Predictive Explanations for and by Reinforcement Learning

Léo Saulières, Martin C Cooper, Florence Dupin de Saint-Cyr

Predictive explanations for and by Reinforcement Learning*

Léo Saulières Martin C. Cooper Florence Dupin de Saint-Cyr

IRIT, University of Toulouse III, France

Abstract

In order to understand a reinforcement learning (RL) agent's behavior within its environment, we propose an answer to 'What is likely to happen?' in the form of a predictive explanation. It is composed of three scenarios: best-case, worst-case and most-probable which we show are computationally difficult to find (W[1]-hard). We propose linear-time approximations by considering the environment as a favorable/hostile/neutral RL agent. Experiments validate this approach. Furthermore, we give a dynamic-programming algorithm to find an optimal summary of a long scenario.

Keywords: Explainable Artificial Intelligence, Reinforcement Learning

1 Introduction

The need for explanations of black-box Artificial Intelligence models, highlighted by researchers [16, 7] but also by legislators [8], has become an important topic during the last few years. As a consequence, the research field of eXplainable Artificial Intelligence (XAI) has developed, with the aim of building trustworthy AI models. These models can be then used in a wider range of applications, including high-risk or safety-critical ones. The aim of this paper is to explore new avenues of explanations in Reinforcement Learning (RL). RL can be summarized as follows. An agent learns while making a sequence of decisions consisting of actions within an environment. At each time-step, the information available to the agent defines a state. In a state, the agent chooses an action, and so arrives in a new state, determined by a transition function (which is not necessarily deterministic), and receives a reward (a negative reward being rather a punishment). The agent aims at maximizing its reward, while striking a balance between exploration (discover new ways to face the problem) and exploitation (use already learnt knowledge). The agent's strategy is learnt in the form of a policy, which maps each state to either an action (if the policy is deterministic) or a probability distribution over actions (if the policy is stochastic).

The subfield of XAI which focuses on RL models is eXplainable Reinforcement Learning (XRL). Researchers used key RL features to explain agent's decisions. As an example, the VIPER algorithm [2] learns a Decision Tree policy which is a surrogate for the actual policy given by a deep neural network. The surrogate policy is easier to verify concerning different properties such as safety, stability and robustness. With states considered as images, [10, 22] explain agent's decisions in different ways. Greydanus *et al.* use a perturbation-based approach to generate a saliency map, i.e. parts of an image that lead the agent to choose an action [10]. Given a state s and a selected action a, a GAN produce a counterfactual state s', to show the user in which settings the agent would select an action b instead of a [22]. Another approach is reward decomposition [15] which focuses on the reward function and is used when an agent has multiple objectives. This XRL method expresses a reward through a vector of scalars instead of a simple scalar. This makes it easier to understand why an agent performs an action, and to identify its objective in choosing this action.

In their survey, Milani *et al.* emphasize the need for explanations that capture the concepts of RL [19]. Our study tries to meet this need by proposing a predictive XRL method based on the sequential aspect of RL. The aim of this method is to answer the question *"What is likely to happen from the state*

*This is a draft version, the article is published in: Rocha, A.P., Steels, L., van den Herik, J. (eds) Agents and Artificial Intelligence. ICAART 2023. Lecture Notes in Computer Science(), vol 14546. Springer, Cham. https://doi.org/10.1007/978-3-031-55326-4_6

s with the current policy of the agent?". To this end, we compute three different state-action sequences (called scenarios), starting from the current state s. This method allows us to explain a policy by giving pertinent examples of scenarios from s, hence the name of the explainer: Scenario-Explanation, shortened to SXp. It provides information about future outcomes by looking forward N time-steps according to three different scenarios: a worst-case scenario, a most-probable scenario and a best-case scenario. To avoid an exhaustive search over all possible scenarios, we propose approximations based on learning policies of hostile/favorable environments. Our approximate SXp's are computed using transition functions learnt by treating the environment as an RL agent. The advantage is that this can be achieved by using the same technology and the same computational complexity as the learning of the agent's policy. We tested our approximate SXp on two problems: Frozen Lake, an Open AI Gym benchmark problem [3], and Drone Coverage, a problem we designed.

This paper is an extended version of the paper [23]. The main additional contributions are a representativeness score and a method for summarizing the Scenario-Explanation. This last addition allows us to consider SXp's with a large N and thus to have a long term vision of what can happen, while keeping the explanations concise.

This paper first gives a theoretical justification for Scenario-Explanation and introduces new metrics for quantifying their quality and representativeness, before describing experimental results on two RL problems. For each problem, short and long scenarios explanations are computed, the long scenarios are summarized thanks to a quadratic algorithm. We then survey related work on XRL, before discussing the efficiency and usefulness of SXp.

2 Scenario-Explanation

Before describing our XRL method, we need to introduce some notation. An RL problem is described by a Markov Decision Process (MDP) [26]. An MDP is a tuple $\langle S, A, R, p\rangle$ where S and A are respectively the state and action space, $R : S \times A \to \mathbb{R}$ is the reward function, $p : S \times A \to Pr(S)$ is the transition function of the environment which provides a distribution over reachable states: given an action $a \in A$ and a state $s \in S$, $p(s'|s,a)$ denotes the probability of reaching the state s' when a is performed in the state s. For a deterministic policy $\pi : S \to A, \pi(s)$ denotes the action the agent performs in s whereas for a stochastic policy $\pi : S \to Pr(A)$, $\pi(a|s)$ denotes the probability that the agent performs action a in s.

Our aim is to answer the question: *"What is likely to happen from the state s with the current policy of the agent?"*. We choose to do this by providing three specific scenarios using the learnt policy π. By scenario, we mean a sequence of states and actions, starting with s. Scenarios are parameterised by their length, denoted by N, which we consider as a parameter determined by the user. We provide a summary of all possible scenarios via the *most-probable*, the *worst-case* and the *best-case* scenarios starting from s.

When considering possible scenarios, we may choose to limit our attention to those which do not include highly unlikely transitions or actions. The following technical definition based on two thresholds α and β allows us to restrict the possible transitions and actions. We do not filter out all transitions with probability less than a certain threshold, but rather those whose probability is small (less than a factor of α) compared to the most likely transition. This ensures that at least one transition is always retained. A similar remark holds for the probability of an action. We filter out those actions whose probability is less than a factor of β from the probability of the most probable action.

Definition 1 *Given $N \in \mathbb{N}^*$, $\alpha, \beta \in [0,1]$, an MDP $\langle S, A, R, p\rangle$ and a stochastic policy π over S, an* (α,β)-credible length-N scenario *is a state-action sequence $s_0, a_0, s_1, a_1, \ldots, a_{N-1}, s_N \in (S \times A)^N \times S$ which satisfies: $\forall i \in \{0,\ldots,N-1\}$, $\pi(a_i|s_i)/\pi^* \geq \beta$ and $p(s_{i+1}|s_i,a_i)/p^* \geq \alpha$, where $\pi^* = \max_{a \in A} \pi(a|s_i)$ and $p^* = \max_{s \in S} p(s|s_i,a_i)$.*

In a $(1,1)$-credible length-N scenario, the agent always chooses an action among it's most likely choices and we only consider the most probable transitions of the environment. At the other extreme, in a (0,0)-scenario, there are no restrictions on the choice of actions or on the possible transitions of the environment.

The following definition is parameterised by $\alpha, \beta \in [0,1]$ and an integer N. For simplicity of presentation, we leave this implicit and simply write credible scenario instead of (α,β)-credible length-N scenario.

In the following definition, $R(\sigma)$ denotes the reward of a credible scenario σ. By default $R(\sigma)$ is the reward attained at the last step of σ.

Definition 2 *For an MDP $\langle S, A, R, p\rangle$ and a policy π over S, a* scenario-explanation *for π from a state s is a credible scenario $\sigma = s_0, a_0, s_1, a_1, ..., a_{N-1}, s_N$ such that $s_0 = s$. σ is a* most-probable scenario-explanation *for π from s if its probability given s, denoted $Pr(\sigma)$, is maximum, where*

$$Pr(\sigma) = \prod_{i=1}^{N} \pi(a_{i-1}|s_{i-1}) p(s_i|s_{i-1}, a_{i-1})$$

σ is a best-case scenario-explanation *for π from s if it maximises the reward $R(\sigma)$. σ is a* worst-case scenario-explanation *from s if it minimizes the reward $R(\sigma)$.*

In the best (worst) case, the environment always changes according to the best (worst) transitions for the agent, i.e., the environment maximises (minimises) the agent's reward after N steps. Not surprisingly, finding such scenarios is not easy, as we now show.

Proposition 1 *For any fixed values of the parameters $\alpha, \beta \in [0,1]$, the problem of finding a best-case or worst-case length-N scenario-explanation, when parameterized by N, is W[1]-hard. Finding a most-probable length-N scenario-explanation is W[1]-hard provided $\alpha < 1$.*

The detailed proof of Proposition 1 can be found in [23].

