Large-Scale Wildfire Mitigation Through Deep Reinforcement Learning

Altamimi, Abdulelah; Lagoa, Constantino; Borges, José G.; McDill, Marc E.; Andriotis, C. P.; Papakonstantinou, K. G.

DOI
10.3389/ffgc.2022.734330

Publication date
2022

Document Version
Final published version

Published in
Frontiers in Forests and Global Change

frontiers | Frontiers in Forests and Global Change

ORIGINAL RESEARCH
published: 05 July 2022
doi: 10.3389/ffgc.2022.734330

Large-Scale Wildfire Mitigation Through Deep Reinforcement Learning

Abdulelah Altamimi[1]*, Constantino Lagoa[1], José G. Borges[2], Marc E. McDill[3], C. P. Andriotis[4] and K. G. Papakonstantinou[5]

[1] School of Electrical Engineering and Computer Science, The Pennsylvania State University, University Park, PA, United States, [2] Forest Research Centre, School of Agriculture, University of Lisbon, Lisbon, Portugal, [3] Department of Ecosystem Science and Management, The Pennsylvania State University, University Park, PA, United States, [4] Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, Netherlands, [5] Department of Civil and Environmental Engineering, The Pennsylvania State University, University Park, PA, United States

OPEN ACCESS

Edited by:
*Xianzhi Wang,
University of Technology Sydney, Australia*

Reviewed by:
*Ruotong Hu,
Ocean University of China, China
Chao Ma,
Harbin Engineering University, China*

***Correspondence:**
*Abdulelah Altamimi
ala230@psu.edu*

Specialty section:
This article was submitted to Forest Management, a section of the journal Frontiers in Forests and Global Change

Received: *01 July 2021*
Accepted: *06 June 2022*
Published: *05 July 2022*

Citation:
Altamimi A, Lagoa C, Borges JG, McDill ME, Andriotis CP and Papakonstantinou KG (2022) Large-Scale Wildfire Mitigation Through Deep Reinforcement Learning. Front. For. Glob. Change 5:734330. doi: 10.3389/ffgc.2022.734330

Forest management can be seen as a sequential decision-making problem to determine an optimal scheduling policy, e.g., harvest, thinning, or do-nothing, that can mitigate the risks of wildfire. Markov Decision Processes (MDPs) offer an efficient mathematical framework for optimizing forest management policies. However, computing optimal MDP solutions is computationally challenging for large-scale forests due to the curse of dimensionality, as the total number of forest states grows exponentially with the numbers of stands into which it is discretized. In this work, we propose a Deep Reinforcement Learning (DRL) approach to improve forest management plans that track the forest dynamics in a large area. The approach emphasizes on prevention and mitigation of wildfire risks by determining highly efficient management policies. A large-scale forest model is designed using a spatial MDP that divides the square-matrix forest into equal stands. The model considers the probability of wildfire dependent on the forest timber volume, the flammability, and the directional distribution of the wind using data that reflects the inventory of a typical eucalypt (Eucalyptus globulus Labill) plantation in Portugal. In this spatial MDP, the agent (decision-maker) takes an action at one stand at each step. We use an off-policy actor-critic with experience replay reinforcement learning approach to approximate the MDP optimal policy. In three different case studies, the approach shows good scalability for providing large-scale forest management plans. The results of the expected return value and the computed DRL policy are found identical to the exact optimum MDP solution, when this exact solution is available, i.e., for low dimensional models. DRL is also found to outperform a genetic algorithm (GA) solutions which were used as benchmarks for large-scale model policy.

Keywords: forest management, wildfire mitigation, Markov Decision Process, dynamic programming, deep reinforcement learning

1. INTRODUCTION

Wildfires threaten and kill people, destroy urban and rural property, degrade air quality, ravage forest ecosystems, and contribute to global warming. For example, in California, an enormous increase in wildfire activity has occurred in the last few years causing 150 fatalities, 25,000 destroyed properties, and a total loss of 50 million dollars in the 2017–2018 period alone (Williams et al., 2019). High temperatures and unusual droughts in some regions are the main drivers that accelerate these wildfires (Goss et al., 2020). Many other parts of the planet, such as the Mediterranean region in Europe, suffer from the risks and damages of the massive increase in wildfires, partly due to climate change (Faivre et al., 2018; Molina et al., 2019). For example, in 2017 alone, wildfires in Portugal claimed over 110 lives and destroyed property worth over 1,000 million euros. This context highlights the need for the development of a new wildfire management paradigm (Castellnou et al., 2019; Moreira et al., 2020). It further places a challenge to forest researchers and managers as it calls for methods and tools that may help integrate forest and fire management planning activities, currently still being carried out mostly independently of each other.

The development of wildfire occurrence probability models (Marques et al., 2012) and wildfire damage models (e.g., Botequim et al., 2017) was influential in further research efforts to address wildfire events in both stand (e.g., González et al., 2006; Ferreira et al., 2014), and landscape level management planning methods (e.g., Wei et al., 2008; González-Olabarria and Pukkala, 2011; Wei, 2012; Ferreira et al., 2015; Marques et al., 2017), within decision support systems (e.g., Pacheco et al., 2015). Yet these methods have focused either on a stand-level spatial scale or, else, in a decomposition of the landscape-level problem that acknowledges neighborhood relations and the potential of wildfire spread between sets of neighbors (e.g., Ferreira et al., 2015; Marques et al., 2017) do not consider wildfire behavior and spread over the whole landscape. Recent studies (e.g., Botequim et al., 2017), highlighted that spatially, explicit fire simulators, such as FARSITE (Finney, 2004) and FlamMap (Finney, 2006), have been used extensively both for research and for practical purposes (Alexander and Cruz, 2013). These simulators integrate biometric and environmental, e.g., topographic and weather, data to estimate fire behavior characteristics, such as fireline intensity and rate of spread (Finney and Andrews, 1999). Nevertheless, the use of wildfire behavior simulators requires dynamic weather information and accurate spatial estimation of fuel characteristics, which are difficult to predict over time and space (He and Mladenoff, 1999), thus complicating their application within a management planning framework.

The work presented in this paper addresses the computational shortcomings of approaches that rely on multiple wildfire behavior simulations. It targets the development of an approach that combines Markov Decision Process (MDP) modeling through a Deep Reinforcement Learning (DRL) framework in order to encapsulate ignition probability, wildfire occurrence, and damage models, e.g., stand flammability and wildfire spread parameters, to address wildfire risk over the whole landscape. Unlike classical high-computational techniques, DRL has the ability to compute the optimal solution of large-scale MDPs. Dynamic programming (DP) has been previously applied to address several forest management planning problems (e.g., Hoganson et al., 2008). Ferreira et al. (2014) further introduced a stochastic dynamic programming approach to address wildfire risk, where stand-level management planning was modeled as an MDP, with the basal area and shrub age as the state space parameters, and cleaning the shrub (fuel treatment) and thinning as the available actions. DRL has been also used in forest management to design a wildfire dynamic model using satellite images (Ganapathi Subramanian and Crowley, 2018). The agent observes the wildfire from images and predicts the propagation direction. This approach does not provide however any decisions that can help prevent the propagation of wildfire. Another DRL approach was introduced in Haksar and Schwager (2018) that aims to autonomously fight forest wildfire *via* unmanned aerial vehicles.

This research builds from this experience to solve a 1-period large-scale forest-level model using a DRL algorithm. We start with an MDP that provides a probabilistic model for the propagation of fire in the landscape of interest. To derive such a model, the area of interest is divided into stands and, for each of them, the possible actions are harvest or do nothing. In this model, given the wind direction, the probability of wildfire occurrence at each stand is estimated, as a function of the probability of ignition in the stand and the probability of wildfire propagation from adjacent stands, leading to a model of fire propagation for the whole landscape. Three case studies are provided in this paper to validate the algorithm. Comparing to the backward induction algorithm for the exact MDP solution, the proposed solution is shown to have satisfactory results in small models. The results of large-scale problems outperform GA benchmarks in both solution quality and computational time.

2. MATERIALS AND METHODS

2.1. Background

This section presents a brief description of MDPs, as well as different solution algorithms and their limitations.

2.1.1. Markov Decision Process

A Markov Decision Process (MDP) is a mathematical model for sequential decision-making problems (Papakonstantinou and Shinozuka, 2014; Puterman, 2014). The decision maker (the agent) interacts with the environment to accomplish a goal. The MDP is defined by the 4-tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is Markovian transition probability, and $\mathcal{R}$ the set of all possible rewards. At each step m, the current state $s_m \in \mathcal{S}_m$ is observed, then the agent selects an action $a_m \in \mathcal{A}$, and receives a numerical reward $r \in \mathcal{R} \subset \mathbb{R}$. Following the conditional probability distribution $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ the system transitions to next state s_{m+1}. The conditional probability $P(s_{m+1}|s_m, a_m)$ satisfies the so-called Markov property; i.e., the next state relies only on current state and action, regardless of all prior history of states and decisions. Thus, the current state includes all information that matters.

