# Learning on a Budget Using Distributional RL

Jonathan Serrano-Cuevas*
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
Jonathan.Serrano@cwi.nl

Eduardo F. Morales
INAOE
Puebla, Mexico
emorales@inaoep.mx

Pablo Hernandez-Leal
Daan Bloembergen
Michael Kaisers
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
{Pablo.Hernandez,D.Bloembergen,
Kaisers}@cwi.nl

## ABSTRACT

Agents acting in real-world scenarios often have constraints such as finite budgets or daily job performance targets. While repeated (episodic) tasks can be solved with existing reinforcement learning algorithms, methods need to be extended if the repetition depends on performance. Recent work has introduced a distributional perspective on reinforcement learning, providing a model of episodic returns. Inspired by these results we contribute the new *budget- and risk-aware distributional reinforcement learning* (BRAD-RL) algorithm that bootstraps from the C51 distributional output and then uses value iteration to estimate the value of starting an episode with a certain amount of budget. With this strategy we can make budget-wise action selection within each episode and maximize the return across episodes. Experiments in a grid-world domain highlight the benefits of our algorithm, maximizing discounted future returns when low cumulative performance may terminate repetition.

## KEYWORDS

Distributional RL; budget aware; risk safe

## 1 INTRODUCTION

The concept of Safe Reinforcement Learning (SRL) has been recently gaining interest in fields such as business, finance or robotics. SRL can be defined as the process of learning policies that maximize the expected total return in problems where it is important to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment processes [5]. In this paper we are interested in the reasonable system performance issue, particularly in *episodic and risky* tasks, where the risk comes in the form of stochastic returns following a certain distribution and when the agent has limited resources or *budget*. An example of this setting is a day trader working for a large bank: he has to analyze historical information to design a path of action to invest a certain amount of money, targeting to obtain a profit at the end of the day. This means that as long as there are positive results by the afternoon he can afford to have a negative balance at some time of the day, perhaps because of unrealized losses [1]. In such scenarios often the actions that lead to the maximum value in expectation are also the riskiest and could deplete the agent's budget. As an example assume an agent in state $s_0$ that has to decide between two actions $a_s$ and $a_r$ with expected rewards of $\mathbb{E}[r|s = s_0, a = a_s] = 2$ and $\mathbb{E}[r|s = s_0, a = a_r] = 3$. With only this information, the common choice, would be to take $a_r$ because it maximizes the expected reward. Notwithstanding, $\mathbb{E}[r|s = s_0, a = a_r]$ is given by a stochastic distribution of -2 and 6 in contrast to $\mathbb{E}[r|s = s_0, a = a_s]$ which is governed by a deterministic value. In light of this information, i.e., the *value distribution*, different strategies could be considered. For example, a *safe* strategy will select only the action that yields a positive deterministic value, and a *risky* strategy will select the action with a larger expected value, even if there exists the probability of receiving a negative reward. Following the previous example we can see that there are issues if the expected value is used as an estimator of a policy's value: $\mathbb{E}[r|s = s_0, a = a_r] = 3$ is a bad estimator of the two real possible rewards. This can be even worse if the returns are multi-modal. The main problem arises when a question like *If I have a budget of b, should I select the risky action?* is asked. If the agent only had the information about the expected value instead of the distribution the answer might be affirmative, but the answer will be different if we account for the full distribution. The problem becomes more relevant if we observe that any given reward will affect the budget we will have in a future decision step. Hence a budget-aware learning policy will consider the current amount of resources and the reward distribution before choosing an action. Let $b(0)$ the amount of resources an agent has at time $t = 0$, then, if the agent follows a budget-aware policy, it will maximize its expected return while ensuring that $b(t) > 0; \forall t_i > t_0$.

Areas where budget and risk consideration are needed are abundant, some examples are: tariff generation for energy markets [16], heating management in smart houses [9], and online advertising [10]. Even when there has been work on risk in RL [5, 7, 11, 13], to our knowledge none of them have addressed the issue of budget and risk, even though these two concepts are very related.