2.1 Approximate Scenario-Explanation

In view of Proposition 1, we consider approximations to scenario-explanations which we obtain via an algorithm whose complexity is linear in N, the length of the SXp. Indeed, since determining most-probable/worst/best scenarios is computationally expensive, we propose to approximate them. For this purpose, it can be useful to imagine that the environment acts in a deliberate manner, as if it were another agent, rather than in a neutral manner according to a given probability distribution. In this paper, as a first important step, we restrict our attention to approximate SXp's that explain deterministic policies π. An *environment policy* π_e denotes a policy that models a specific behavior of the environment.

There are different policies π_e for the most-probable, worst and best cases which correspond to policies of neutral, hostile and favorable environments respectively. In the case of a hostile/favorable environment, π_e denotes an environment policy that aims at minimizing/maximizing the reward of the agent. The policy of a neutral environment is already given via the transition probability distribution p. On the other hand the policies of hostile or favorable environments have to be learnt. We propose to again use RL to learn these two policies. Compared to the learning of the agent's policy, there are only fairly minor differences. Clearly, in general, the actions available to the environment are not the same as the actions available to the agent. Another technical detail is that as far as the environment is concerned the set of states is also different, since its choice of transition depends not only on the state s but also on the action a of the agent.

Recall, from Definition 1, that in a (1,1)-scenario a most-probable action and a most-probable transition are chosen at each step. Of course, for deterministic policies or transition functions there is no actual choice.

Definition 3 *A* probable scenario-explanation *(P-scenario) of π from s is a (1,1)-scenario for π starting from s.*

A favorable-environment scenario-explanation *(FE-scenario) for π from s is a (1,1)-scenario, in which the transition function (p in Definition 1) is a learnt policy π_e of a favorable environment.*

A hostile-environment scenario-explanation *(HE-scenario) for π from s is a (1,1)-scenario, in which the transition function p is a learnt policy π_e of a hostile environment.*

A length-N P-scenario is computed by using an algorithm that simply chooses, at each of N steps starting from the state s, the action determined by π and a most-probable transition according to p. In the case of the FE/HE scenario, p is replaced by the environment policy π_e which is learnt beforehand. The same RL method that was used to learn the agent's policy π is used to learn π_e (which hence is deterministic since we assume that π is deterministic). The fact that the learnt environment policy π_e is deterministic

means that scenario-explanations can be produced in linear time. In the favorable-environment (FE) case the reward for the environment is R (the same function as for the agent) and in the hostile-environment (HE) case the reward function is (based on) $-R$.

Proposition 2 *Consider an MDP $\langle S, A, R, p\rangle$ for which we learn by RL a deterministic policy π. Producing length-N P/HE/FE scenario-explanations does not increase the asymptotic worst-case (time and space) complexity of the training phase. Moreover the computation of the explanation only incurs a cost which is linear in N.*

The proof of Proposition 2 can be found in [23]. Having shown that our algorithm is efficient in time, hence avoiding the complexity issue raised by Proposition 1, in Section 3 we describe experiments which indicate that the returned results are good approximations of the most-probable, best and worst explanations.

2.2 Scenario-Explanation rendering

The length of an SXp is parameterized by N, as already mentioned. Accordingly, the SXp's rendering to the user will depend on it. On one hand, for N relatively low, we simply display each scenario (FE-scenario/HE-scenario/P-scenario) to the user by assuming that the amount of information does not impact the user's understanding of the explanation. So, the SXp provides a summary of what is likely to happen in the short term. On the other hand, for N high, the SXp provides more information on the agent's future interaction with the environment. In this case, displaying the three scenarios can give rise to an important cognitive load for the user, thus making the explanation inefficient. In order to avoid this, we propose to summarize the scenarios for large values of N. Before describing how to summarize such scenarios, we define which goals the summary must achieve to be of good quality.

First, a scenario summary has to highlight the most important states within it. We call *state importance* the notion introduced by Clouse [4]. This notion is used to provide explanations in the form of a policy summary in [1]. Formally, given a Q function and a state s, the importance of s is defined as the difference between the best and worst Q value in s by performing an action a:

$$I(s) = \max_a Q(s,a) - \min_a Q(s,a)$$

Secondly, the summary should cover the entire scenario as well as possible. In other words, the selected states should be distributed as uniformly as possible along the scenario. We achieve this by minimizing the sum of the squares of the distances between consecutive selected states. The following definition expresses the *summary problem* discussed above through an objective function.

Definition 4 *Let $\sigma = (s_1, a_1, s_2, a_2, \ldots, a_{N-1}, s_N)$ be a length-N scenario and $I(s_1), \ldots, I(s_N)$ the corresponding state importance scores. For given constants $M \leq N$ and $\lambda \geq 0$, the* summary problem *consists in finding a sample of M states $s_{i_1}, \ldots, s_{i_M}$, with $1 = i_1 < i_2 < \ldots < i_M = N$ such that the following objective function is maximised:*

$$f_N(\sigma, i_1, \ldots, i_M) = \sum_{j=1}^{M} I(s_{i_j}) - \lambda \sum_{k=1}^{M-1} (i_{k+1} - i_k)^2$$

The spread-regularity score is given by the negation of the sums of the squares of the differences between consecutive values, i.e. $-\sum_{k=1}^{M-1}(i_{k+1}-i_k)^2$. Observe that maximizing this score is equivalent to minimizing the variance of these differences, since their average $\mu = \frac{1}{M-1}\sum_{k=1}^{M-1}(i_{k+1}-i_k) = \frac{1}{M-1}(i_M - i_1) = (N-1)/(M-1)$ is a constant and their variance $\frac{1}{M-1}\sum_{k=1}^{M-1}(i_{k+1}-i_k-\mu)^2 = \frac{1}{M-1}\sum_{k=1}^{M-1}((i_{k+1}-i_k)^2 - 2(i_{k+1}-i_k)\mu + \mu^2) = \frac{1}{M-1}\sum_{k=1}^{M-1}(i_{k+1}-i_k)^2 - \mu^2$.

Thus, with the objective function given in Definition 4, we want a sample of M states which have high state importance scores and are evenly spread. The summary problem can be solved using dynamic programming with the following equations.

Definition 5 *For $2 \leq m \leq n \leq N$ and a length-N scenario σ, let $g_\sigma(m,n)$ be the optimal value of $f_n(\sigma, i_1, \ldots, i_m)$ such that $i_1 = 1$ and $i_m = n$. We can calculate all values of $g(m,n)$ using dynamic programming via the equations*

$$\begin{aligned} g_\sigma(m,n) &= \max_{p \in \{m-1,\ldots,n-1\}} \left(g_\sigma(m-1,p) + I(s_n) - \lambda(n-p)^2 \right) \quad (2 < m \leq n \leq N) \\ g_\sigma(2,n) &= I(s_1) + I(s_n) + \lambda(n-1)^2 \qquad (2 \leq n \leq N) \end{aligned}$$

Once we have calculated $g_\sigma(M,N)$, we can determine for which values of p the maximum is reached: this gives us the values of $i_{M-1}, \ldots, i_2$, thus the states $s_{i_{M-1}}, \ldots, s_{i_2}$.

Note that the value of λ has a significant impact on the summary. Indeed, the higher the value is, the more evenly spread out the summary states are. On the contrary, a small value can lead to summaries which only maximize the sum of the importance of the states in the summary.

We normalize the two terms of the objective function f_N to lie in the interval $[0,1]$ to have the two terms within the same amplitude (hence f_N lies in the interval $[-\lambda, 1]$) whatever the amplitude of the Q-values and the scenario length. The normalization of a value b that is obtained by a function h is performed by means of the *norm* function which requires b, the value to be normalized, together with h_{min} and h_{max}, the extrema of the function h. Formally, we have: $norm_h(b) = (b - h_{min})/(h_{max} - h_{min})$. For the sum of the importance scores, we perform a normalization of the importance scores. In this context, the extrema used are the minimum and maximum importance score reachable in a scenario. To keep the sum of the importance scores of M states selected among N in the range [0,1], the following formula is used: $\frac{1}{M} \sum_{s \in S^M} norm_I(I(s))$, where S^M is the set of M selected states.