The goal is to maximize the expected total reward starting at any given step m and going forward into the future. The expected reward r to be received, when the system is at state s_m and action a_m is taken, is the function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. The total reward R_m is the summation of a sequence of rewards starting at step m until the end of the horizon M:

$$\begin{aligned} R_m &= r(s_m, a_m) + r(s_{m+1}, a_{m+1}) + ... + r(s_M, a_M) \\ &= \sum_{i=0}^{M-m} r(s_{m+i}, a_{m+i}) \end{aligned} \tag{1}$$

Equation (2) below shows a recursive relationship between rewards:

$$\begin{aligned} R_m &= r(s_m, a_m) + [r(s_{m+1}, a_{m+1}) + r(s_{m+2}, a_{m+2}) \\ &+ r(s_{m+3}, a_{m+3}) + ...] \\ &= r(s_m, a_m) + R_{m+1} \end{aligned} \tag{2}$$

The policy π is a function that assigns the sequence of actions to be chosen at each step m. It can be either deterministic $\pi(s_m) : \mathcal{S} \rightarrow \mathcal{A}$, where $\pi(s_m)$ is a single action, or stochastic $\pi(a_m|s_m) : \mathcal{S} \rightarrow P(\mathcal{A})$ where $\pi(a_m|s_m)$ is a conditional probability distribution over actions given the state. In other words, the deterministic policy can be also interpreted as a stochastic policy with a probability of 1. Following the policy π, the expected total reward at state s_m is called the value function $v^{\pi}(s_m)$:

$$v^{\pi}(s_m) = \mathbb{E}_{\pi}[R_m^{\pi}|s_m] \tag{3}$$

where R_m^{π} is the total reward starting at m under the policy π. The expected return of the policy π at state s_m and taking action a_m is the action-value function $Q^{\pi}(s_m, a_m)$ (or simply the Q-function):

$$Q^{\pi}(s_m, a_m) = \mathbb{E}_{\pi}[R_m^{\pi}|s_m, a_m] \tag{4}$$

Both the value function and the Q-function satisfy recursive relationships. Using (2) in (3), for any policy π, the value function $v^{\pi}(s_m)$ can be written as:

$$v^{\pi}(s_m) = \mathbb{E}_{\pi}[R_m^{\pi}|s_m] = r(s_m, a_m) + \mathbb{E}_{\pi}[R_{m+1}^{\pi}|s_{m+1}] \tag{5}$$

Similarly for Q-function, for any policy π:

$$\begin{aligned} Q^{\pi}(s_m, a_m) &= \mathbb{E}_{\pi}[R_m^{\pi}|s_m, a_m] \\ &= r(s_m, a_m) + \mathbb{E}_{\pi}[R_{m+1}^{\pi}|s_{m+1}, a_{m+1}] \end{aligned} \tag{6}$$

Solving the MDP means finding the policy that maximizes the value function. Thus, for each MDP, there is at least one policy that is better than or equal to all other policies (Sutton and Barto, 2018). This policy is the optimal policy π^* that leads to the optimal value functions $v^*(s_m)$ and optimal Q-function $Q^*(s_m, a_m)$. These optimal functions satisfy the recursive relation in (5) and (6) that define the Bellman equation:

$$\begin{aligned} v^*(s_m) &= \max_{a_m \in \mathcal{A}} Q^*(s_m, a_m) \\ &= \max_{a_m \in \mathcal{A}} \left\{ r(s_m, a_m) + \mathbb{E}_{\pi^*}[R_{m+1}^{\pi^*}|s_{m+1}, a_{m+1}] \right\} \\ &= \max_{a_m \in \mathcal{A}} \left\{ r(s_m, a_m) + \mathbb{E}_{\pi^*}[v^*(s_{m+1})] \right\} \\ &= \max_{a_m \in \mathcal{A}} \left\{ r(s_m, a_m) + \sum_{s_{m+1} \in \mathcal{S}} P(s_{m+1}|s_m, a_m) v^*(s_{m+1}) \right\} \end{aligned} \tag{7}$$

2.1.2. Dynamic Programming: Curse of Dimensionality

Dynamic programming is a technique for finding an optimal MDP solution (Puterman, 2014). To find the exact MDP solution with a dynamic programming algorithm, such as value iteration with backward induction, the value of each state is computed and stored first to identify the optimal actions at the last step. This procedure is then repeated at each step in a backward manner until the first step. More details about value iteration and backward induction are provided in Section 2.3.1. Thus, exact dynamic programming solutions require full knowledge of the model and although powerful and convenient for low dimensional MDPs, they are practically inefficient in large-scale MDP settings where the computations grow exponentially with the numbers of variables involved, a phenomenon also referred to as the curse of dimensionality. Hence, several solution techniques are developed to approximate the MDP value function and policy, such as Deep Reinforcement Learning (DRL) (Sutton and Barto, 2018).

2.1.3. Reinforcement Learning

Reinforcement Learning (RL) is an efficient solution for dynamic decision-making problems. Without explicitly knowing/needing the model of the environment and to overcome the curse of dimensionality, the RL agent learns through direct interaction with the environment and updates the Q-functions accordingly. It is thus a model-free technique since full knowledge of the model is not required, as the agent learns from the observations from the environment. Temporal-Difference (TD) learning is central approach in RL that uses the experience of the agent-environment interaction to update the Q-functions. RL algorithms belong to two families of approaches called on-policy and off-policy, respectively. On-policy algorithms evaluate the same policy the agent is learning or exploring, while off-policy algorithms evaluate a target policy different than the behavior policy of the agent. In off-policy algorithms, the agent evaluates a target policy from the data collected by the behavior policy (Sutton and Barto, 2018). Separating these two policies helps the agent explore the environment during its learning as a parallel task. Moreover, updating the target policy by sampling the data experienced by the behavior policy in off-policy training is more sample-efficient compared to the on-policy method (Wang et al., 2016). These advantages make off-policy algorithms attractive and suitable for many fields (Degris et al., 2012). Hence, only off-policy algorithms are considered in this paper.

For state s and action a at step m with reward $r_m(s_m, a_m)$, the Q-learning algorithm (Watkins and Dayan, 1992), one of the basic TD approaches, learns, and updates $Q(s_m, a_m)$ as follows:

$$\begin{aligned}&Q(s_m, a_m) \leftarrow Q(s_m, a_m)\\ &+\eta\left(r(s_m, a_m) + \max_{a_{m+1}\in\mathcal{A}} Q(s_{m+1}, a_{m+1}) - Q(s_m, a_m)\right)\end{aligned} \tag{8}$$

where η is the learning rate. The TD target Y_m is:

$$Y_m = r(s_m, a_m) + \max_{a_{m+1}\in\mathcal{A}} Q(s_{m+1}, a_{m+1}) \tag{9}$$

and the difference between Y_m and $Q(s_m, a_m)$ is known as the TD error δ_m:

$$\delta_m = Y_m - Q(s_m, a_m) \tag{10}$$

The Q-functions for all state-action pairs are learned and stored in tabular form that depends on the cardinality of the state and action spaces. However, this leads to a difficult and inefficient learning process in the case of large-scale spaces and complex environments. The Q-functions approximation, therefore, is being used to overcome this high-dimensional challenge. The action-value function is approximated using a parameter vector $\boldsymbol{\theta}$ with a relatively small dimension compared to the cardinality of the state and action spaces, i.e., $|\boldsymbol{\theta}| \ll |\mathcal{S} \times \mathcal{A}|$. Specifically, the Q-function is approximated as follows:

$$Q(s_m, a_m) \approx Q(s_m, a_m|\boldsymbol{\theta}) \tag{11}$$

where $\boldsymbol{\theta}$ is the weight vector that is updated to minimize the TD error. Several function approximation processes can be used, such as a linear combination of features and neural networks.

One notable algorithm of approximating the action-value function through a neural network was introduced in Mnih et al. (2013), where the Deep Q-Network (DQN) algorithm was developed. The key idea of DQN is to approximate the Q-function, where the state vector is the input of the neural network and the action-value functions are the outputs. The number of hidden layers expresses how deep the network is. A memory buffer is used to store rewards and past transitions generated by the behavior policy as tuple of current state, action, and next state, (s_m, a_m, r_m, s_{m+1}), known as experience replay. At each training step, the agent replays a sample batch from the experience replay to update the neural network parameters/weights $\boldsymbol{\theta}$.

The state usually transitions to a state that is dependent on the previous one. Sampling these consecutive states from the memory builds up correlations between samples that slow-downs the learning process. Hence, the sample batch is randomized to break the correlation between transitions and to accelerate the learning process. Instead of sampling uniformly, prioritized sampling from the experience replay based on the TD error can further speed-up the learning progress by choosing transitions that have high error more often than other transitions (Schaul et al., 2015).

The $\boldsymbol{\theta}$ updating to minimize the TD error in (10) is performed using stochastic gradient descent (SGD). Instead of updating the target Y_m at each training step that may cause instability in the learning process, a separate copy from the original network called target network is used to calculate the TD target Y_m (Mnih et al., 2015) as follows:

$$Y_m = r(s_m, a_m) + \max_{a_{m+1}\in\mathcal{A}} Q^{-}(s_{m+1}, a_{m+1}|\boldsymbol{\theta}^{-}) \tag{12}$$

where Q^{-} refers to the target network that is a delayed copy of the original network. The $\boldsymbol{\theta}^{-}$ is the target network's parameter vector which is updated from step to step with an appropriate update interval. However, the DQN algorithm following this strategy keeps selecting a certain action that maximizes the Q-function for a certain state that can lead to a positive bias and, therefore, overestimation. Using both the original network and the target network as in the DDQN algorithm overcomes this issue (Van Hasselt et al., 2016):

$$Y_m = r(s_m, a_m) + \max_{a_{m+1}\in\mathcal{A}} Q^{-}(s_{m+1}, \arg\max Q(s_{m+1}, a_{m+1}|\boldsymbol{\theta})|\boldsymbol{\theta}^{-}) \tag{13}$$

Here one network selects the maximization action, and the other network calculates the Q-function with that action. However, this maximization is computationally expensive which is a major drawback in case of large action spaces (Wang et al., 2016).

