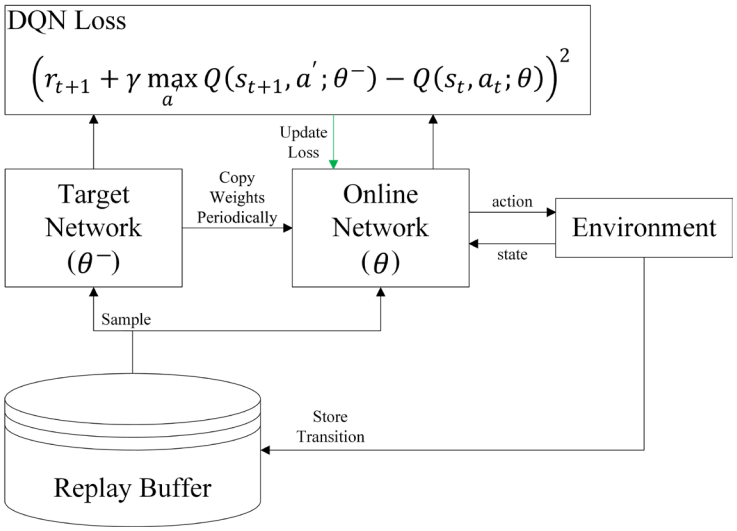

Figure 1. DQN dataflow diagram

4.2 Soft-Updating weights

In DQN, the *target network* is updated by hardly copying the weights from the *online network* at regular intervals during training. In contrast, a soft-update mechanism updates the target network gradually. The soft-update blends the weights of the *target network* with the *online network* at each step, instead of performing a hard copy of weights. This gradual update is defined as:

$$\theta^{-} \leftarrow \tau\theta + (1 - \tau)\theta^{-} \quad (12)$$

where τ is the soft-update factor, which determines how much of the online network's weights are blended into the target network. In the present work, τ is set equal to 0.001, which is the value usually employed to keeps a satisfactory balance between updating the target network and maintaining stability during the learning process (P. Zhang et al., 2023).

This soft-update mechanism provides a more stable learning process by ensuring that the target network changes slowly over time, reducing the risk of large updates that could make the learning process unstable and ensuring

smoother training dynamics. As a result, the target values used for the Q-learning update are more consistent, leading to better convergence.

5. Experimental Setup

A synthetic case study is considered to reduce the large computational effort required if a full-scale dynamic simulation of the subsurface is used. Despite the simulated synthetic case is not physics-based, the case study allows investigating the capability of the proposed method of solving an optimization problem characterized by a number of possible solutions similar to that of drilling scheduling in a FDP project.

5.1 Simulation of the Environment

Unlike real reservoirs, where pressure propagation follows complex dynamics governed by geological properties, this work assumes that the pressure at any given reservoir location is the sum of contributions from the surrounding wells, each one inversely proportional to the distance between the well and the location. Specifically, the oil production of a well is modeled considering two effects: natural depletion and pressure influence of the surrounding wells. The oil production of well i during time-step t is:

$$o_{t,i} = \left(on_{t,i} + p_{t,i}\,\psi\right) h_{t,i} \tag{13}$$

where $on_{t,i}$ is the oil production of well i during time-step t due to natural depletion, ψ is a geological parameter which represents the effect of the BHP on oil production and $h_{t,i}$ is a binary variable equal to 1 if the well i is a producer and equal to 0 otherwise.

The quantity $on_{t,i}$, representing the oil production by natural depletion of well i in time-step t, is assumed to decrease with constant rate ρ:

$$on_{t+1,i} = on_{t,i}\,\rho \tag{14}$$

where $on_{0,i}$ is a fixed value representing the initial production of well i and $\rho \in (0,1)$ is a depletion coefficient.

The contribution of a well to the pressure in the i^{th} location of the reservoir, $p_{t,i}$, is assumed to be inversely proportional to the distance between the well and the location. Therefore, the closer two producing wells are, the smaller the oil production rate, and, on the contrary, the closer an injection well is located to a production well, the larger the production. To account for uncertainties in subsurface geological properties, a matrix $\phi = [\phi_{i,j}]$, $i,j = 1,\dots,N_w$, is introduced. The generic element $\phi_{i,j}$ of ϕ represents the uncertainty in the interaction between wells i and j. N_r samples of the matrix, ϕ_r, are simulated by randomly sampling each element $\phi_{i,j}^r$ from a Gaussian distribution with mean μ and standard deviation σ. Then, the pressure in the location of well i during time-step t is:

$$p_{t,i} = \sum_{j=1}^{N_w} \frac{pm_{t,j}}{distance(i,j)\;\phi_{i,j}^r}, \quad i \neq j \tag{15}$$

where, $pm_{t,j}$ is a prefixed coefficient that represents the maximum possible pressure contribution of the j^{th} well at time-step t on the other wells.