Lemma 1 *The complexity of the normalization of the sum of importance scores is in O(N)*

proof : It is the complexity of finding the maximal and minimal importance scores. □ □

Lemma 2 *The complexity of calculating the normalization factor for the spread-regularity term is in O(1).*

proof : To normalize the sum of the squares of the gaps $i_{k+1} - i_k$, we have to determine the extrema as a function of M and N. With $\sigma = (s_1, a_1, s_2, a_2, \ldots, a_{N-1}, s_N)$ a length-N scenario, these extrema are attained for a sample of $M \leq N$ states $s_{i_1}, \ldots, s_{i_M}$ such that the following equation is maximised/minimised:

$$\sum_{k=1}^{M-1} (i_{k+1} - i_k)^2$$

$$\text{s.t.} \quad i_1 = 1, i_M = N \text{ and } i_k < i_{k+1} \text{ for } k \in [1, .., M-1]$$

By elementary methods, we find that the maximum value is: $(M-2)(1)^2 + (N-M+1)^2$. With $v = (N-1)/(M-1)$ and $w = (v - \lfloor v \rfloor) \times (M-1)$, the minimum value is: $w \times (\lceil v \rceil)^2 + (M-1-w) \times (\lfloor v \rfloor)^2$. Therefore, the complexity of this calculation is in $O(1)$. □ □

Proposition 3 *The complexity of solving the summary problem is in $O(MN^2)$*

proof : Based on Definition 5, and lemmas 1 and 2, it is easy to see that the complexity of solving the summary problem is in $O(MN^2)$. Indeed, with the negligible complexity of the normalization step, the complexity depends on the complexity of the dynamic programming algorithm. □

It is important to specify that the normalizations mentioned above do not impact this complexity (the extraction of the minimal and maximal importance scores being performed before the use of the dynamic programming method). Note that the summary problem would be more difficult with a different objective function. Indeed, minimizing redundancy between all pairs of states would make the summary problem NP-hard, as observed by McDonald [18].

In several different experiments, we noticed that scenarios can contain loops, i.e. repeating subsequences of state-actions. This inevitably leads to a redundancy in the explanation. So, before producing

a summary of a scenario, we perform a pre-processing step to detect such loops in order to display to the user only one iteration. It is interesting to note that when the agent is in a loop, it can never get out of it if the environment response is deterministic, as stated in the following proposition:

Proposition 4 *Given a deterministic policy of the agent π, a deterministic response of the environment π_e, consider a length-N scenario $\sigma = s_0, a_0, s_1, \ldots, s_{N-1}, a_{N-1}, s_N$, corresponding to π and π_e. Suppose that $s_j = s_k$ where $0 \leq j < k \leq N$ and that (j,k) is the lexicographically smallest pair with this property. Then σ is a periodic sequence from rank j.*

proof : By design, the agent's choice and environment response are deterministic, hence the produced scenario σ is also deterministic. A loop starting at time-step j, always repeats. Thus, this ensures the sequence periodicity of σ from rank j. □ □

As a reminder, with SXp we are interested in explaining deterministic policies. Moreover, in the case of FE-scenarios and HE-scenarios, the environment policy π_e is deterministic. The proposition therefore holds for them. For P-scenarios, the response of the environment depends on its transition function p.

Corollary 1 *Let σ^j be a loop in a P-scenario σ_P, and $tr_b(s,a)$ be the highest environment transition probability from (s,a). Proposition 4 holds for a P-scenario iff :*

$$\forall (s,a) \in \sigma, \quad card(\{s' \in S \text{ and } p(s'|s,a) = tr_b(s,a)\}) = 1$$

where card is the cardinality of a set.

In other words, the existence of a non-deterministic transition in a scenario invalidates Proposition 4.

The method described in Definition 5 thus provides a solution to the selection of M states among a set of size N. In practice, for the generation of SXp summaries, the value of M cannot be fixed beforehand, as the length-N scenario may contain sub-sequences to be summarized (in the case of loop detection). It is therefore necessary to introduce a ratio that defines the number of elements to be extracted. Thus, μ defines the compression ratio, i.e. the number of states to be summarized into a single one. Consequently, with μ fixed, the number of states m to be extracted for any sequence of length n is defined by: $m = \lfloor n/\mu \rfloor$.

The summarizing of a scenario, taking into account the loops, is described in Algorithm 1 where the function $findLoop$ looks for a loop and its starting index in the scenario (it is only performed when Proposition 4 holds), and the function *summarize* returns the elements to display to the user according to a set of states and the compression ratio μ. When Algorithm 1 is executed, depending on the values of N and μ, a length-N scenario is either summarized or simply displayed. If a loop is detected, the scenario is split into two sub-sequences, and each part is summarized according to its length and μ. Finally, states are displayed to the user.

Algorithm 1 Scenario summary

Input: $\sigma = s_0, a_0, s_1, \ldots, s_{N-1}, a_{N-1}, s_N, \quad \mu \in \mathbb{N}$

$states \leftarrow [s_0]$
$loop, j \leftarrow findLoop(\sigma)$
if $loop$ **then**
  if $j > 1$ **then** ▷ Summarize first part of σ
    $states.add(summarize([s_1, \ldots, s_{j-1}], \mu))$
  end if
  $states.add(summarize(loop, \mu))$ ▷ Summarize the loop
else
  $states.add(summarize([s_1, \ldots, s_{n-1}], \mu))$ ▷ Summarize the scenario σ
end if
$states.add(s_n)$
$display(states)$

Proposition 5 *The complexity of summarizing an SXp is in $O(MN^2)$.*

proof : Summarizing an SXp consists of the *summary problem* (Definition 4) and loop detection. Since loop detection can be carried out in $O(N^2)$ time (even $O(N)$ with an appropriate data structure), the complexity of summarizing an SXp is in $O(MN^2)$ by Proposition 3. □ □

Although the linearity of the time complexity of producing explanations is not preserved when the length of the scenario requires a summary, explaining nevertheless remains efficient.

2.3 Metrics

Since SXp is an original approach to provide explanations, we did not find in the literature a way to evaluate them. Therefore, we introduce three scores related to the quality of SXp which we used in a validation phase. Moreover, in order to provide additional information to the user, we introduce a score of how representative the displayed scenarios are. For example, whereas the quality of an HE-scenario measures how good of an approximation it is to the true worst-case scenario, its representativeness measures how likely it is.

2.3.1 SXp's quality

To measure the quality of the SXp produced, we implemented three simple scores to answer the question: *"How good is the generated Scenario-Explanation?"*. Let the function q denote the quality evaluation function of a scenario σ; $q(\sigma)$ can vary depending on the application domain and the quality aspect we choose to measure. By default it is equal to the reward $R(\sigma)$, but may be refined to incorporate other criteria for technical reasons explained later. $q(\sigma_F)$ and $q(\sigma_H)$ are respectively the quality of a FE-scenario σ_F and a HE-scenario σ_H. They are used to measure to what extent the scenario is similar to a best-case or worst-case scenario respectively. The resulting *FE-score/HE-score* is *the proportion of k randomly-generated scenarios that have a not strictly better/worse quality (measured by q) than the FE/HE-scenarios themselves* (hence the score lies in the range $[0,1]$). For the P-scenario, the *P-score* is *the absolute difference between the normalized quality $q(\sigma_P)$ of a P-scenario σ_P and the normalized mean of $q(\sigma)$ of k randomly-generated scenarios* (hence lies again in the range $[0,1]$). Formally, given a FE-scenario σ_F, a HE-scenario σ_H and a P-scenario σ_P from s:

$$\text{FE-score}(\sigma_F) = \frac{card(\{\sigma \in S_s^k \text{ and } q(\sigma) \leq q(\sigma_F)\})}{k}$$

$$\text{HE-score}(\sigma_H) = \frac{card(\{\sigma \in S_s^k \text{ and } q(\sigma) \geq q(\sigma_H)\})}{k}$$

$$\text{P-score}(\sigma_P) = \left| norm_q(q(\sigma_P)) - norm_q(\sum_{\sigma \in S_s^k} q(\sigma)/k) \right|$$

where S_s^k is a set of k randomly-generated scenarios s.t. $\forall \sigma = (s_0, a_0, \ldots, s_N) \in S_s^k$, $s_0 = s$. The closer the HE-score, FE-score of a HE/FE-scenario is to 1, the closer it is respectively to the worst/best-case scenario because no other, among the k scenarios produced, is worse/better. A P-score close to 0 indicates that the P-scenario is a good approximation to an average-case scenario. In each case, the scenarios randomly-generated for comparison are produced using the agent's learnt policy π and the transition function p. As mentioned above, by default, the function q is the last-step reward of a scenario, i.e. $q(\sigma) = R(s_{N-1}, a_{N-1})$.

It is important to specify that these quality scores are only used to validate the SXp, given learnt agent and environment policies. These scores are not calculated for each SXp requested by the user but rather upstream during the validation phase of the explainer. On the contrary, the following metric is computed for each SXp to provide additional information to the user.

2.3.2 SXp's representativeness

The objective of this score is to measure the representativeness of a scenario. For each SXp, it is computed and displayed to the user next to the scenarios. This score aims at answering the question: *"How*

representative is this scenario of all possible scenarios starting at state s and with a horizon of N?". This information is important because it indicates, as an example, which scenario is the most representative at horizon N between the FE-scenario and the HE-scenario and to what extent. If the representativeness score is low for the HE-scenario (resp. FE-scenario), it may reassure (resp. alert) the user because this scenario is not very representative. On the contrary, if its score is high, the HE-scenario (resp. FE-scenario) is representative and it may alert (resp. reassure) the user.