DRL approaches can also approximate the policy $\pi(a_m|s_m) \approx \pi_{\boldsymbol{\theta}}(a_m|s_m)$ by computing the policy gradient based on $\nabla_{\theta^\pi} \log \pi_{\boldsymbol{\theta}}(a_m|s_m)$ (Sutton et al., 2000). The policy is thus gradually improved to maximize the expected reward by updating parameters $\boldsymbol{\theta}^{\pi}$ through the policy gradient:

$$\mathbf{g}_{\boldsymbol{\theta}^{\pi}} = \mathbb{E}_{s_m, a_m}\left[\sum_{m=0} \nabla_{\boldsymbol{\theta}^{\pi}} \log \pi_{\boldsymbol{\theta}}\,(a_m|s_m)\, Q^{\pi_{\boldsymbol{\theta}}}(s_m, a_m)\right] \tag{14}$$

The $Q^{\pi_\theta}(s_m, a_m)$ in Equation (14) can be also replaced by several functions such as the value function $v^{\pi_\theta}(s_m)$ and the advantage function $A^{\pi_\theta}(s_m, a_m)$ (Schulman et al., 2015) :

$$A^{\pi_\theta}(s_m, a_m) = Q^{\pi_\theta}(s_m, a_m) - v^{\pi_\theta}(s_m) \tag{15}$$

where the value function $v^{\pi_\theta}(s_m)$ under the policy π is used as a baseline. The policy gradient in (14) is then computed:

$$\mathbf{g}_{\boldsymbol{\theta}^{\pi}} = \mathbb{E}_{s_m, a_m}\left[\sum_{m=0} \nabla_{\boldsymbol{\theta}^{\pi}} \log \pi_{\boldsymbol{\theta}}\,(a_m|s_m)\, A^{\pi_{\boldsymbol{\theta}}}(s_m, a_m)\right] \tag{16}$$

Equation (16) is supported by two neural networks. The first network provides the policy and is called actor network with parameter $\boldsymbol{\theta}^{\pi}$. The value function $v^{\pi_\theta}(s_m)$ in (15) is approximated by another network called the critic-network with parameters $\boldsymbol{\theta}^{v}$. This approach with two interacting networks is called actor-critic RL. An approximation of the advantage function was proposed in (Mnih et al., 2016):

$$A^{\pi_\theta}(s_m, a_m; \boldsymbol{\theta}^{v}) = \sum_{i=0}^{k-1} r(s_{m+i}, a_{m+i}) + v^{\pi_\theta}(s_{m+k}; \boldsymbol{\theta}^{v}) - v^{\pi_\theta}(s_m; \boldsymbol{\theta}^{v}) \tag{17}$$

where $v^{\pi_\theta}(s_m; \boldsymbol{\theta}^v)$ is the value function approximated by the critic network and k is the number of steps. To compute the gradient in (16) in an efficient way, an importance sampling technique is used (Wang et al., 2016). The importance sampling weights are based on $w_m = \frac{\pi(a_m|s_m)}{\mu(a_m|s_m)}$ where $\pi(a_m|s_m)$ is the target policy that is predicted by the actor network and $\mu(a_m|s_m)$ is the behavior policy that the agent experienced and stored in the memory replay. Equation (16) then becomes:

$$\mathbf{g}_{\boldsymbol{\theta}^\pi} = \mathbb{E}_{s_m \sim \rho, a_m \sim \mu}\left[w_m \nabla_{\boldsymbol{\theta}^\pi} \log \pi_{\boldsymbol{\theta}}(a_m|s_m) A^{\pi_{\boldsymbol{\theta}}}(s_m, a_m; \boldsymbol{\theta}^v)\right] \tag{18}$$

where ρ is the limit distribution of the states related to the behavior policy μ. This gradient is unbiased although it can suffer from high variance due to the unbounded values of the weights w_m. Truncating these weights to prevent these high values rectifies this issue, as proposed in Wawrzyński (2009).

In both these main DRL approaches, i.e., DQNs and deep policy gradient approaches, the output dimension of the neural network is a function of the action space dimension $|A|$. This becomes problematic in the case of multi-component control systems, where each component has its state variable and own available actions, causing the number of possible system actions to grow exponentially. In Andriotis and Papakonstantinou (2019), a Deep Centralized Multi-agent Actor-Critic (DCMAC) approach was developed that treats the actions of each system components as conditionally independent:

$$\pi(\mathbf{a}_m|\mathbf{s}_m) = \prod_{i=1}^{n} \pi_i(a_m^{(i)}|\mathbf{s}_m), \tag{19}$$

where $\mathbf{s}_m = \{s_m^{(i)}\}_{i=1}^{l}$, l is the number of system components, and $\mathbf{a}_m = \{a_m^{(i)}\}_{i=1}^{n}$ with n the number of control units. The number of control units n differs from the number of system components l because two or more components (sub-systems) can share the same set and actions. In DCMAC, all control units (agents) use shared actor-network parameters to approximate their policies, resulting in an output probability mass function over all possible actions for each component. A system critic-network helps to approximate the advantage function A^{π_θ} in a centralized fashion. The output of the actor-network thus now grows linearly instead of exponentially as a function of the number of control units n. DCMAC is shown to have better performance in several sequential decision-making problems related to inspection and maintenance planning of multi-component engineering systems, when compared with state-of-the-art policies (Andriotis and Papakonstantinou, 2019, 2021).

2.2. Fire Propagation Model

This section introduces the forest model structure and the probability of fire occurrence.

2.2.1. Notation

Here we summarize our relevant notation in **Table 1**.

TABLE 1 | List of notation for wildfire occurrence model.

Notation	Description
M	Number of stands in the forest.
N	North direction.
W	West direction.
NW	Northwest direction.
$P_F^{NW}(m)$	The probability of wildfire in stand m with (NW) wind direction.
V[age(m)]	The timber volume based on the age of the stand m.
D	The Do-Nothing action.
H	The Harvest action.

2.2.2. Forest Wildfire Model

The forest wildfire modeling is utilized in computing a policy to mitigate fire risks and maximize timber production during one period. The model consists of M stands arranged as cells in a square matrix with sides equal to $\sqrt{M}$. The model is designed as an MDP to solve a sequential-decision-making problem within a spatially varied environment. The MDP has a finite horizon equal to the number of stands M. The model proceeds from one step to another in a spatial manner, i.e., one stand is treated at each step. This spatial dynamics of the model considers the probability of fire that propagates in the northwest (NW) direction, as an example. However, the model can be easily modified accordingly for any wind direction.

To consider the fire propagating from the NW direction, the stands in the square matrix are sorted in order from left to right starting in the upper-left corner of the matrix. Fire propagates from three adjacent stands in the northwest (NW), north (N), and west (W) directions that form a set of stands called up-wind stands. The probability of fire for each stand mainly depends on actions taken on these up-wind stands. Hence, the fire transition from one stand to another relies on actions taken on the up-wind stands. More details of the fire propagation are provided in Section 2.2.3.

Action a_m applied to each stand m, where $m = 1, 2, ..., M$, is either Harvest (H) or Do-nothing (D). The harvest action aims to remove all the timber volume, while there is no intervention with the do-nothing action that lets the stand timber grow. As the model considers the wildfire propagating from the NW direction, the action a_m affects the states of the stand m and all neighboring stands in the down-wind direction. The state s_m is represented by binary values based on action a_m, i.e., (0) for H and (1) for D. The state also includes relevant action values for related stands that affect the probability of fire at stand $m+1$, and the state s_{m+1} contains values needed at stand $m+2$, etc. Furthermore, any stand and its north stand -if it exists- are located in two consecutive lines in the square matrix. In other words, the stand m has the stand with index $m - \sqrt{M}$ (the upper one) on its N direction, which is also on the NW direction of the stand $m+1$. Thus, the state s_m contains represented values of all actions occurring from stand $m - \sqrt{M}$ to stand m to include all the up-wind set of stands related to stand $m+1$. The state space $\mathcal{S}_m$ thus comprises binary combinations of different possible actions up to $\sqrt{M}+1$ stands, so, the maximum number of states is $2^{\sqrt{M}+1}$. Hence, the number of states at each step grows exponentially with the number of stands M. A simple clarifying example is further illustrated in the next section.

Each stand m is assigned an age that corresponds to the timber volume in that stand. The reward function relies on the inventory volume available in each stand now and one year from now, after a wildfire season. Particularly, the reward $r(s_m, a_m)$ of taking action a in state s of the stand m is a function of the timber volume:

$$r(s_m, a_m) = \begin{cases} V\left[\text{age}(m)\right] + \left[1 - \text{P}_F(m)\right] \times V(1), & \text{if } a_m = H, \forall s_m \in \mathcal{S}_m \\ \left[1 - \text{P}_F(m)\right] \times V\left[\text{age}(m) + 1\right], & \text{if } a_m = D, \forall s_m \in \mathcal{S}_m \end{cases} \tag{20}$$

where $V[\text{age}(m)]$ is the timber volume of the age of stand m, $V(1)$ is volume at age 1, and $\text{P}_F(m)$ is the probability of stand m to get burned from NW fire propagation. After the stand is harvested, its timber volume gets renewed to the minimum volume at age 1. On the other hand, the timber volume grows to the next age class if a do-nothing action is applied. This volume is simulated to reflect the inventory of a eucalypt (Eucalyptus globulus Labill) plantation in Portugal. The reward also relies further on the ignition, stand flammability and wildfire spread parameters that formed the probability of getting burned $\text{P}_F(m)$, as described in the next section. The decision to harvest or do-nothing is based on the same $\text{P}_F(m)$ at each step. If $\text{P}_F(m)$ is higher, then the rewards are generally smaller for both decisions because the expected volume is lower, and vice versa. As the timber volume can be fully traced through the action, only the action is essential in the state variables.