In this paper we make use of the distributional information provided by the C51 algorithm proposed by Bellemare [2] to build budget- and risk-aware policies. This task is not trivial because the budget is a variable that changes with every reward received (see Eq. 2). For this reason, simply including it as part of the state will lead to discretization issues and an exponential increase in the number of states that could make it infeasible to find an optimal policy. To address this issue we followed a different strategy by

*Work performed during an internship at the *Intelligent and Autonomous Systems* Group at CWI.

*Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. Andre, S. Koenig (eds.), July 2018, Stockholm, Sweden*

https://doi.org/doi

bootstrapping from an unbudgeted policy to learn budget-aware policies.

This paper is structured as follows: Section 2 describes the closest related work to risk and budget, Section 3 shows the relevant background including distributional RL. Section 4 describes our approach to use distributional RL to learn budget and risk-aware policies. Section 5 presents the experimental setting and our results. Section 6 presents a discussion about related work and finally in Section 7 we offer some conclusions.

## 2 RELATED WORK

The concept of budget has been studied before, for example in multi-armed bandits (MABs). Tran-Thanh et al. proposed the budget-limited MAB with an overall budget, where the exploration and exploitation phases have a limited budget [19]. However, the setting is more limited than the one considered here, since the reward is observed immediately after executing an action (pulling an arm), while we used an experiment setting where the reward is received after following a (small) series of actions.

Safety in RL [12] is a concept that is gaining a lot of interest. It includes a wider variety of topics such as safe interruptibility, distributional shift, robustness to adversaries, and safe exploration. Approaches to address safe exploration in RL have proposed ideas such as the use of ergodicity in MDPs to define safety [14], or the use of backup policies to help the system return to a controlled (safe) state [6].

One of the earliest works that analysed risk in reinforcement learning was presented by Heger [7]. The basic idea is to adjust the Q-learning algorithm to account for risky behaviors. Heger proposed $\hat{Q}$-learning which maximizes the *worst-case* scenarios, this is, it finds policies that minimize the worst-case total discounted costs. However, only considering worst-case scenarios might be too extreme. Instead, other works have studied a more general version to parametrize the desired level of risk and consider, for example, risk-averse and risk-seeking behaviors. Examples of this case are the *risk-sensitive* algorithms proposed by Mihatsch and Neuneir [13] which use the *variance* of the returns [18] as a measure of risk. Yu and Nikolova [22] proposed the use of several variations of *Value at Risk*, which a common measure of risk in finance [8, 21]. Similar ideas had been analysed in the context of actor-critic algorithms [11]. An important remark is that none of these works consider risk and budget jointly as we do in this work.

## 3 BACKGROUND

In this section we review the frameworks of reinforcement learning and distributional reinforcement learning.

### 3.1 Reinforcement learning

Reinforcement learning (RL) formalizes the interaction of an agent with the environment using a Markov decision process (MDP). An MDP is defined by the tuple $\langle S, A, R, T\rangle$ where $S$ represent the world divided up into a finite set of possible states. $A$ represents a finite set of available actions. The transition function $T : S \times A \to \Delta(S)$ maps each state-action pair to a probability distribution over the possible successor states, where $\Delta(S)$ denotes the set of all probability distributions over $S$. Thus, for each $s, s' \in S$ and $a \in A$, the function $T$ determines the probability of a transition from state $s$ to state $s'$ after executing action $a$. The reward function $R : S \times A \times S \to \mathbb{R}$ defines the immediate and possibly stochastic reward that an agent would receive for being in state $s$, executing action $a$ and transitioning to state $s'$.