To compute this score, it is first necessary to obtain the probability of the FE-scenario, HE-scenario and P-scenario. The agent's policy being deterministic, the probability of a scenario $\sigma = s_0, a_0, s_1, \ldots, s_{N-1}, a_{N-1}, s_N$, denoted $Pr(\sigma)$, is defined by: $Pr(\sigma) = \prod_{i=1}^{k} p(s_i|s_{i-1}, a_{i-1})$. The P-scenario being an approximation of the most probable scenario, it is considered the most representative. Therefore, the representativeness score is simply the ratio between the probability of a scenario σ and the probability of the P-scenario σ_P.

Definition 6 *Given a P-scenario σ_P, the representativeness score of a scenario σ is:*

$$rep(\sigma, \sigma_P) = \min(1, Pr(\sigma)/Pr(\sigma_P))$$

The score lies in range $[0, 1]$. A score of 1 means that the scenario is at least as representative as the P-scenario and a score close to 0 reflects a scenario with little representativeness. The *min* operation limits the result to 1. If the ratio is higher, it can mean two things. First meaning: the HE/FE-scenario length is smaller than that of the P-scenario, which may make the HE/FE-scenario more likely. This can occur if the agent reaches a terminal state in less time in the HE/FE-scenario than in the P-scenario. Second meaning: the HE/FE-scenario is the same length as the P-scenario but it is a better approximation of the *most-probable* scenario than the P-scenario. This latter case was not encountered during the experiments.

3 Experimental results

The Frozen Lake (FL) and Drone Coverage (DC) problems illustrate, respectively, a single and a multi-agent context. Furthermore, the training process was managed by two distinct algorithms, respectively Q-learning [30] and a Deep-Q-Network [21]. Recall that the algorithm used to train environment-agents is similar to the one used to train the agent. The exploration/exploitation trade-off is achieved by using an ε-greedy action selection where ε is a probability to explore. For short SXp, the hyper-parameter N (scenario-length) is set to 5 and 6 respectively for the FL and DC problems. For long SXp's, it is set respectively to 20 and 50 for the FL and DC problems. The compression ratio μ is set to 5 for both problems. FL experiments were run on an ASUS GL552VX, with 8 GB of RAM and a 2.3GHz quad-core i5-6300HQ processor and DC experiments were carried out using a Nvidia GeForce GTX 1080 TI GPU, with 11 GB of RAM (source code available at: https://github.com/lsaulier/SXp-ICAART23). Each SXp presented in this section (see e.g. Figure 2) is displayed as follows. On the left, there is the starting state, from which the SXp is generated. On the right, there are 3 lines, each representing a scenario, which are in order the FE-scenario, HE-scenario and P-scenario. Next to each scenario is displayed its representativeness score. The explanation quality scores in Tables 1 and 2 are based on $k = 10000$ to reduce the randomness of score calculation. The Avg_i and σ_i columns show the average and standard deviation of explanation scores based on i different states, or configurations (i.e. states of all agents in a multi-agent problem such as DC).

3.1 Frozen Lake (FL)

3.1.1 Description

The FL problem is an episodic RL problem with discrete state and action spaces. The agent (symbolized in Figure 2 by a blue dot) moves in a 2D grid world, representing the surface of a frozen lake, with the aim to reach an item in a specific cell of the grid (marked with a star). There are holes in the frozen lake (symbolized by blue cells in the map) and the others cells are solid ice. When an agent falls into a hole, it loses. The agent's initial state is at the top-left corner cell of the map.

A state is represented by a single value, corresponding to the agent's position in the map, $S = \{1, \ldots, l \times c\}$ with, l, c the map dimensions. For the sake of readability, in the results a state is denoted by the agent's

coordinates *(line,column)*, where $(1,1)$ is the top left cell and $(4,4)$ is the bottom right cell which are respectively the initial state of the agent and its goal on the 4×4 map in Figure 2. The action space is $A = \{left, down, right, up\}$. The reward function is sparse and described as follows: for $s \in S, a \in A$, s' denoting the state reached by performing action a from s, and s^g being the goal state:

$$R(s,a) = \begin{cases} 1, & \text{if } s' = s^g. \\ 0, & \text{otherwise.} \end{cases}$$

The transition function p is the same from any state. Because of the slippery nature of the frozen lake, if the agent chooses a direction (e.g. *down*), it has 1/3 probability to go on this direction and 1/3 to go towards each remaining direction except the opposite one (here, 1/3 to go *left* and 1/3 to go *right*).

To solve this RL problem, we use the tabular Q-learning method because the state and action space is small. The end of an episode during training is characterized by the agent reaching its goal or falling into a hole.

As stated in the proof of Proposition 2, an environment-agent's state contains an extra piece of information compared to an agent's state: the action executed by the agent from this position, according to its policy π. As the environment-agent reflects the transitions of the environment, there are only 3 actions available and they depend on the agent's choice of action. The reward function of the favorable agent is similar to the agent's reward function. The hostile agent receives a reward of 1 when the agent falls into a hole, of -1 if the agent reaches its goal and a reward of 0 otherwise.

3.1.2 Results

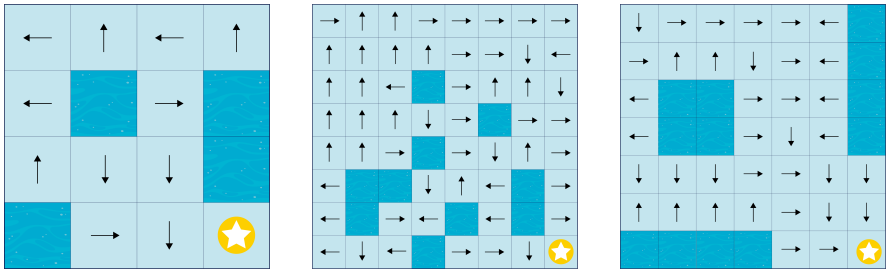

Figure 1: Agent's learnt policies for the 4×4 and 8×8 maps and a safe 7×7 grid [23].

Short SXp In order to test our approximate SXp on different environment sizes, we used a 4×4 map and a 8×8 map, the ones presented in OpenAI Gym [3]. Since the reward is sparse (0 except in goal states), FE/HE/P-scores computed purely with $q(\sigma) = R(s_{N-1}, a_{N-1})$ are uninformative (when the number of steps N is not large enough to reach the goal). Accordingly, the quality evaluation function was defined as follow: $q(\sigma) = R(\sigma) + \lambda Q(\sigma)$, where $Q(\sigma) = \max_{a_N \in A} Q(s_N, a_N)$ is the maximum last-step Q-value, $R(\sigma) = R(s_{N-1}, a_{N-1})$ is the reward of scenario σ and $\lambda < 1$ is a positive constant. Another particularity of this problem, is that since the transitions are equiprobable, many P-scenarios are possible.

The agent's learnt policy for the 4×4 map is represented in Figure 1. Each arrow represents the action performed by the agent from this state. We note that the agent learns to avoid to enter the top-right part of the map (i.e. the two first lines without the first column), which is the most dangerous part, due to the $(2,3)$ state. In the remaining parts of the map, the only dangerous state is $(3,3)$ since the agent action choice is *down*, so it has a probability of $\frac{1}{3}$ to fall into a hole.

The SXp calculated starting from the state $(2,1)$ is shown in Figure 2. The P-scenario is one scenario among many, and it highlights the difficulty for the agent to succeed in this particular grid with a few steps. The hostile agent exploits well its only way to *force* the agent to fall into a hole given the agent's policy (Figure 1) which is to push it towards the hole located at (3, 4). The favorable agent also learns well and provides an FE-scenario where the agent reaches its goal in the minimum number of steps. The *HE-score* and *FE-score* of SXp's from state $(2,1)$ are presented in Table 1. These are perfect scores (equal to 1).

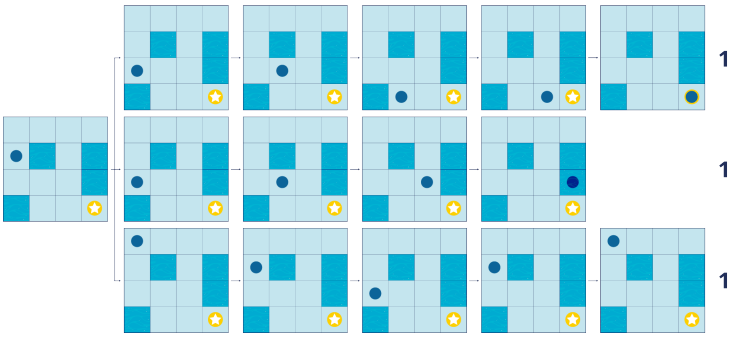

Figure 2: Scenario-Explanations (from top to bottom: FE-scenario, HE-scenario, P-scenario) from a specific state in the 4×4 map [23].