2.2.3. Wildfire Probability

The wildfire probability is designed to include risk of fire spreading from stands in the NW wind direction. The probability of the stand m to get burned is related to the probability of ignition at stand m, $P_{ign}(m)$, or in one of the three upwind stands in the NW direction:

$$\text{P}_F(m) = P_{ign}(m) + P_F(NW) + P_F(N) + P_F(W) \tag{21}$$

where

- $P_{ign}(m) = \text{Ig}(m) \times \text{flam}(m)$, $\text{Ig}(m)$ is ignition probability and $\text{flam}(m)$ is the flammability of stand m.
- $P_F(NW) = P_{NW} \times [1 - P_{ign}(m)]$ is the probability of fire in the NW stand.
- $P_{NW} = \text{Ig}(NW) \times \text{flam}(NW) \times S(NW)$ is the probability of wildfire spread from the NW stand to m, and $S(NW)$ the fire spread probability from NW direction.
- $P_F(N) = P_N \times [1 - P_{ign}(m) - P_F(NW)]$is the probability of fire in the N stand.
- $P_N = \text{Ig}(N) \times \text{flam}(N) \times S(N)$ is the probability of wildfire spread from the N stand to m, and $S(N)$ the fire spread probability from N direction.
- $P_F(W) = P_W \times [1 - P_{ign}(m) - P_F(NW) - P_F(N)]$is the probability of fire in the W stand.
- $P_W = \text{Ig}(W) \times \text{flam}(W) \times S(W)$ is the probability of wildfire spread from the W stand to m, and $S(W)$ the fire spread probability from W direction.

Algorithm 1: Value iteration for solving Finite-Horizon MDP.

```
1 Set: $v(s_M) = 0$
2 for $m = M-1, M-2, ..., 1$ do
3     for each $a \in \mathcal{A}$ do
4         for each $s \in \mathcal{S}$ do
5             $Q(s_m, a_m) = R_m + v^*(s_{m+1})$
6     $\pi^*(s_m) = \text{argmax}_{a \in \mathcal{A}} Q(s_m, a_m)\ \forall a \in \mathcal{A}$
7     $v^*(s_m) = \max_{s \in \mathcal{S}} Q(s_m, a_m)\ \forall s \in \mathcal{S}$
8 Output: $\pi^*(s)$, $v^*(s)$
```

That is, the probability of each stand to get burned depends both on the same stand and actions taken in the previous three upwind stands. This simple model is utilized in this paper, given that the focus is on the suggested DRL solution approach. However, other, more sophisticated, wildfire spread models and quantified uncertainties can also be used here without loss of generality.

A 2×2 forest model matrix is considered as an example to clarify the MDP formulation. It contains $M = 4$ stands organized as shown in **Figure 1A**. For each stand, the up-wind set in the NW direction contains a number of stands ranging from 0 to 3 stands. **Figure 1B** shows the up-wind set for the stand 4. At the beginning, for stand 1, there are two actions, i.e., H and D that could be applied and there are no up-wind stands that affect stand 1. Thus, (0) and (1) are the only two possible states at step 1 that have effects on further stands, as in **Figure 1C**.

In step 2, the number of states increases to 4 overall states, including actions in stand 2 and actions in the previous step, because the following steps depend on these two stands. The number of states keeps increasing in the third step to be $2^{\sqrt{4}+1} = 8$ states, before it reduces to only 2 states in the last step (H_4 and D_4) because there are no more dependent stands.

2.3. Solution of the MDP Problem

In this section, the exact solution of the MDP, a genetic algorithm (GA), and DQN are presented, alongside with their challenges and complications for comparison purposes. The proposed solution in this paper is also explained together with its advantages.

2.3.1. Value Iteration Algorithm

The exact solution of a finite-horizon MDP is provided by solving the Bellman Equation (22) by backward induction and value iteration, to determine the optimal value v^* under the optimal policy π^*.

$$v(s_m) \leftarrow \max_{a_m} \left\{ r(s_m, a_m) + \sum_{s_{m+1} \in \mathcal{S}} p(s_{m+1}|s_m, a_m) v(s_{m+1}) \right\} \tag{22}$$

Algorithm 1 solves the MDP *via* value iteration and obtains the optimal policy and the optimal value function (Bertsekas, 2005). At each step, **Algorithm 1** moves over all states per each action, requiring full knowledge of the forest model. Thus, the

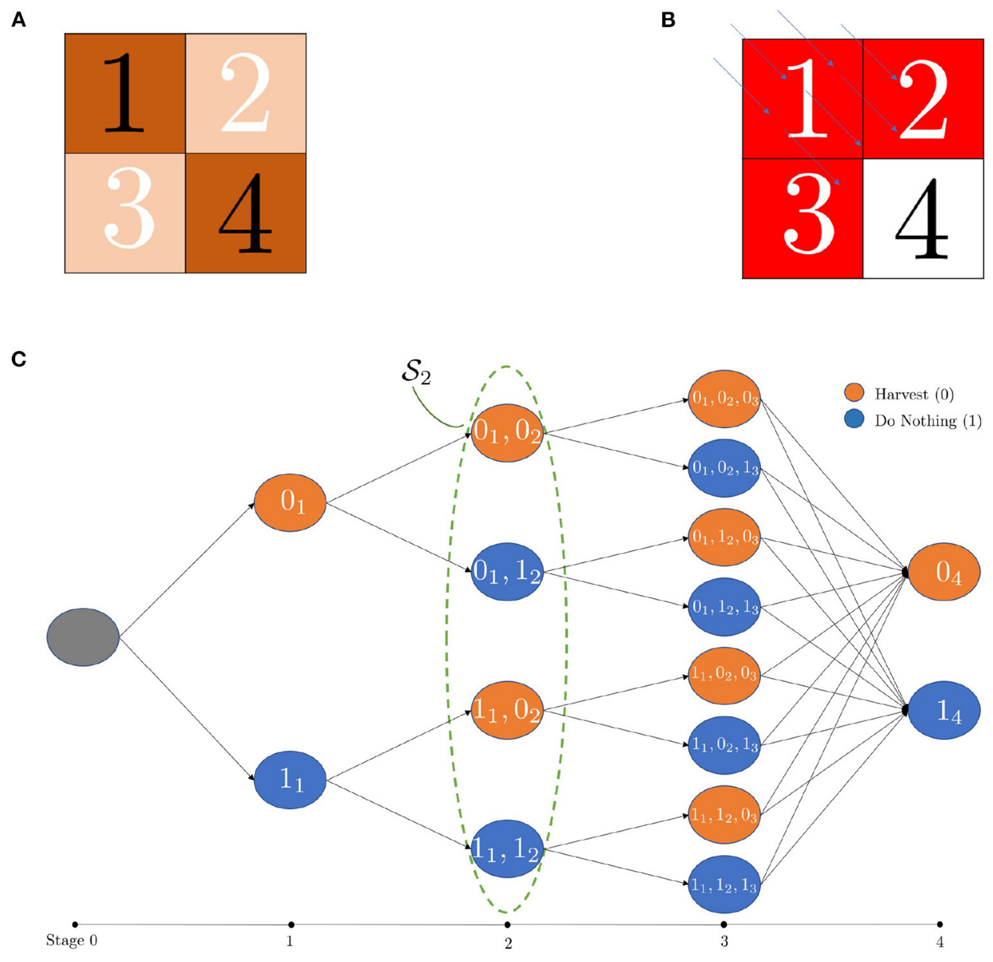

FIGURE 1 | (A) 2×2 matrix forest. **(B)** Northwest wind (blue) and the up-wind stands (red) for stand 4. **(C)** The state space of 2×2 stands.

complexity of the **Algorithm 1** is $\mathcal{O}(M \times \mathcal{S} \times \mathcal{A})$ which becomes prohibiting for combinatorial state spaces for large models. For example, in the case of 3×3 stands, there are only $2^3 + 1 = 16$ states per step while a larger forest model of 100×100 stands has 2^{101} states per step. This huge state number that makes the optimal solution intractable by **Algorithm 1** for large models.

2.3.2. Genetic Algorithm

Genetic Algorithm (GA) is a global optimization heuristics tool (Man et al., 1996) that is applicable to a variety of representations including MDP problems. In GA, a fixed population of solutions is first randomly created. The objective function (the fitness function) is computed for each of these solutions. These solutions are then sorted based on their fitness scores to select the best among them. Usually, the best half of the population is chosen as the so-called parents seeds. New solutions through offspring generations are created by a crossover process between two parents. Mutations by slightly changing some genes in the new solutions are taking place after crossover to create a more diverse population and to enable wider exploration of the state space. Hence, a new generation with the same population size is formed at each step that has improved performance. The same process is then repeated until convergence .

GA is used in this work as a benchmark solution for large-scale models where value iteration is difficult to implement. However, the population size must be large enough to adequately explore the state space and converge to a relevant value. The number of generations must be large as well to start seeing acceptable results. Thus, the high computational complexity that depends on the population size, number of generations, state space dimensions, and the complexity of evaluating the objective function often makes GA inefficient and expensive to implement. **Algorithm 2** summarizes the GA used in this work for solving the problem.