Other assumptions about the environment are:

- The cost of injecting water is considered negligible ($v_w = 0$).
- Drilling of the next well starts immediately after the end of the drilling of the previous well, i.e., without idle time between completing one well and starting the drilling of the next well.

- Operational well rate capacity and pressure limits are preset.
- The number of available well slots per platform, N_w, is equal to 20.

5.2 Experiment Definition

Experiment A

A soft-update mechanism is implemented in the DRL agent algorithm with $\tau = 0.001$. Table 3 describes the available locations for drilling in Cartesian coordinates. The maximum pressure contribution to other wells, $pm_{t,i}$, is defined according to Eq. 16 and the geological parameter ψ is set equal to 1 $\left[\frac{Oil\ production\ unit}{Pressure\ unit}\right]$. The price of oil, v_o, has been set equal to 1 monetary unit (mu) per oil produced unit and the depletion coefficient ρ has been set equal to 0.85. The parameters for the DQN configuration are reported in Table 4. The parameters (μ, σ) defining the gaussian probability distribution of $\phi_{i,j}$ are reported in Table 5. At the beginning of each episode, one of the $N_r = 50$ realizations, ϕ^r, of the matrix $\phi = [\phi_{i,j}]$, $i, j = 1, \dots, N_w$ is randomly sampled. The proposed DRL agent with the soft-update implementation is compared with the DQN agent configuration proposed by (Paola et al., 2020). The agent is trained 10 times, where each training consists of 1000 episodes.

Well Location ID	Location	Initial Capacity[1] $[on_{0,i}]$
A	(1, 2, 5)	0.14
B	(5, 0, 5)	0.04
C	(5, 2, 5)	0.2
D	(3, 2, 5)	0.06
E	(4, 1, 5)	0.07
F	(1, 0, 5)	0.19
G	(0, 5, 5)	0.06
H	(2, 4, 5)	0.02
I	(5, 5, 5)	0.1
J	(4, 0, 5)	0.19
K	(5, 1, 5)	0.09
L	(2, 2, 5)	0.2
M	(1, 1, 5)	0.09
N	(4, 3, 5)	0.15
O	(0, 4, 5)	0.09
P	(2, 3, 5)	0.1
Q	(1, 5, 5)	0.17
R	(3, 5, 5)	0.18
S	(3, 1, 5)	0.11
T	(3, 0, 5)	0.16

Table 3. Experiment Parameters

$$pm_{t,i} = \begin{cases} -0.06, & Type_{t,i} = 1 \\ +0.03, & Type_{t,i} = -1 \end{cases} \tag{16}$$

γ	0.9
d	0.3
ε	$e^{-0.04t}, \quad t \in T$

[1] Relative volume per time unit

Table 4. DQN Hyper-Parameters

N_r	50
μ	1
σ	0.1

Table 5. Realization parameters

Experiment B

The objective of the experiment is to verify the effect of using different numbers of realizations for training DQN on the performance of the identified policy. To this aim, a set of $N_r = 500$ realizations, ϕ^r, of the matrix $\phi = [\phi_{i,j}]$, $i, j = 1, \dots, N_w$, has been generated and split into train (80%) and test (20%) sets. Two different scenarios have been considered, where the number of realizations used for training is 1 and 400, respectively. Each scenario is then evaluated on the test set. For each scenario, the experiment has been conducted 10 times in order to consider the variability of the learning process. The parameters of the environment and hyper-parameters of the agent have been set as in the previous experiment.