Table 1: Quality scores for Scenario-Explanation in the 4×4 map and 8×8 map [23].

	4x4 map			8x8 map		
	State $(2,1)$	Avg_7	σ_7	State $(3,6)$	Avg_{20}	σ_{20}
FE-score	1	1	0	1	0.824	0.281
HE-score	1	1	0	1	0.828	0.346
P-score	0.081	0.211	0.115	0.031	0.08	0.09

Moreover, since the *P-score* is close to 0, the provided P-scenario is a good approximation. We computed SXp's based on the same agent's policy π but starting from 7 reachable states, i.e. states that can be reached following the policy π (Figure 1) and which are neither holes nor the goal. Results are reported in the Avg_7 column of Table 1. Hostile and favorable agents learnt perfectly.

The SXp method was also tested with an 8×8 map. As we can see in Figure 1, the agent has learned to avoid as much as possible the left zone of the map which is dangerous. Figure 3 depicts an SXp starting from the state $(3,6)$. Due to the agent's policy, the hostile agent can't just push the agent down from state $(3,6)$, but it manages to push the agent along a path which ensures that the agent falls into a hole, hence the HE-scenario ends after only 3 steps. In the FE-scenario, the favorable agent brings the agent closer to its goal over the $N = 5$ time-steps. The P-scenario again provides evidence that the agent is likely not to succeed in this difficult environment in a small number of steps.

From the scores presented in Table 1 concerning the 8×8 map, we can again conclude that the 3 produced scenarios are of good quality. The scores presented in the Avg_{20} column were obtained by SXp from 20 randomly-chosen starting states. The average score is lower than 1 but note that 1 is achieved

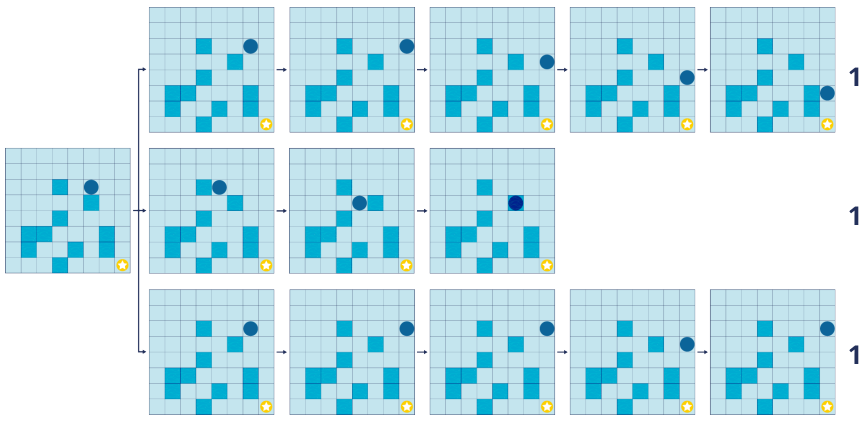

Figure 3: Scenario-Explanations from a specific state in the 8×8 map [23].

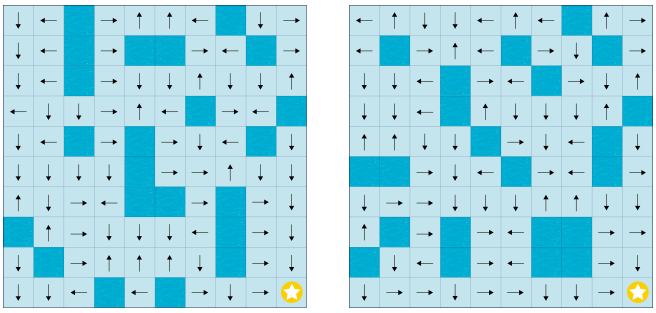

Figure 4: Agent's learnt policies for two 10×10 maps (respectively named map A and map B).

for respectively more than 75% and 60% of HE-scenarios and FE-scenarios. Hence, apart from some randomly-generated starting states located in the little explored left-zone of the map, the scores indicate that HE/FE-scenarios are good approximations of worst/best scenarios.

The representativeness scores are similar for each scenario in Figure 2 and Figure 3. This is due to the fact that each transition is equiprobable in the FL problem. In addition, the time horizon of the displayed HE/FE-scenarios do not exceed that of the P-scenarios. This is why all scenarios are equally representative.

In order to check the impact of the agent's policy on the environment-agents' learning process, we designed a 7×7 map, shown in Figure 1, in such a way that if the agent learns well, it can avoid falling into a hole. Once the learning phase is over, we noticed that the hostile agent learns nothing. Since the agent learns an optimal policy π, the hostile agent can't push the agent into a hole. Accordingly, it can't receive any positive reward and therefore can't learn state-action values. *This is strong evidence that the agent's policy is good.*

Long SXp The summarized SXp's were tested on two maps of size 10×10 to show the usefulness of summaries with a relatively large scenario-length. The maps and their policies are shown in Figure 4 (named map A and map B respectively). The agents for the 2 maps have learned well to reach the objective, avoiding as much as possible to fall into the holes. However, there are still many opportunities for the hostile agent to make the agents fall. The scenario length is large ($N = 20$) to have a long term view of what can happen from a given state. λ is set to 1.0 for the calculation of the objective function f_N of the summary problem (Definition 4). This means that we consider as equally important the importance of the states and the distribution of the displayed states. Figure 5 and 6 show summarized SXp's for respectively map A and B beginning in the agent's initial state. Each number shown between states (such as 4, 4, 5, 1 in the FE-scenario of Figure 5) represent the number of omitted states.

In Figure 5, the hostile agent succeeds in causing the agent to fall into the nearest reachable hole from its starting point. The HE-scenario summary gives a good idea of what happened. It is more difficult to understand what happened in the P-scenario by looking at its summary. This is due to the problem difficulty of the agent reaching the goal; non-deterministic transitions can cause the agent to stagnate in a region of the map for a long time. The favorable agent is successful in helping the agent achieve the objective and the FE-scenario summary is of good quality. Indeed, the states are sufficiently spread out for the user to understand how the scenario unfolds. In the FE-scenario, the last states displayed are relatively close in terms of distance. Importance scores for states near the goal are higher than those far from it; as a result, the summary is impacted. These remarks are also relevant for the SXp summarized in Figure 6. Similarly to the presented short SXp, the scenarios are equally representative.

We now focus on the FE-scenario in Figure 5 and vary the value of λ to observe its impact on the scenario summary provided. The different summarized FE-scenarios are displayed in Figure 7. The values of λ are respectively 1, 0.5 and 0. With $\lambda = 0$, the uniformity of the gaps between the selected states is not taken into account. Note that the associated summary only displays the last elements, because they are the most decisive for achieving the objective. The problem is that we have no idea about what happened during 14 time-steps. Accordingly, this summary is not informative enough. This justifies taking into account the

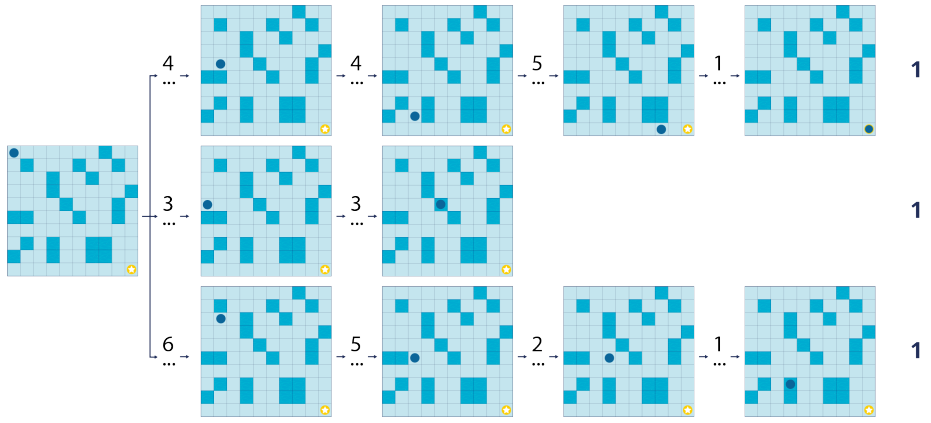

Figure 5: Scenario-Explanations from a specific state in map A.