2.3.3. Proposed Deep Reinforcement Learning Solution

In this section, we consider an off-policy actor-critic algorithm with experience replay to overcome the discussed curse of dimensionality and be able to solve large and complex forest models. In the actor-critic algorithm, two neural networks interact with each other; the critic network evaluates the state value function following the policy updated by the actor-network,

Algorithm 2: Genetic Algorithm implementation.

```
1  Generate population $\mathcal{P}$ of random policies of size $\mathcal{N}$
2  Set $c$ and $\eta$
3  for $generation = 1, 2, ...$ do
4      for $n = 1, 2, ..., \mathcal{N}$ do
5          Compute fitness score by evaluating the policy $n$
6      Select $\frac{\mathcal{N}}{2}$ policies (parents) from $\mathcal{P}$ according to highest fitness scores
7      for $i = 1, 2, ....., \frac{\mathcal{N}}{2}$ do
8          Select two policies
9          Generate offspring by crossover the two policies at $c$ level
10         Perform mutation for the offspring by switching actions in $\mu$ stands
11     Update population $\mathcal{P} \leftarrow [\frac{\mathcal{N}}{2}$ parents $\frac{\mathcal{N}}{2}$ offsprings$]$
```

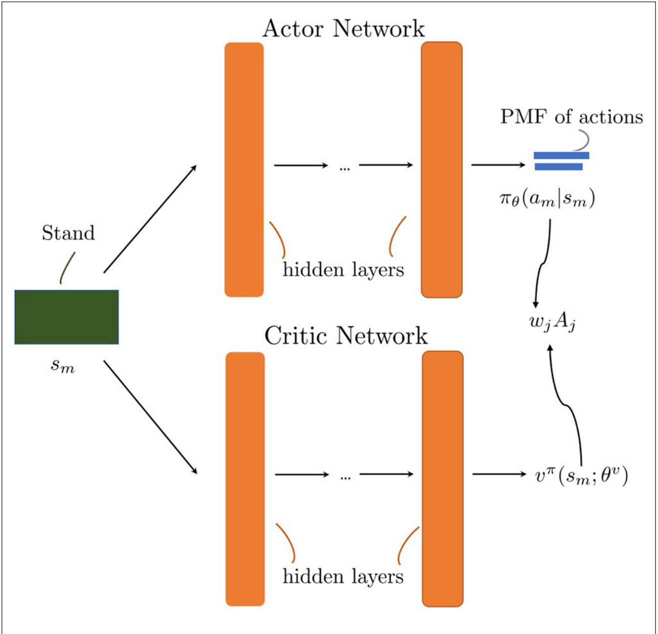

FIGURE 2 | Actor and critic networks with their interaction to compute the weighted advantage that is needed for the gradients.

as shown in **Figure 2**. The actor network updates the policy distribution and learns the optimal policy through the policy gradients. The networks are trained separately and update their relevant weights at each step.

The actor-network outputs the probability mass function over all possible actions using a softmax function:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{23}$$

where z_i is the input vector element of the action $i = 1, .., K$. As the input vector is $\mathbf{z}$, σ is the probability output vector with the sum of components equaling 1. The actor network updates the policy using the policy gradient, with the aid of the critic

Algorithm 3: Off-Policy Actor-Critic with Experience Replay

```
1  Initialize experience replay
2  Initialize actor and critic network weights $\boldsymbol{\theta}^\pi$, $\boldsymbol{\theta}^v$
3  select $\epsilon > 0$
4  for $episode = 1, 2, ...$ do
5      for $m = 1, 2, ..., M$ do
6          Select action $a_m$ at random with probability $\epsilon$, otherwise select action $a_m$ according to policy $\pi_{\boldsymbol{\theta}}(a_m|s_m)$
7          Receive reward $r(s_m, a_m)$ and next state $s_{m+1}$
8          Store transition $(s_m, \mu_m, r(s_m, a_m), s_{m+1})$ in experience replay
9          Sample batch $(s_j, \mu_j, r(s_j, a_j), s_{j+1})$ from experience replay
10         Compute the advantage $A_j$
11         Update actor parameters $\boldsymbol{\theta}^\pi$ according to gradient:
12         $\mathbf{g}_{\boldsymbol{\theta}^\pi} = \sum_{j=0} w_j \nabla_{\boldsymbol{\theta}^\pi} \log \pi\left(a_j|s_j\right) A_j$
13         Update critic parameters $\boldsymbol{\theta}^v$ according to gradient:
14         $\mathbf{g}_{\boldsymbol{\theta}^v} = \sum_{j=0} w_j \nabla_{\boldsymbol{\theta}^v} v^\pi\left(s_j\right) A_j$
```

network as in (18). The critic network assists the actor network by approximating the value function $v^\pi(s_m; \boldsymbol{\theta}^v)$ needed for stand m, that is needed to compute the advantage function $A^{\pi_{\boldsymbol{\theta}}}(s_m, a_m; \boldsymbol{\theta}^v)$ for the policy gradient as in (17). The inputs for the critic network are the index of stand m and the state $s_m \in \mathcal{S}_m$ for the stand m, which as mentioned is the combination of different relevant actions of the forest model. The loss function of the critic network is the mean squared error to minimize the TD error together with the importance sampling weight (Andriotis and Papakonstantinou, 2019):

$$L_v(\boldsymbol{\theta}^v) = \mathbb{E}_{s_m\sim\rho, a_m\sim\mu}\left[w_m(r(s_m, a_m) + v^\pi(s_{m+1}; \boldsymbol{\theta}^v) - v^\pi(s_m; \boldsymbol{\theta}^v))^2\right] \tag{24}$$

where ρ, μ, and w_m are as explained in (18). The weights θ^v are updated by computing the corresponding gradient (Andriotis and Papakonstantinou, 2019):

$$\mathbf{g}_{\boldsymbol{\theta}^v} = \mathbb{E}_{s_m\sim\rho, a_m\sim\mu}\left[w_m \nabla_{\boldsymbol{\theta}^v} v^\pi(s_m; \boldsymbol{\theta}^v) A^{\pi_{\boldsymbol{\theta}}}(s_m, a_m; \boldsymbol{\theta}^v)\right] \tag{25}$$

The pseudocode of actor-critic algorithm is presented in **Algorithm 3**.

The off-policy actor-critic algorithm is selected over other DRL approaches due to its discussed advantages, such as sample efficiency and scalability in solving large MDP problems, and can be similarly applied not only to the herein presented fire propagation model but to others as well, as illustrated in Degris et al. (2012) and Wang et al. (2016). However, the DQN algorithm in Mnih et al. (2015), described in Section 2.1.3, is also implemented in this paper to compare its performance with the proposed algorithm.

TABLE 2 | The flammability and timber volume.

Age	Initial volume ($m^3/hectare$)	Final volume ($m^3/hectare$)	Flammability (%)
1		0	10
8	100	106	35
9	106	110	30
10	110	114	25

3. RESULTS

To demonstrate the proposed solution, we consider three different forest sizes in this section. The DRL solution is also compared with the exact value iteration for the small dimension model, where such an exact solution can be obtained. GA solutions are used as benchmarks to evaluate the DRL results in the large-scale models, where exact solutions cannot be obtained.

For all experiments, the probability of fire at each stand is computed using Equation (21). The ignition probability is fixed for all stands to be $\text{Ig}(m) = 0.08$. The flammability that depends on the stand age is shown in **Table 2**. The fire spread probability from diagonally adjacent stands, i.e., NW direction, is set to be $S(NW) = 0.75$, while from side way adjacent stands is $S(N) = S(W) = 1$.

Two neural networks with architectures of 2 fully connected hidden layers are used for the actor and critic networks of **Algorithm 3**. A vector that includes the state variables and the index of the stand is used as an input to both networks. Rectified linear unit (ReLU) is the used activation function in the two hidden layers. The output layer of the actor network has 2 neurons with a softmax activation function, to obtain the probability mass function of the two possible actions for each stand. The output of the critic network provides the value function estimation and it has one neuron with a linear activation function at that output layer. The Adam optimizer (Kingma and Ba, 2014) with a learning rate of 1×10^{-4} for the actor network and 1×10^{-3} for the critic network is used. A batch size of 64 is used to train the networks at each step. The exploration rate is high at the beginning of the learning process and the probability of choosing a random action starts at 100% and decreases during learning until it reaches a 1% value. The number of the network neurons and the size of the memory buffer differ in each experiment. The Keras-Tensorflow library in Python is used to implement **Algorithm 3**. All experiments are run on a computer with Intel(R) Core(TM) i5-2400 CPU at 3.19 GHz processor and 8 GB RAM.

3.1. Case Study 1: 3 × 3 Stands

A forest model of 9 stands that are designed in a 3 × 3 matrix as in **Figure 3A** is presented in this example. Different age classes are assigned to each stand as in **Figure 3B**. Each age class corresponds to a timber volume and flammability as described in **Table 2**. These values are used to reflect the inventory and flammability of a eucalypt (Eucalyptus globulus Labill) plantation in Portugal, at different ages.

The maximum number of states in this 3 × 3 case is $2^{\sqrt{3}+1} = 16$ and the state space is shown in **Figure 4**. Because there are two actions for each stand (H or D), the number of states is 2, 4, and 8 for stands 1, 2, and 3, respectively. There are 16 states for the stands in the remaining rows. The number of states is reduced for stand 6 because the first three stands (first row) are no longer needed for the remaining stands. This reduction in the number of states is repeated for any stand located at the end of each row.

As the 3×3 forest is relatively small, the problem can be solved by **Algorithm 1**, in order to compute the exact optimal value and compare it to the implemented DRL approach outlined in **Algorithm 3**. To show the validity and eligibility of the proposed **Algorithm 3**, the DQN algorithm is also implemented in this case. Using the described architectures, the neural networks here have 128 neurons each. The size of the memory buffer is 1×10^4 to store the last 10,000 transition tuples, i.e., (s_m, a_m, s_{m+1}, r_m).

The obtained optimal value using **Algorithm 1** is 952.83 m^3/ha and the DRL approach in **Algorithm 3** managed to achieve the same value in less than 1, 200 episodes, as shown in **Figure 5**. DQN algorithm shows a worse convergence compared to **Algorithm 3**. As a result, only **Algorithm 3** is considered in the next cases as the preferred DRL approach. Detailed comparisons between DRL algorithms are beyond the scope of this paper.