MDPs are adequate models to obtain optimal decisions in *single* agent environments. Solving an MDP will yield a policy $\pi : S \to A$, which is a mapping from states to actions. An optimal policy $\pi^*$ is the one that maximises the expected discounted reward. There are different techniques for solving MDPs assuming a complete description of all its elements. One of the most common techniques is the value iteration algorithm [3] which is based on the Bellman equation:

$$V^{\pi}(s) = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V^{\pi}(s')],$$

with $\gamma \in [0, 1)$. This equation expresses the *value* of a state which can be used to obtain the optimal policy $\pi^* = \arg\max_{\pi} V^{\pi}(s)$, i.e., the one that maximizes that value function, and the optimal value function is

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in S.$$

$Q$-learning [20] is one well known algorithm for RL. It has been devised for stationary, single-agent, fully observable environments with discrete actions. A $Q$-learning agent can be in a state $s \in S$ and can choose an action $a \in A$. It keeps the estimate of its expected payoff starting in state $s$, taking action $a$ as $Q(s, a)$. Each entry $Q(s, a)$ is an estimate of the corresponding optimal $Q^*$ function that maps state-action pairs to the discounted sum of future rewards when starting with the given action and following the optimal policy thereafter. Each time the agent makes a transition from a state $s$ to a state $s'$ via action $a$ receiving payoff $r$, the $Q$ table is updated as follows:

$$Q(s,a) = Q(s,a) + \alpha[(r + \gamma \max_{b} Q(s', b)) - Q(s,a)]$$

with the learning rate $\alpha$ and the discount factor $\gamma \in [0, 1]$ being parameters of the algorithm, with $\alpha$ typically decreasing over the course of many iterations. $Q$-learning is proved to converge towards $Q^*$ if each state-action pair is visited infinitely often under specific parameters [20].

### 3.2 Distributional RL

A common approach to reinforcement learning is to model the expectation of the returns, commonly referred as value. However, if the returns are a bimodal (or even a multi-modal) random variable, the expected value might be a very bad estimator of the goodness of an action, i.e., the positive expected value of an action might be the result of negative and positive rewards. In this context, Bellemare at al. [2] proposed a distributional perspective of RL, which argues for the fundamental importance of the full distribution of the random returns received by a reinforcement learning agent, whose expectation is the value $Q$. Following this idea they propose a random variable $Z(s, a)$ which is a mapping from state-action pairs to distributions over the returns, called *value distribution*. This means that, instead of having a scalar for each state-action pair a distribution over the returns is approximated, in particular using a categorical distribution with $n$ categories, or atoms, over a

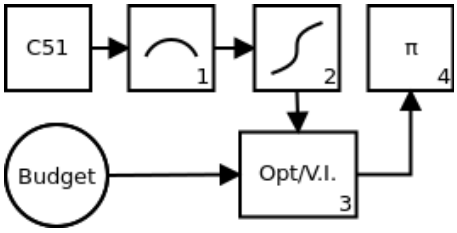


**Figure 1: BRAD-RL building blocks. BRAD-RL uses the result of C51 (unbudgeted case): the value distributions (1) to compute cumulative density distributions (2), then these are plugged together with the budget into a modified value iteration (3) to compute a budget-aware policy (4).**

fixed range. Stated in another way, whenever a new reward $r_t$ is received it is distributed into one of the $n$ possible atoms. The authors evaluated different number of atoms and reported that $n = 51$ yield the best scores, hence the algorithm's name C51, shown in Algorithm 1. This distribution describes the probability of a reward in the distribution's support $\hat{z}$, which is kept constant by using a projection technique at each update (lines 5-9). We will refer with $p_i(s, a)$ to the $ith$ probability that corresponds to the $ith$ atom in $\hat{z}$ for any given state-action pair. With this in mind the expected value $Q$ can be calculated using Eq. 1,

$$Q(s, a) = \sum(\hat{z} \cdot Z(s, a)) = \sum_i z_i p_i(s, a). \quad (1)$$

The limits of the support $\hat{z}$ are $z_0 = V_{min}$ and $z_{n-1} = V_{max}$, the set that contains all the available supports $z_i$ is $\bar{z}$, and the support step size is $\Delta z = \frac{V_{max} - V_{min}}{N-1}$.