6. Results

Experiment A

The agent with the soft-update mechanism is compared to the agent used in (Paola et al., 2020) without soft-updating. Figure 2 shows that the proposed soft-update mechanism enables the agent to achieve a larger reward with a smaller standard deviation. This indicates that the soft-update mechanism allows reducing instability during training, leading to a smoother learning process which enables the agent to identify a policy characterized by more satisfactory performance and smaller variability. Utilizing a high-performance computing node equipped with two Intel® Xeon® Gold 6252 CPUs (24 cores per socket, 2.1 GHz base frequency), totaling 48 cores, 190 GiB of system memory, and an NVIDIA Quadro P5000 GPU (16,278 MiB memory, CUDA 11.2) the average computational time is increased by 8% (from 37s to 40s) when DQN with soft-updates is used instead of standard DQN. This increase is attributed to the more frequent NN updates required by the soft-update strategy.

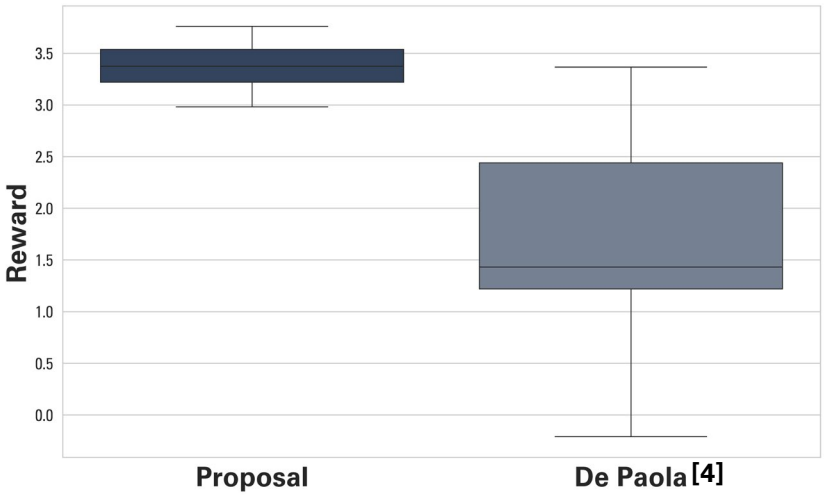

Figure 2. Experiment A: Box-plot of the reward.

The analysis of the selected drilling strategies shows that they prioritize the early drilling of the most productive wells, and then, strategically place injectors wells in the central portion of the reservoir to maximize the pressure in the reservoir.

Experiment B

Figure 3 shows that increasing the number of available realizations from 1 to 400 allows to significantly increase the average reward in the test set.

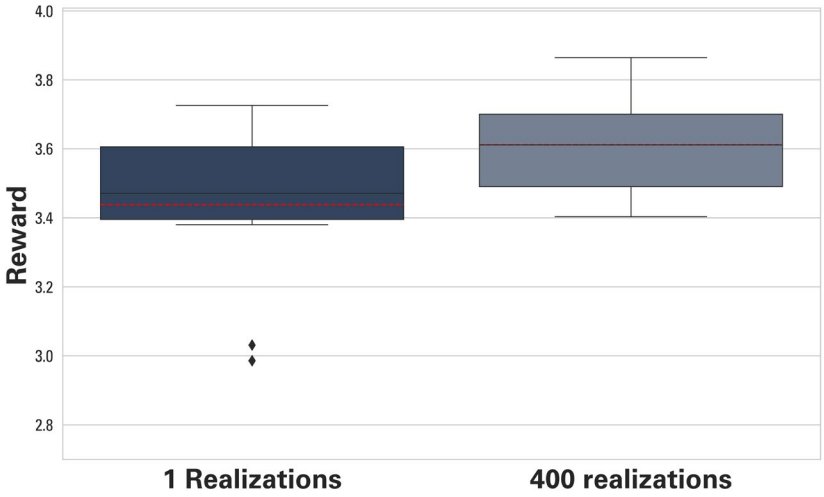

Figure 3. Experiment B: Box-plot of the reward considering 10 runs of the experiment.

Figure 4 shows the distribution of the rewards over the 100 realizations of the test set in one out of the ten repetitions of the experiment. It can be noticed that training with several realizations enables the agent to achieve rewards larger on average (red line) and with lower variability, confirming that increasing the number of realizations used for training leads to a more robust policy.