The light blue and green lines show the rewards in one episode for **Algorithm 3** and the DQN algorithm, respectively. A one-episode line is defined as a total expected combined reward from stand 1 to stand 9. As the actions are changing from one episode to another, the total expected reward is also changing until convergence. This variation makes monitoring the learning process difficult. Hence, the movings average of total rewards over 50 previous episodes is computed and displayed in **Figure 5**.

3.2. Case Study 2: 10 × 10 Stands

In this second example there are 100 stands distributed in a 10 × 10 matrix. The age-class distribution is shown in **Figure 6**. Due to the high computational complexity, the value iteration algorithm can not be implemented in this case due to the high number of states, as mentioned in Section 2.3.1. Instead, the GA approach in **Algorithm 2** is used as a benchmark solution to compare with the proposed DRL approach. A GA population size of $\mathcal{N}$ =2,000 is used for 100 generations. The population is updated by recombining every two policies at cross point $c = 50$. Mutation is then performed by randomly switching the actions in $\eta = 10$ stands.

DRL approach is also applied to this 10 × 10 model case. The architectures of the neural networks are the same as in the previous 3 × 3 case except for the number of neurons that is increased now to 512 for the hidden layers in both networks, as the state vector has increased. A larger memory buffer of size 1×10^5 is also used. **Figure 7** shows the GA obtained value as well as the actor-critic algorithm solution for the 10 × 10 model case using the moving average of the last 50 episodes.

Figure 7 illustrates how the agent is learning until it converges to a value that is 18% higher than the converged GA value

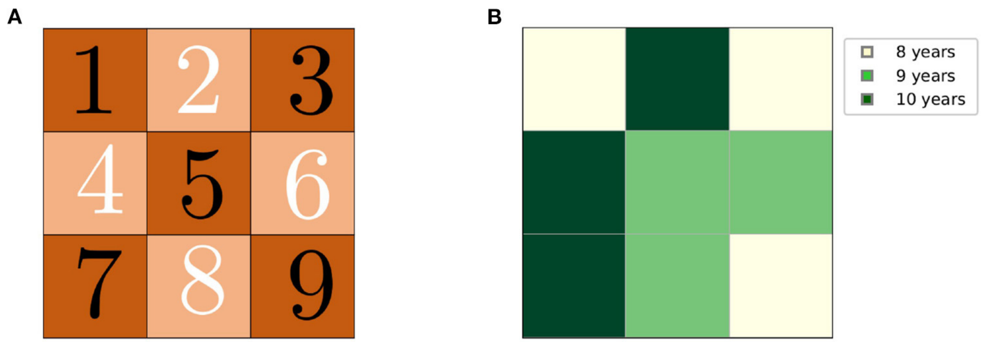

FIGURE 3 | (A) The 3 × 3 stands formation. **(B)** Timber ages for each stand.

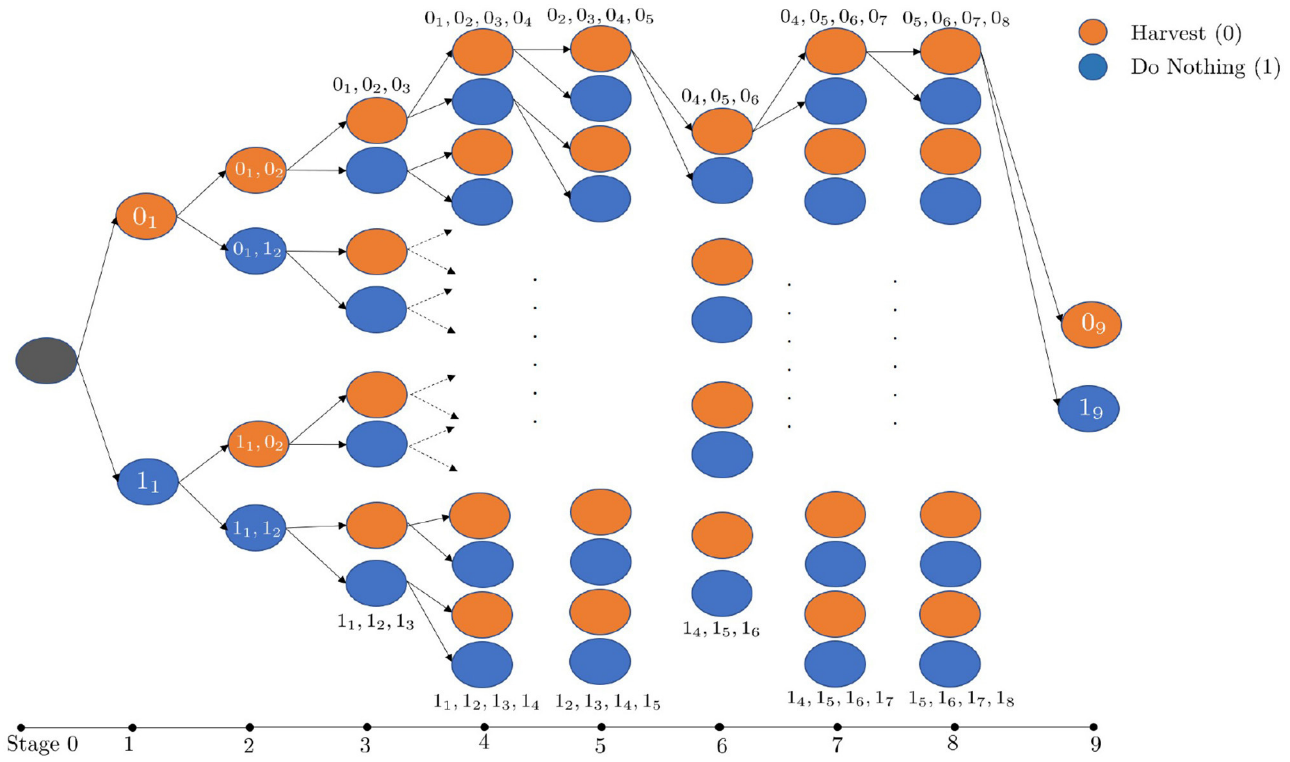

FIGURE 4 | The state space of the 3 × 3 stands case. The binary values indicate the actions taken and the lower subscript is the stand index.

(as compared to the random initial policy) in less than 1,000 episodes. DRL rewards exceed GA rewards in 1.68 times faster computational time.

3.3. Case Study 3: 100 × 100 Stands

The main goal of the presented DRL approach is of solving large-scale forest models that classical solutions are incapable to solve. In this 100 × 100 case there are more than $> 2^{101}$ possible states, but using the DRL method, the state input of the networks is the current trajectory which is a vector of length 102, including the location of the stand. The stand ages used are described in **Figure 8**. This example is also solved using GA, for comparison purposes, with a population size $\mathcal{N}$=1,000 in 100 generations, c =5,000, and $\eta = 100$.

Using **Algorithm 3** with the same neural networks architectures as before, 1,024 neurons in hidden layers, and a memory buffer of size 1×10^6, around 40 episodes were enough in this case to reach a stable value. **Figure 9** shows the resulting volume with the computed policy for the 100 × 100 stands model using the moving average of the last 10 episodes. DRL solution

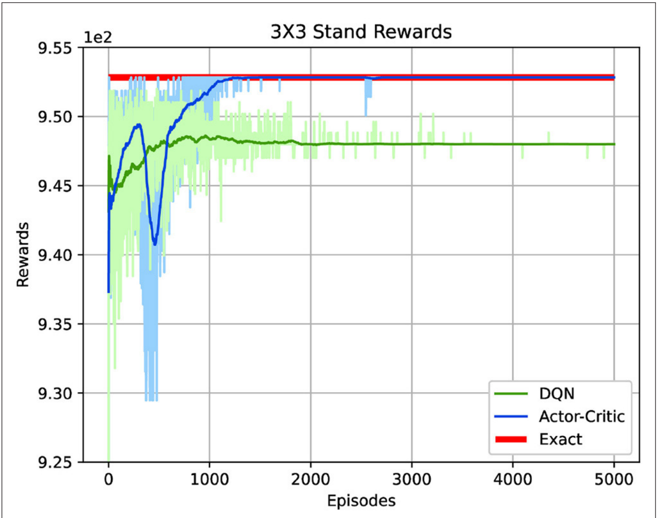

FIGURE 5 | Obtained DRL solutions compared to the optimal value for the 3 × 3 stands case.

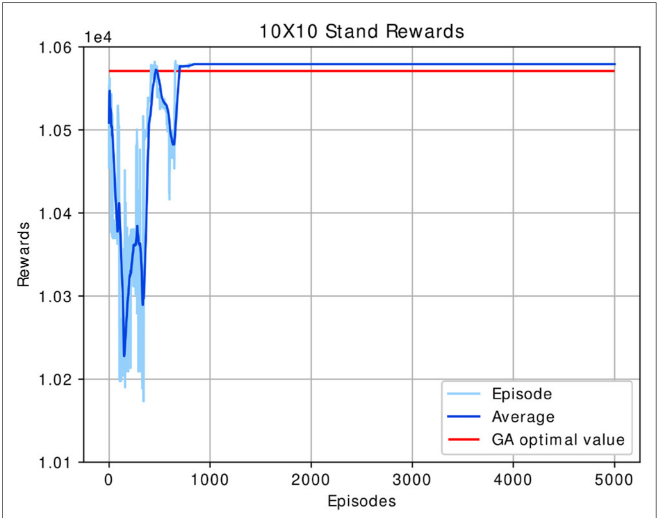

FIGURE 7 | Obtained DRL solution for the 10 × 10 stands case compared with the GA obtained value.