Note that C51 algorithm uses a discrete approximation of the real value distribution. Using this discrete version has the advantages of being highly expressive and computationally friendly [15]. Lastly, it is important to mention that C51 used a simple $\epsilon-$greedy policy over the expected action values [2], without using the full distribution information in the action selection.

**Algorithm 1** Categorical update algorithm as proposed in [2].

**Require:** A transition $x_t, a_t, r_t, x_{t+1}, \gamma_t \in [0, 1]$
1: $Q(x_{t+1}, a) := \sum_i z_i p_i(x_{t+1}, a)$
2: $a^* \leftarrow argmax_a\, Q(x_{t+1}, a)$
3: $m_i = 0, i \in 0, ..., N-1$
4: **for** $j \in 0, ..., N-1$ **do**
5: $\quad \hat{\tau}_{z_j} \leftarrow [r_t + \gamma_t + z_j]_{V_{MIN}}^{V_{MAX}}$
6: $\quad b_j \leftarrow (\tau_{z_j} - V_{MIN}/\Delta_z)$
7: $\quad l \leftarrow \lfloor b_j \rfloor, u \leftarrow \lceil b_j \rceil$
8: $\quad m_l \leftarrow p_j(x_{t+1}, a^*)(u - b_j)$
9: $\quad m_u \leftarrow p_j(x_{t+1}, a^*)(b_j - l)$
10: **end for**
11: **return** $-\sum_i m_i \, log \, p_i(x_t, a_t)$

## 4 BRAD-RL

The C51 distributional approach provides valuable information which we can leverage to take budget- and risk- aware decisions. Before we describe our algorithm we will define the budget as a cumulative variable which is updated at every time $t$ when a new reward is received, as indicated by Eq. 2.

$$b(t + 1) = r(t) + b(t) \quad (2)$$

Since the reward distribution (provided by C51) and the current budget are the two main inputs of our Budget and Risk Aware RL (BRAD-RL) algorithm, we are now ready to describe it. In a nutshell the process is as follows. We assume that we learned $Z(s, a)$ by using C51 in an *unbudgeted* process. Then, we use the corresponding CDF to calculate an *optimistic* estimate of starting an episode with a certain amount of budget. Finally, we use value iteration to obtain a more accurate estimate of a budget-aware policy. A high-level idea of this process is depicted in Figure 1 and the details of this process will be described in this section.

### 4.1 Optimistic estimate

To compute the optimistic estimate for any given budget we use the cumulative density function (CDF) for each action of the distribution $Z$ learned by C51 algorithm, which is defined for a fixed support $z \in [V_{min}, V_{max}]$. Then, for every action $a$, the CDF of $Z$ is defined as $F_{Z_a}(z)$.

Now, assume the current agent's budget is $b$, then, for any state-action pair the probability of receiving an episodic return $r_e$ smaller than $-b$ corresponds to the probability of running out of budget as stated by Eq. 3

$$\Pr(r_e \leq -b|s, a) = F_{Z_a}(-b|s). \quad (3)$$

However, since we are dealing with a discrete version of the CDF we can compute

$$Pr(r_e \leq 0|s, a) = \sum_{i=0}^{h} p_i(s, a)$$

where $h$ corresponds to the first index of the support where $V > -b$. We propose an alternative measurement as the probability $\beta$ of having an episodic return $r_e$ larger than a certain budget $b$, in state $s$ taking action $a$, which we write as,

$$\beta(b, s, a) = 1 - \Pr(r_e \geq b|s, a). \quad (4)$$

With Eq. 4 we can define an *optimistic estimate* starting an episode at state $s_0$ while having a certain budget as

$$Q_{optimistic}(\langle s_0, b\rangle, a) = Q(s_0, a) + \beta(b, s_0, a)\frac{\gamma_e}{1 - \gamma_e}Q^*, \quad (5)$$

where $Q^* = \max_a Q(s, a)$, Eq. 5 is optimistic because it assumes that if the episodic return is larger than the current budget, with a probability $\beta$ it will receive the best known episodic return $Q^*$ at each episode ever after discounted by $\gamma_e$. In what follows we show that this estimate is a particular case of the value iteration form.