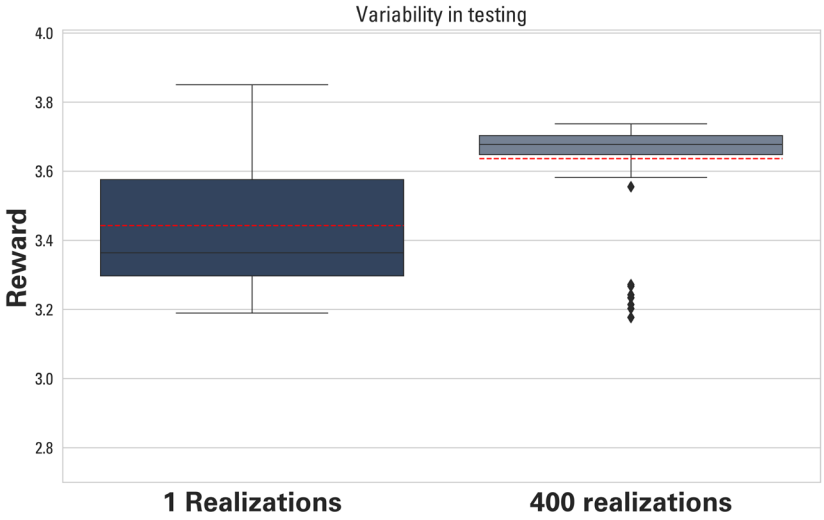

Figure 4. Experiment B: Box-plot of the reward in a single run of the experiment.

7. Conclusions

We have developed a method based on DQN for the optimization of the drilling scheduling of a FDP project. It employs a soft-updating mechanism to reduce training volatility and improve long-term rewards. The comparison of the proposed method with a traditional DQN on a synthetic case study that emulates the behavior of a reservoir has shown that the soft-updating mechanism allows obtaining larger rewards with smaller variability to the uncertainty on the reservoir geological properties. Also, the capacity of DQN to learn and adapt the scheduling strategy according to the specific geological properties of the reservoir has been demonstrated.

Considering the promising achieved results, future research will focus on using a GPU-enhanced reservoir simulator for the simulation of the environment and applying it to the Olympus challenge (Fonseca et al., 2020).

References

Barros, E. G. D., Chitu, A., & Leeuwenburgh, O. (2020). Ensemble-based well trajectory and drilling schedule optimization—application to the Olympus benchmark model. *Computational Geosciences*, *24*(6), 2095–2109. https://doi.org/10.1007/S10596-020-09952-7/METRICS

Bellman, R. (1957). *Dynamic Programming* (First). Princeton University Press. https://press.princeton.edu/books/paperback/9780691146683/dynamic-programming

Bergey, P. (2020). Generative well pattern design—principles, implementation, and test on OLYMPUS challenge field development problem. *Computational Geosciences*, *24*(6), 2079–2094. https://doi.org/10.1007/S10596-019-09912-W/METRICS

Cárdenas Pantoja, N., Abdin, A., Baraldi, P., Pinciroli, L., & Zio, E. (2024). Deep Reinforcement Learning for Strategic Asset Management in Oil & Gas Recovery Projects. *2024 8th International Conference on System Reliability and Safety, ICSRS 2024*.

Carta, S., Ferreira, A., Podda, A. S., Reforgiato Recupero, D., & Sanna, A. (2021). Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting. *Expert Systems with Applications*, *164*, 113820. https://doi.org/10.1016/J.ESWA.2020.113820

Chang, Y., Lorentzen, R. J., Nævdal, G., & Feng, T. (2020). OLYMPUS optimization under geological uncertainty. *Computational Geosciences*, *24*(6), 2027–2042. https://doi.org/10.1007/S10596-019-09892-X/METRICS

Ferguson, T. S. (1967). Sequential Decision Problems. In *Mathematical Statistics* (pp. 309–387). Elsevier. https://doi.org/10.1016/b978-1-4831-8253-7.50011-x

Fonseca, R. M., Rossa, E. Della, Emerick, A. A., Hanea, R. G., & Jansen, J. D. (2020). Introduction to the special issue: Overview of OLYMPUS Optimization Benchmark Challenge. *Computational Geosciences*, *24*(6), 1933–1941. https://doi.org/10.1007/S10596-020-10003-4/METRICS

Halat, S., & Ebadzadeh, M. M. (2021). *Modified Double DQN: addressing stability*. https://arxiv.org/abs/2108.04115v1

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

He, J., Tang, M., Hu, C., Tanaka, S., Wang, K., Wen, X.-H., & Nasir, Y. (2022). Deep Reinforcement Learning for Generalizable Field Development Optimization. *SPE Journal*, *27*(01), 226–245. https://doi.org/10.2118/203951-PA