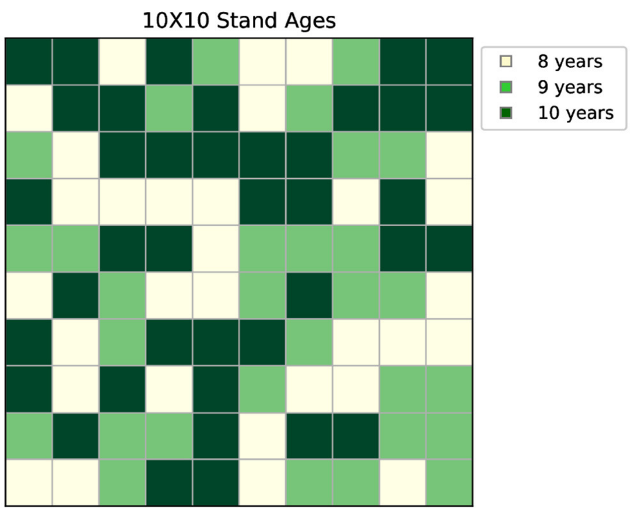

FIGURE 6 | Timber ages for the 10 × 10 stands case.

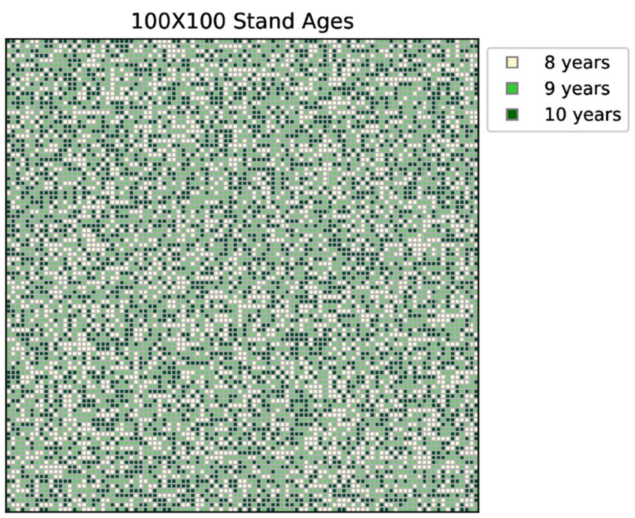

FIGURE 8 | Timber ages for the 100 × 100 stands case.

provides a 78% improvement over the GA one (as compared to the random initial policy), and achieves this 108 times faster than GA. **Table 3** summarizes the results of both the 10 × 10 and 100 × 100 cases.

4. DISCUSSION

DRL is shown to provide good results in solving wildfire risk management problems that are intractable by other methods. For small-size models such as 3 × 3, the proposed solution achieves the optimal value of timber volume as shown in **Figure 5**. The suggested **Algorithm 3** also exceeds the DQN result, as anticipated and explained before. The optimal policy obtained from both algorithms is illustrated in **Figure 10**. Since the wildfire propagates in the NW direction in this model, the optimal policy shows that the stands are harvested in that direction to mitigate the wildfire propagation.

The DRL approach also works well in the larger model of 10 × 10 stands. In this case, the exact solution from **Algorithm 1** is infeasible and GA is used instead. The main point behind comparing to GA is getting an indication that DRL is converging to the right solution. The DRL solution in **Algorithm 3** converges to a timber volume that is higher than the attained GA value by 18%, as shown in **Figure 7**. The DRL solution obtains this value in faster computational time compared to GA that is known for its high time complexity. Similar to the 3 × 3 case, the DRL obtained policy of the 10 × 10 stands case is shown in **Figure 11**

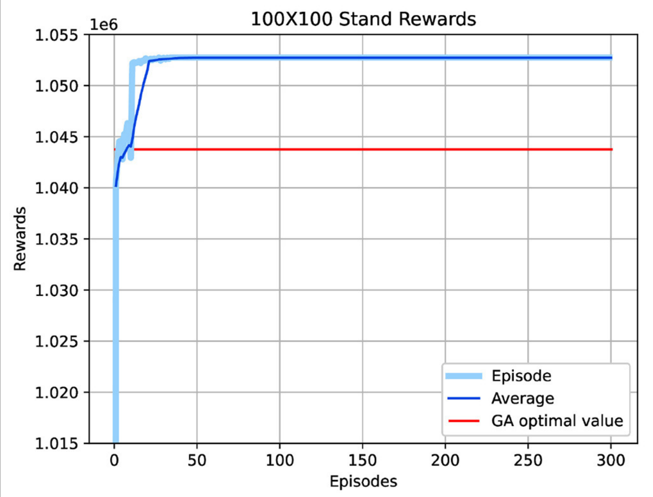

FIGURE 9 | Obtained DRL solution for the 100 × 100 stands case compared with the GA obtained value.

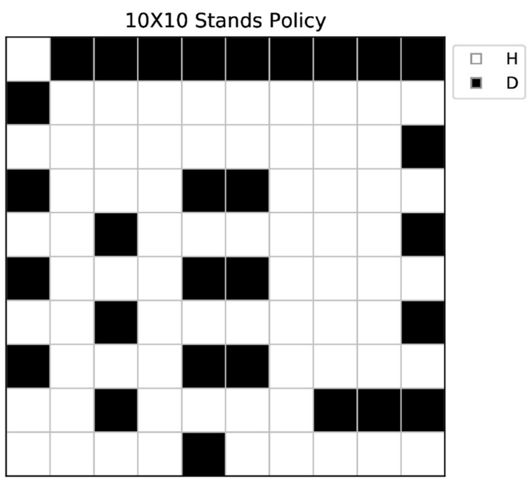

FIGURE 11 | Computed DRL policy for the 10 × 10 stands case, where stand 1 is at the upper left corner.

TABLE 3 | Summary of DRL performance compared to GA for Cases 2 and 3.

Case	Rewards (%)	Computational time
10 × 10 stands	DRL 18% ↑	DRL = GA/1.68
100 × 100 stands	DRL 78% ↑	DRL = GA/108

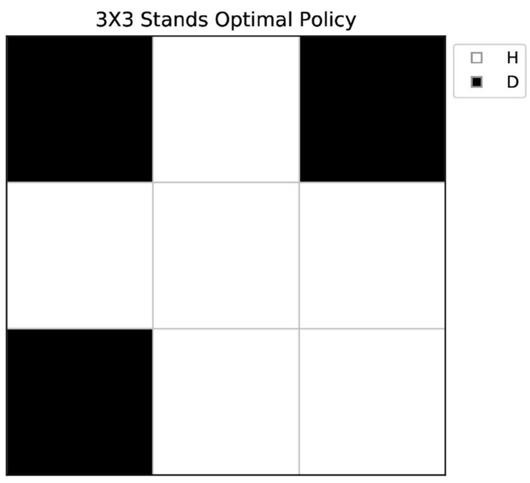

FIGURE 10 | The optimal policy of the 3 × 3 stands case.

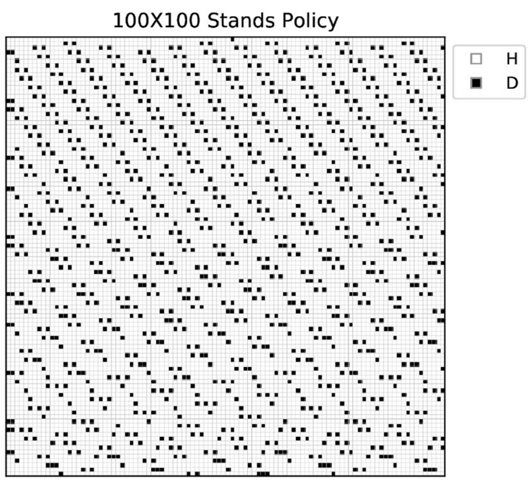

FIGURE 12 | Computed DRL policy for the 100 × 100 stands case.

and the agent tries to mitigate the fire propagation by harvesting in the NW direction. In the first line, the wildfire propagates from only one side since the NW and N up-wind stands do not exists. Thus, the algorithm has chosen not to harvest in these stands.

The difference in the value of timber volume and time complexity between the two algorithms becomes obvious in a large-scale model. **Figure 9** for the 100 × 100 stands case shows that the timber volume reached by the DRL algorithm surpasses the obtained GA value by 78% achieving that 108 times faster. Thus, in terms of speed, GA is greatly suboptimal to DRL and it is considered only as a benchmark, not as practical way of solving a large problem. **Figure 12** illustrates the DRL obtained policy for the 100 × 100 stands case. As the ignition probability Ig(m) in equation(21) is the same for all stands, the agent suggests harvesting zones consistent with the NW wind in a largely repeated pattern because of the large number of stands. The risk of wildfire propagation decreases between stands if they are far away from each other.

Comparing **Figures 5–7**, **9**, we notice that the policy converges in fewer episodes as the number of stands increases. The reason behind this is that in the 100 × 100 stands case, one episode corresponds to 10,000 learning steps. At each step, a sample batch is drawn from the experience replay, TD targets are computed, and a gradient step is taken to update the neural network parameters. Thus, one episode corresponds to 10,000 gradient descent steps in the 100 × 100 case, compared to 9 and 100 gradient steps in one episode of the 3 × 3 case and 10 × 10 case, respectively.

In conclusion, we consider the wildfire propagation problem given the NW wind direction during one period. The probability of the stand getting burned is either for the wildfire starting at that stand or one of the adjacent stands. The entire forest model is designed as a large-scale MDP with spatial dynamics. As classical solutions of large-scale MDPs are inefficient due to the curse of dimensionality, a tractable and efficient DRL algorithm is proposed for the solution of large-scale problems. DRL solution is implemented using the off-policy actor-critic algorithm. Obtained DRL matched exact values, when these could be computed, and surpassed DQN and GA benchmarks both in computational time and obtained objective function value. We have shown that the proposed DRL approach can estimate a near-optimal policy even for very complex problems where the number of relevant possible decisions is extremely high. Hence, DRL is an efficient new approach to solve large-scale problems of this type, with many capabilities for generalizations and several extensions.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors upon request, without undue reservation.