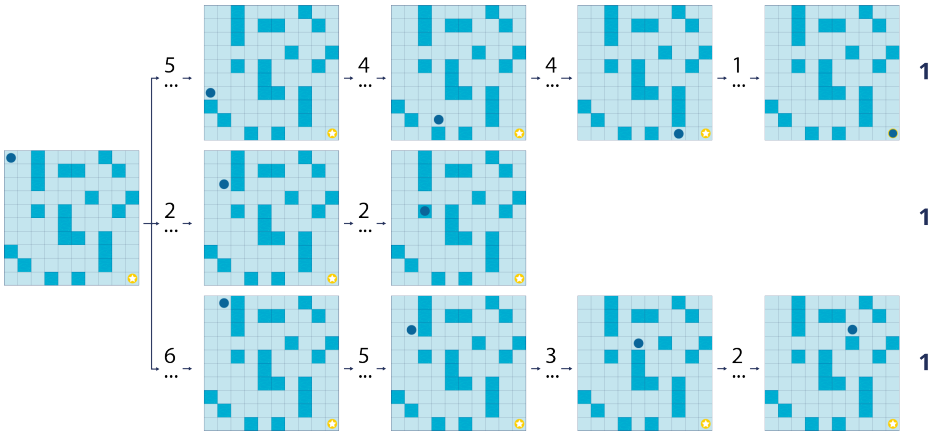

Figure 6: Scenario-Explanations from a specific state in map B.

uniformity of the distribution of selected states in selecting SXp summaries. The other two summaries provided are of good quality, with a more evenly spread summary for the summary provided with $\lambda = 1$.

In the experiments, no loops were detected for the FE-scenarios and HE-scenarios. This makes sense since both the hostile and favorable agents must bring the agent into specific states to obtain a reward. In other words, looping between different states (i.e. positions on the map) does not bring any reward, which is why there are no loops in FE-scenarios/HE-scenarios (assuming they have learned well). The P-scenarios have probability *zero* of containing infinite loops. This is because the environment response is non-deterministic. Whatever the state s and the action a chosen from s, transitions are non-deterministic. Therefore, Proposition 4 does not hold in this problem for P-scenarios.

3.2 Drone Coverage (DC)

3.2.1 Description

The DC problem is a novel multi-agent, episodic RL problem with discrete state and action spaces. The agents' goal is to cover the largest area in a windy 2D grid-world containing trees (symbolized by a green triangle in Figure 8). The coverage of each drone (represented as a dot) is a 3×3 square centered on its position. A drone is considered as lost and indeed disappears from the grid if it crashes into a tree or another drone.

A state for an agent is composed of the contents of its neighbourhood (a 5×5 matrix centered on the agent's position) together with its position on the map. The action space is $A = \{left, down, right, up, stop\}$.

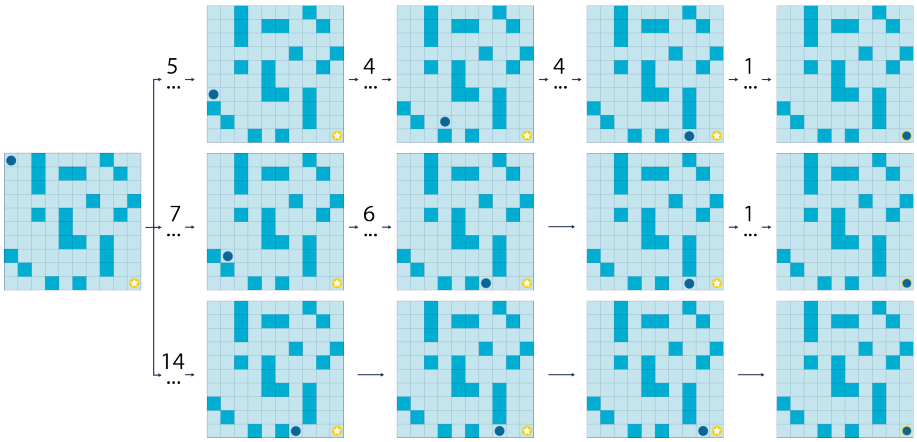

Figure 7: Summarized FE-scenarios from a specific state in map A.

The reward function R of an agent is impacted by its coverage, its neighbourhood, and whether it has crashed or not (the reward is -3 in case of *crash*): if there is no tree or other drone in the agent's 3×3 coverage, it receives a reward (called *cover*) of $+3$ and $+0.25 \times c$ otherwise, where c is the number of free cells (i.e. with no tree or drone) in its coverage; the agent receives a *penalty* of -1 per drone in its 5×5 neighbourhood (since this implies overlapping coverage of the two drones). With s' the state reached by executing action a from s, the reward function is as follows: for $s \in S, a \in A$,

$$R(s,a) = \begin{cases} -3, & \text{if } crash \\ cover(s') + penalty(s'), & \text{otherwise} \end{cases}$$

As there are 4 drones, the maximum cumulative reward (where cumulative reward means the sum of all agents rewards in a given configuration) is 12 and the minimum is -12. The transition function p, which represents the wind, is similar in each position and is given by the following distribution: $[0.1, 0.2, 0.4, 0.3]$. This distribution defines the probability that the wind pushes the agent respectively *left*, *down*, *right*, *up*. After an agent's action, it moves to another position and then is impacted by the wind. As an additional rule, if an agent and wind directions are opposite, the agent *stays in its new position*, so the wind has no effect. After a *stop* action, the drone does not move and hence is not affected by the wind.

In order to train the agents, we used the first version of Deep-Q Networks [21] combined with the Double Q-learning extension [12]. The choice of this algorithm was motivated by two factors. First, we wanted to investigate our XRL method's ability to generalise to RL algorithms, such as neural network based methods, used when the number of states is too large to be represented in a table. Secondly, this setting enables us to deal with a problem which is scalable in the number of drones and grid size.

The end of an episode of the training process occurs when either one agent crashes, or a time horizon is reached. This time horizon is a hyper-parameter fixed before the training; it was set to 22 for the training of the policy which is explained in the following subsection. When restarting an episode, the agents' positions are randomly chosen. This DC problem is a multi-agent problem, and to solve it, we use a naive approach without any cooperation between agents. Only one Deep-Q Network is trained with experiences from all agents. The reward an agent receives is only its own reward; we do not use a joint multi-agent reward.

Concerning the implementation of the hostile and favorable environment: the extra information in the environment-agent's state is the action performed by the agent in its corresponding state. Actions are similar to the agents' except that there is no *stop* action. The favorable-agent reward function is similar to the one of the agent's and the reward function of the hostile agent is exactly the opposite.

3.2.2 Results

Short SXp For the sake of simplicity, each drone has an associated color in Figure 8. Above each map, there is a list of colored arrows, or stop symbols, corresponding to each colored drone's action which leads them to the configuration displayed in the map. A colored cell means that the area is covered by the drone

Figure 8: Scenario-Explanation of a specific configuration of the DC problem [23].

of the same color and a dark grey cell indicates an overlap of the coverage of different drones. To compute the SXp quality scores, we use $q(\sigma) = \sum_i R_i(s_{N-1}, a_{N-1})$ with R_i denoting the reward of agent i (i.e. q is the last-step aggregate reward). Note that the policy to be explained is good, but not optimal. Measuring the performance of a policy by the average of the cumulative rewards obtained at the end of the last hundred training episodes, the performance is 11.69 (out of 12).

The SXp for a particular configuration, denoted as *configuration A*, is shown in Figure 8. The hostile agent succeeds in crashing two drones and positioning the remaining drones in bad covering positions. The P-scenario demonstrates well the most probable transition (the wind pushes the drones to the *right*) and the favorable agent manages to reach a perfect configuration in only 5 steps. The representativeness scores reassure the user for this SXp. Indeed, the catastrophic HE-scenario is not representative at all while the successful FE-scenario is. SXp quality scores are given in Table 2 where the last columns show the average and standard deviation of scores obtained from 30 random configurations. These results indicate good approximate SXp's. Moreover experiments showed that 19 *HE-scores* are higher than 0.95 and 24 *FE-scores* are perfect (equal to 1). The quality of the learnt policy is also attested by the fact that a maximum reward is attained in 21 out of 30 P-scenarios.

Table 2: Quality cores for Scenario-Explanation in 10×10 map, showing the mean and standard deviation over 30 trials [23].