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., & Perez, P. (2022). Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, *23*(6), 4909–4926. https://doi.org/10.1109/TITS.2021.3054625

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*. https://arxiv.org/abs/1509.02971v6

Mirzaei-Paiaman, A., Santos, S. M. G., & Schiozer, D. J. (2022). Iterative sequential robust optimization of quantity and location of wells in field development under subsurface, operational and economic uncertainty. *Journal of Petroleum Science and Engineering*, *218*, 111005. https://doi.org/10.1016/J.PETROL.2022.111005

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*. https://arxiv.org/abs/1312.5602v1

Muther, T., Qureshi, H. A., Syed, F. I., Aziz, H., Siyal, A., Dahaghi, A. K., & Negahban, S. (2022). Unconventional hydrocarbon resources: geological statistics, petrophysical characterization, and field development strategies. *Journal of Petroleum Exploration and Production Technology*, *12*(6), 1463–1488. https://doi.org/10.1007/s13202-021-01404-x

Nasir, Y. (2020). *Deep Reinforcement Learning for Field Development Optimization*. https://arxiv.org/abs/2008.12627v1

Nasir, Y., He, J., Hu, C., Tanaka, S., Wang, K., & Wen, X. H. (2021). Deep Reinforcement Learning for Constrained Field Development Optimization in Subsurface Two-phase Flow. *Frontiers in Applied Mathematics and Statistics*, *7*, 689934. https://doi.org/10.3389/FAMS.2021.689934/BIBTEX

OpenAI, :, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H. P. d. O., Raiman, J., … Zhang, S. (2019). *Dota 2 with Large Scale Deep Reinforcement Learning*. https://doi.org/https://doi.org/10.48550/arXiv.1912.06680

Paola, G. De, Ibanez-Llano, C., Rios, J., & Kollias, G. (2020, October 19). Reinforcement Learning for Field Development Policy Optimization. *Proceedings - SPE Annual Technical Conference and Exhibition*. https://doi.org/10.2118/201254-MS

Schulze-Riegert, R., Nwakile, M., Skripkin, S., Whymark, M., Baffoe, J., Geissenhoener, D., Anton, A., Meulengracht, C. S., & Ng, K. J. (2020). Olympus challenge—standardized workflow design for field development plan optimization under uncertainty. *Computational Geosciences*, *24*(6), 2059–2077. https://doi.org/10.1007/S10596-019-09905-9/METRICS

Silva, V. L. S., Cardoso, M. A., Oliveira, D. F. B., & de Moraes, R. J. (2020). Stochastic optimization strategies applied to the

OLYMPUS benchmark. *Computational Geosciences*, *24*(6), 1943–1958. https://doi.org/10.1007/S10596-019-09854-3/METRICS

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). The MIT Press. https://mitpress.mit.edu/9780262039246/reinforcement-learning/

Wang, L., & Oliver, D. S. (2019). Efficient Optimization of Well-Drilling Sequence with Learned Heuristics. *SPE Journal*, *24*(05), 2111–2134. https://doi.org/10.2118/195640-PA

Wang, L., & Oliver, D. S. (2021). Fast robust optimization using bias correction applied to the mean model. *Computational Geosciences*, *25*(1), 475–501. https://doi.org/10.1007/s10596-020-10017-y

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning 1992 8:3*, *8*(3), 279–292. https://doi.org/10.1007/BF00992698

Yu, K., Jin, K., & Deng, X. (2022). Review of Deep Reinforcement Learning. *IMCEC 2022 - IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference*, 41–48. https://doi.org/10.1109/IMCEC55388.2022.10020015

Zhang, P., Zhang, J., & Kan, J. (2023). A Research on Manipulator-Path Tracking Based on Deep Reinforcement Learning. *Applied Sciences 2023, Vol. 13, Page 7867*, *13*(13), 7867. https://doi.org/10.3390/APP13137867

Zhang, Q., Lin, M., Yang, L. T., Chen, Z., & Li, P. (2019). Energy-Efficient Scheduling for Real-Time Systems Based on Deep Q-Learning Model. *IEEE Transactions on Sustainable Computing*, *4*(1), 132–141. https://doi.org/10.1109/TSUSC.2017.2743704