AUTHOR CONTRIBUTIONS

AA, CL, JB, and MM developed the idea and designed the model. CA and KP contributed on the methodology and discussion. AA and CL conducted the experiments and drafted the manuscript. All co-authors provided assistance in writing the manuscript and approved the submitted version.

FUNDING

This research was supported by the Portuguese Science Foundation (FCT), Portugal, namely through the Forest Research Center Project UIDB/00239/2020, the BIOECOSYS project (grant number LISBOA-01-0145-FEDER-030391, PTDC/ASP-SIL/30391/2017), and the MODFIRE project (grant number PCIF/MOS/0217/2017).

ACKNOWLEDGMENTS

AA thanks Taibah University in Saudi Arabia for funding his research activities.

REFERENCES

Alexander, M. E., and Cruz, M. G. (2013). Are the applications of wildland fire behaviour models getting ahead of their evaluation again? *Environ. Model. Softw*. 41, 65–71. doi: 10.1016/j.envsoft.2012.11.001

Andriotis, C. P., and Papakonstantinou, K. G. (2019). Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliabil. Eng. Syst. Saf*. 191, 106483. doi: 10.1016/j.ress.2019.04.036

Andriotis, C. P., and Papakonstantinou, K. G. (2021). Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints. *Reliabil. Eng. Syst. Saf*. 212, 107551. doi: 10.1016/j.ress.2021.107551

Bertsekas, D. P. (2005). *Dynamic Programming and Optimal Control, Vol. 1*. Belmont, MA: Athena Scientific.

Botequim, B., Arias-Rodil, M., Garcia-Gonzalo, J., Silva, A., Marques, S., Borges, J. G., et al. (2017). Modeling post-fire mortality in pure and mixed forest stands in Portugal-a forest planning-oriented model. *Sustainability* 9, 390. doi: 10.3390/su9030390

Castellnou, M., Prat-Guitart, N., Arilla, E., Larra naga, A., Nebot, E., Castellarnau, X., et al. (2019). Empowering strategic decision-making for wildfire management: avoiding the fear trap and creating a resilient landscape. *Fire Ecol*. 15:31. doi: 10.1186/s42408-019-0048-6

Degris, T., White, M., and Sutton, R. S. (2012). Off-policy actor-critic. *arXiv [Preprint]*. arXiv: 1205.4839v5. Available online at: https://arxiv.org/pdf/1205.4839.pdf

Faivre, N., Rego, F., Moreno, J., Vallejo, V., and Xanthopoulos, G. (2018). "Forest fires-sparking firesmart policies in the EU," in *Research & Innovation Projects for Policy. DG Research and Innovation* (Luxembourg: European Commission), 48.

Ferreira, L., Constantino, M., and Borges, J. G. (2014). A stochastic approach to optimize maritime pine (pinus pinaster ait.) stand management scheduling under fire risk. An application in Portugal. *Ann. Oper. Res*. 219, 359–377. doi: 10.1007/s10479-011-0845-z

Ferreira, L., Constantino, M. F., Borges, J. G., and Garcia-Gonzalo, J. (2015). Addressing wildfire risk in a landscape-level scheduling model: an application in Portugal. *For. Sci*. 61, 266–277. doi: 10.5849/forsci.13-104

Finney, M. A. (2004). *FARSITE: fire area simulator-model development and evaluation*. Research Paper RMRS-RP-4 Revised. U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, Ogden, UT, United States.

Finney, M. A. (2006). "An overview of flammap fire modeling capabilities," in *Fuels Management-How to Measure Success: Conference Proceedings*, eds P. L. Andrews and B. W. Butler (Portland, OR; Fort Collins, CO: US Department of Agriculture, Forest Service, Rocky Mountain Research Station), 213–220. volume 41.

Finney, M. A., and Andrews, P. L. (1999). Farsite-a program for fire growth simulation. *Fire Manage. Notes* 59, 13–15.

Ganapathi Subramanian, S., and Crowley, M. (2018). Using spatial reinforcement learning to build forest wildfire dynamics models from satellite images. *Front. ICT* 5, 6. doi: 10.3389/fict.2018.00006

González, J. R., Palahí, M., Trasobares, A., and Pukkala, T. (2006). A fire probability model for forest stands in Catalonia (north-east Spain). *Ann. For. Sci*. 63, 169–176. doi: 10.1051/forest:2005109

González-Olabarria, J.-R., and Pukkala, T. (2011). Integrating fire risk considerations in landscape-level forest planning. *For. Ecol. Manage*. 261, 278–287. doi: 10.1016/j.foreco.2010.10.017

Goss, M., Swain, D. L., Abatzoglou, J. T., Sarhadi, A., Kolden, C. A., Williams, A. P., et al. (2020). Climate change is increasing the likelihood of extreme autumn wildfire conditions across California. *Environ. Res. Lett*. 15, 094016. doi: 10.1088/1748-9326/ab83a7

Haksar, R. N. and Schwager, M. (2018). "Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Piscataway, NJ: IEEE), 1067–1074.

He, H. S., and Mladenoff, D. J. (1999). Spatially explicit and stochastic simulation of forest-landscape fire disturbance and succession. *Ecology* 80, 81–99. doi: 10.1890/0012-9658(1999)080[0081:SEASSO]2.0.CO;2

Hoganson, H., Borges, J. G., and Wei, Y. (2008). "Coordinating management decisions of neighboring stands with dynamic programming," in *Designing Green Landscapes, Vol. 15* (Dordrecht: Springer), 187–213. doi: 10.1007/978-1-4020-6759-4_8

Kingma, D. P., and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv [Preprint]*. arXiv: 1412.6980v9. Available online at: https://arxiv.org/pdf/1412.6980.pdf

Man, K.-F., Tang, K.-S., and Kwong, S. (1996). Genetic algorithms: concepts and applications. *IEEE Trans. Indus. Electron*. 43, 519–534. doi: 10.1109/41.538609

Marques, S., Garcia-Gonzalo, J., Botequim, B., Ricardo, A., Borges, J., Tomé, M., et al. (2012). Assessing wildfire occurrence probability in pinus pinaster ait. Stands in Portugal. *For. Syst*. 21, 111–120. doi: 10.5424/fs/2112211-11374

Marques, S., Marto, M., Bushenkov, V., McDill, M., and Borges, J. (2017). Addressing wildfire risk in forest management planning with multiple criteria decision making methods. *Sustainability* 9, 298. doi: 10.3390/su9020298

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning* (New York, NY: PMLR), 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. *arXiv [Preprint]*. arXiv: 1312.5602. doi: 10.48550/arXiv.1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi: 10.1038/nature14236

Molina, J. R., González-Cabán, A., and Rodríguez y Silva, F. (2019). Potential effects of climate change on fire behavior, economic susceptibility and suppression costs in Mediterranean ecosystems: Córdoba province, Spain. *Forests* 10, 679. doi: 10.3390/f10080679

Moreira, F., Ascoli, D., Safford, H., Adams, M. A., Moreno, J. M., Pereira, J. M., et al. (2020). Wildfire management in Mediterranean-type regions: paradigm change needed. *Environ. Res. Lett*. 15, 011001. doi: 10.1088/1748-9326/ab541e

Pacheco, A. P., Claro, J., Fernandes, P. M., de Neufville, R., Oliveira, T. M., Borges, J. G., et al. (2015). Cohesive fire management within an uncertain environment: a review of risk handling and decision support systems. *For. Ecol. Manage*. 347, 1–17. doi: 10.1016/j.foreco.2015.02.033

Papakonstantinou, K. G., and Shinozuka, M. (2014). Planning structural inspection and maintenance policies via dynamic programming and markov processes. *Part I Theory Reliabil. Eng. Syst. Saf*. 130, 202–213. doi: 10.1016/j.ress.2014.04.005

Puterman, M. L. (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley and Sons.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv [Preprint]*. arXiv: 1511.05952. doi: 10.48550/arXiv.1511.05952

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv [Preprint]*. arXiv: 1506.02438. doi: 10.48550/arXiv.1506.02438

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems* (Cambridge, MA: The MIT Press), 1057–1063.

Van Hasselt, H., Guez, A., and Silver, D. (2016). "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30* (Palo Alto, CA: AAAI Press), 2094–2100.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., et al. (2016). Sample efficient actor-critic with experience replay. *arXiv [Preprint]*. arXiv: 1611.01224. Available online at: https://arxiv.org/pdf/1611.01224.pdf

Watkins, C. J., and Dayan, P. (1992). Q-learning. *Mach. Learn*. 8, 279–292. doi: 10.1023/A:1022676722315

Wawrzyński, P. (2009). Real-time reinforcement learning by sequential actor-critics and experience replay. *Neural Netw*. 22, 1484–1497. doi: 10.1016/j.neunet.2009.05.011

Wei, Y. (2012). Optimize landscape fuel treatment locations to create control opportunities for future fires. *Can. J. For. Res*. 42, 1002–1014. doi: 10.1139/x2012-051

Wei, Y., Rideout, D., and Kirsch, A. (2008). An optimization model for locating fuel treatments across a landscape to reduce expected fire losses. *Can. J. For. Res*. 38, 868–877. doi: 10.1139/X07-162

Williams, A. P., Abatzoglou, J. T., Gershunov, A., Guzman-Morales, J., Bishop, D. A., Balch, J. K., et al. (2019). Observed impacts of anthropogenic climate change on wildfire in California. *Earth's Fut*. 7, 892–910. doi: 10.1029/2019EF001210

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.