	Configuration A	Avg_{30}	σ_{30}
FE-score	1	0.919	0.198
HE-score	1	0.936	0.08
P-score	0.073	0.034	0.04

Long SXp The DC problem is a multi-agent one. This means that a new *configuration importance score* must be expressed to summarize SXp. Since a configuration is the aggregation of each drone's state, the *configuration importance* is simply the average importance score of drone states. With C denoting the set of states of a configuration, the *configuration importance score* I_{config} is defined by:

$$I_{\text{config}}(C) = \frac{1}{card(C)} \sum_{s \in C} I(s)$$

An example of an SXp summary based on the same policies as the short SXp is shown in Figure 9. λ is set to 1.0 for the computation of the objective function f_N of the *summary problem* (Definition 4). Scenario-length is voluntarily large because, during the experiments, we noticed that the drones looped quite quickly between one or several states. Therefore, loop handling is useful for SXp. Moreover, Proposition 4 holds

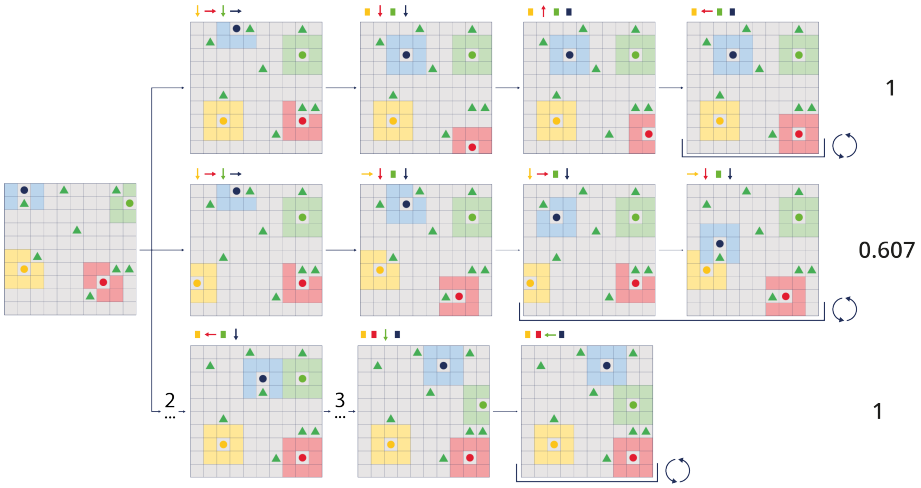

Figure 9: A long Scenario-Explanation of a specific configuration of the DC problem.

for P-scenarios since the environment response is deterministic. Indeed, at each time step, whatever the state is, there is only one most probable transition (push the agent to the *right*).

In the long SXp shown in Figure 9, each scenario ends with a loop (of length 1, 2 and 1, respectively). The hostile agent definitively prevents 3 out of 4 drones from having a perfect coverage by 'blocking' them in a loop of two configurations. The P-scenario shows a scenario where drones quickly reach a perfect configuration that they maintain. The summary of the first part of the P-scenario allows the user to deduce the drones' movements (despite the compression ratio of $\mu = 3$). The FE-scenario presents a scenario where drones reach a perfect configuration faster. For the FE/HE-scenario, there is no need to summarize the part before the loop because there are few states. In each scenario, loop management allows us to keep SXps both concise and easily comprehensible.

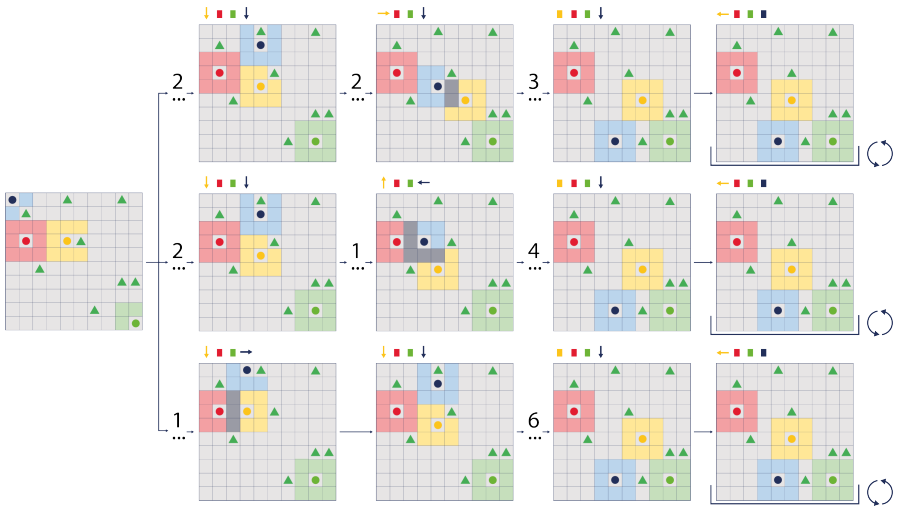

Figure 10: Summarized FE-scenarios from a specific configuration of the DC problem.

We now focus on the FE-scenario and vary the value of λ in order to observe its impact on the scenario summary. λ takes a value among 0, 0.5 and 1 and the compression ratio μ is set to 3. In Figure 10, the summary only has an impact on the first part of the scenario. The different scenarios are of good quality, except for the third one in which the gap between the configurations is not taken into account ($\lambda = 0$). In this summarized FE-scenario, only the first states, considered important, are displayed. Therefore, it is more difficult to understand how the drones reached the loop configuration. Again, these experiments

confirm that the default value $\lambda = 1$ is a sensible choice.

Results obtained in the FL and DC problems show that, whether the agent's policy is optimal or not, we can obtain interesting information via our SXp. Furthermore, this XRL method does not increase asymptotic complexity of RL. The user can choose to display short-term scenarios or long-term scenarios. These latter are summarized by a combination of loop detection and a dynamic programming algorithm to find the best balance between the importance of states and their uniform spread over the scenario.

4 Related work

XRL methods use different key features of Reinforcement Learning to provide explanations. As an example, we can cite the interpretable reward proposed in [15]. Using exclusively agent's states which are images, Greydanus *et al.* present a method to produce saliency maps for Atari agents [10]. By adding object recognition processing, Iyer *et al.* produce object saliency maps from states to gain more insights about the agent's decisions [14]. Olson et al. and Hubert et al. answer the question 'Why perform action a rather than action b from the state s?' in a counterfactual fashion [22, 13]. To do so, a state s' close to s is generated using a GAN, which leads to the choice of action b. In order to focus on causal relationship between action and state variables, Madumal *et al.* [17] build an action influence model (AIM) used for explanation. Yu et al. extend the use of AIM to problems with continuous action space, without the need of prior knowledge of the environment causal structure [32]. Additional information can be collected during the agent's training process, including XRL methods [5] which extract success probabilities and number of transitions, or methods which learn a belief map [31]. All these XRL methods allow one to essentially explain the choice of an action in a specific state. For policy-level explanations, EDGE highlights the most critical time-steps, states, given the agent's final reward in an episode [11]. Policy explanations can be used to improve the policy. As an example, the ReCCoVER algorithm shows to developers the spurious correlations between critical states features which impact the generalisation of the agent's policy and proposes a revision [9].

Several works aim at making the policy interpretable, i.e. making it understandable for the user. A fairly common approach consists in training a black-box policy represented by a NN and approximating it, or even outperforming it, with an interpretable model also called a *surrogate model*. The VIPER algorithm results in verifiable policies in the form of decision trees [2]. With the PIRL framework, the NDPS algorithm allows to check a policy represented by an intuitive and understandable programming language [29]. Danesh et al. transform RNNs into compact Moore machines [6]. In a multi-task RL context, Shu et al. propose to represent each learned policy for a sub-task as a human language description [25]. Thus, the global policy is directly interpretable. Rather than obtaining an interpretable policy by design, some works add an interpretability layer to black-box policies. In this sense, Zahavy et al. interpret policies represented by DQN's in a post hoc manner, by highlighting the hierarchical state aggregation, as well as the policy's strengths and weaknesses [33]. Another post-hoc explainer tool is PolicyExplainer which is a visual analytic interface that provides a set of tools to understand the policy [20].

An alternative way to explain the agent's policy is through state-action sequences, like our SXp. One part of the framework proposed by Sequeira and Gervasio provides a visual summary, based on sequences obtained during the learning phase, to globally explain the policy [24]. With the same goal, HIGHLIGHTS extracts sequences based on a notion of state importance to provide a summary of the agent's learnt behaviour [1]. In a context of MDP, the method implemented in [27] computes sequences that differ in at most n actions from the sequence to explain, as counterfactual explanations. Explaining a sequence in a contrastive way, is achieved in [28] by producing a contrastive policy from the user question and then comparing both sequences. These XRL methods do not solve the same problem as our SXp. Indeed, [24] and [1] provide high-level policy explanation through summaries in a general context of the agent's interaction with the environment. [27] and [28] explain the policy in a counterfactual way; the problem is to generate a sequence in which actions differ from π. Thus, these approaches are incomparable with our SXp, which explain the policy from a particular state, by producing scenarios using the policy π.

5 Discussion

The experiments illustrate the different possible uses of SXp. Apart from understanding policies, SXp also provide a way to evaluate them. Indeed, even if the policy π learnt is not optimal, HE-scenarios and FE-scenarios provide useful information. If from multiple starting states and by varying the scenario-length N, an FE agent cannot bring the agent closer to its goal, this is a proof that the policy π is inadequate. Conversely, an HE agent which cannot prevent the agent from reaching its goal is a evidence of a good policy π. Concretely, in the FL problem this means that the agent has learnt not to give a hostile environment the opportunity to force it to fall into a hole, and in the DC problem the agent has learnt to stay sufficiently far away from trees and other drones. To ensure that the scenario-length is not impacting the results, a more complete study could be carried out by increasing N and using summarized SXp in order to examine the generated scenarios. In other words, our XRL method can also be used as a *debugging tool*.

The experiments have also taught us some valuable lessons. Since we use the same RL method and the same resources to learn π_e as were used to learn π (in order not to increase asymptotic time and space complexity), we cannot expect the quality of explanations to be better than the quality of the original policy π. For example, when states are represented by a simple index in a table, as in Q-learning, π_e can provide no useful information concerning states which were not visited during the learning of π_e. Indeed, whatever the RL method used, since π_e is learnt after (and as a function of) the agent's policy π, the latter will be of better quality on (states similar to) states visited more frequently when following the agent's policy π. A higher/lower quality of explanation for those states that are more/less likely to be visited is something the user should be aware of. If it is important that quality of explanations should be independent of the probability of a state, then the training phase of π_e should be adapted accordingly. This is an avenue of future research.

We should point out the limitations of our method. The three scenarios which are produced are only approximations to the worst-case, best-case and most-probable scenarios. Unfortunately, approximation is necessary due to computational complexity considerations, as highlighted by Proposition 1. We should also point out that the distinction between these three scenarios only makes sense in the context of RL problems with a stochastic transition function. Moreover, the transition function must be known or approximated using model-based methods and SXp summaries require Q-values for the *importance score* computation. Finally, due to the relative novelty of the notion of scenario-explanation, no metric was found in the literature to evaluate the quality of SXp's.

6 Conclusion

In this paper, we describe an RL-specific explanation method based on the concept of transition in Reinforcement Learning. To the best of our knowledge, SXp is an original approach for providing predictive explanations. This predictive XRL method explains the agent's deterministic policy through scenarios starting from a certain state. Moreover, SXp is agnostic concerning RL algorithms and can be applied to all RL problems with a stochastic transition function. An SXp is composed of 3 scenarios: HE-scenario, FE-scenario and P-scenario. They respectively give an approximation of a worst-case scenario-explanation, a best-case scenario-explanation and a most-probable scenario explanation. Experiments indicate that these approximations are informative according to SXp quality scores. In the FL and DC problems, SXp provides a good answer to the question *"What is likely to happen from the state s with the current policy of the agent?"*. Of course, in any new application, experimental trials would be required to validate this XRL method and evaluate its usefulness. Our 3-scenario-based method appears promising and can be used in more complex problems: we only require that it is possible to learn policies for hostile/favorable environments.

SXp's are parameterized by the length N of the scenarios generated. This parameter must be entered by the user in order to look at the short- or long-term future of the agent's interactions in the environment. The greater the N, the longer the scenarios, and therefore the more long-term the vision. A large N results in a high cognitive load for the user. To eliminate this cognitive overload, we have introduced summarized SXp. It summarizes HE-scenario, FE-scenario and P-scenario according to two criteria: the importance of a state and its position in the scenario. We consider that an optimal summary should display important

states that are well distributed along the scenario. We have developed a dynamic programming solution to generate these summaries. We use the parameter λ to modulate the importance of the spread of states displayed in the summary produced. Summarized SXp provides a concise and effective way of seeing what can happen in the long term.

An avenue of future work would be to focus on the probabilistic aspect of stochastic policies and provide specific approximate SXp definitions. Another avenue of future work would be to provide SXp's for 2-person games (e.g. Connect 4) where the presence of an opponent player makes the environment stochastic. In this context, the environment policies would simply be opposing player policies. The hostile environment can be seen as a player who is good and the favorable environment as a bad player who gives the agent plenty of opportunities. To provide the P-scenario, one would have to determine the likely behavior of an average opponent. Another idea would be to provide the user with more tools to understand the agent's policy in detail. This could be achieved by learning several environment policies, based on more refined properties than the agent's success or failure in achieving its goal, such as the risk or safety of the agent's policy. An avenue of future research would be to study possible theoretical guarantees of SXp performance.

In a nutshell, after introducing a theoretical framework for studying predictive explanations in RL, we presented a novel practical predictive-explanation method. A pleasing aspect of our method is that explanation employs the same tools as the original reinforcement learning method. In addition, when scenarios are too long to be displayed to the user, we provide a dynamic programming solution that enables us to generate scenario summaries in which a compromise is made between displaying the most important states and evenly covering the scenario steps.

Acknowledgements

The authors would like to thank Arnaud Lequen for his valuable suggestions that have led to the improvement of this paper. This work was supported by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future – PIA3" under grant agreement no. ANR-19-PI3A-0004.

References

[1] Dan Amir and Ofra Amir. HIGHLIGHTS: summarizing agent behavior to people. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS*, pages 1168–1176. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2018.

[2] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *NeurIPS*, pages 2499–2509, 2018.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] Jeffery Allen Clouse. *On integrating apprentice learning and reinforcement learning*. University of Massachusetts Amherst, 1996.

[5] Francisco Cruz, Richard Dazeley, and Peter Vamplew. Memory-based explainable reinforcement learning. In Jixue Liu and James Bailey, editors, *AI 2019: Advances in Artificial Intelligence - 32nd Australasian Joint Conference*, volume 11919 of *Lecture Notes in Computer Science*, pages 66–77. Springer, 2019.

[6] Mohamad H. Danesh, Anurag Koul, Alan Fern, and Saeed Khorram. Re-understanding finite-state representations of recurrent policy networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2388–2397. PMLR, 2021.

[7] Adnan Darwiche. Human-level intelligence or animal-like abilities? *Commun. ACM*, 61(10):56–67, 2018.

[8] European Commission. Artificial Intelligence Act, 2021.

[9] Jasmina Gajcin and Ivana Dusparic. ReCCoVER: Detecting causal confusion for explainable reinforcement learning. In Davide Calvaresi, Amro Najjar, Michael Winikoff, and Kary Främling, editors, *Explainable and Transparent AI and Multi-Agent Systems - 4th International Workshop, EXTRAAMAS 2022, Revised Selected Papers*, volume 13283 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2022.

[10] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding Atari agents. In Jennifer G. Dy and Andreas Krause, editors, *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1787–1796. PMLR, 2018.

[11] Wenbo Guo, Xian Wu, Usmann Khan, and Xinyu Xing. EDGE: explaining deep reinforcement learning policies. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *NeurIPS*, pages 12222–12236, 2021.

[12] Hado Hasselt. Double Q-learning. *Advances in neural information processing systems*, 23, 2010.

[13] Tobias Huber, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André. GANterfactual-RL: Understanding reinforcement learning agents' strategies through visual counterfactual explanations. *CoRR*, abs/2302.12689, 2023.

[14] Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia P. Sycara. Transparency and explanation in deep reinforcement learning neural networks. In Jason Furman, Gary E. Marchant, Huw Price, and Francesca Rossi, editors, *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, AIES*, pages 144–150. ACM, 2018.

[15] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI workshop on explainable artificial intelligence*, page 7, 2019.

[16] Zachary C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, 2018.

[17] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 2493–2500. AAAI Press, 2020.

[18] Ryan McDonald. A study of global inference algorithms in multi-document summarization. In *Advances in Information Retrieval: 29th European Conference on IR Research, ECIR 2007, Proceedings 29*, pages 557–564. Springer, 2007.

[19] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. A survey of explainable reinforcement learning. *CoRR*, abs/2202.08434, 2022.

[20] Aditi Mishra, Utkarsh Soni, Jinbin Huang, and Chris Bryan. Why? Why not? When? Visual explanations of agent behavior in reinforcement learning. *CoRR*, abs/2104.02818, 2021.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[22] Matthew L. Olson, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. Counterfactual states for Atari agents via generative deep learning. *CoRR*, abs/1909.12969, 2019.

[23] Léo Saulières, Martin C. Cooper, and Florence Dupin de Saint Cyr. Reinforcement learning explained via reinforcement learning: Towards explainable policies through predictive explanation. In *15th International Conference on Agents and Artificial Intelligence (ICAART 2023)*, pages 35–44, 2023.

[24] Pedro Sequeira and Melinda T. Gervasio. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artif. Intell.*, 288:103367, 2020.

[25] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. *CoRR*, abs/1712.07294, 2017.

[26] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[27] Stratis Tsirtsis, Abir De, and Manuel Rodriguez. Counterfactual explanations in sequential decision making under uncertainty. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *NeurIPS 2021*, pages 30127–30139, 2021.

[28] Jasper van der Waa, Jurriaan van Diggelen, Karel van den Bosch, and Mark A. Neerincx. Contrastive explanations for reinforcement learning in terms of expected consequences. *CoRR*, abs/1807.08706, 2018.

[29] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5052–5061. PMLR, 2018.

[30] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[31] Herman Yau, Chris Russell, and Simon Hadfield. What did you think would happen? Explaining agent behaviour through intended outcomes. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *NeurIPS*, 2020.

[32] Zhongwei Yu, Jingqing Ruan, and Dengpeng Xing. Explainable reinforcement learning via a causal world model. *CoRR*, abs/2305.02749, 2023.

[33] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding DQNs. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1899–1908. JMLR.org, 2016.