### 4.2 Budget-aware value iteration

According to Sutton and Barto [17] the process of value iteration is described by the update operation

$$V(s) = \max_a \sum_{s', r} p(s', r|s, a)\left[r + \gamma V(s')\right],$$

**Table 1: Possible episodic returns. Absorbing states are marked with ∗.**

| Route | Reward |
|---|---|
| $s_3$, $s_4$* | -1-1+6=4 |
| $s_1$* | -1-4=-5 |
| $s_1$, $s_0$* | -1-1+12=10 |

that can be written for $Q$ as:

$$Q(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_a Q(s',a)\right]$$

for all states $s \in S$. We aim to estimate the expected value of starting an episode from a fixed state $s_0$, with a certain amount of budget, given the return distribution learned by C51 algorithm, i.e., all the possible returns. Also we know from Eq. 2 that the current budget depends on the last recorded budget and the last received reward. Thus, the expected value of having a budget $b$ as calculated by value iteration is defined by Eq. 6, where $p_i(s,a)$ and $z_i$ are the same as described in Eq. 1, and $\gamma_e$ is an episodic discount rate.

$$Q(\langle s_0, b\rangle, a) = \sum_i p_i(s_0,a)[z_i + \gamma_e \max_a Q(\langle s_0, b + z_i\rangle, a)] \quad (6)$$

It is important to note that if $Q(\langle s_0, b+z_i\rangle, a) = 0$, then the value of having a budget of $b$ corresponds to the C51 episodic expectation as shown in Eq. 1. On the other hand, if we assume that we have a large amount of budget so that we obtain the best possible return for an infinite number of episodes afterwards, then

$$\begin{aligned} Q(\langle s_0, b\rangle, a) &= \sum_i p_i(s_0,a)\cdot z_i + \sum_i p_i(s_0,a)\cdot \frac{\gamma_e}{1-\gamma_e} Q^* \\ &= Q(s_0,a) + \frac{\gamma_e}{1-\gamma_e} Q^*. \end{aligned}$$

Since for large budgets $\beta \to 1$, then if these assumptions hold Eq. 5 corresponds to Eq. 6.

One last issue to solve is the fact that the budget is a continuous variable. To deal with this we discretized the budget in a finite number of bins within the same range $[V_{min}, V_{max}]$ for which the support $z$ is defined. Then, after calculating $b + z_i$ we projected the result to the closest integer bin and performed the update.

## 5 EXPERIMENTS

This section describes our experimental results. First, we present a grid-world scenario that highlights the importance of considering a budget. Then we provide details about the C51 training process and we present experimental results comparing BRAD-RL with standard C51 and two baselines (risky and safe strategies). Finally, we show results of the optimistic approach and the value iteration results.

### 5.1 Scenario description

To model a problem where there exists a trade off between expected reward and risk we propose a grid-world created as a custom Open AI Gym [4] environment depicted in Figure 2. The scenario consists of a 5 column grid with two fixed absorbing states $s_0$ and $s_4$ at the edges. The agent starts in the middle, $s_2$, with two possible actions, to move to the left ($a_0$) or to the right ($a_1$). With probability $p_w$ the state $s_1$ becomes an absorbing state and provides a negative reward, $r_{danger}$, to the agent. With probability $1 - p_w$ it can safely pass $s_1$ towards $s_0$. Every step, (including the ones to the absorbing states) provide the agent with a reward of $-1$. Possible episodic returns are shown in Table 1. The reason behind using a small grid-world to test our agent is that we want to model the fact that a series of actions is required to complete the task, and there are many ways to do so. For instance, the series of actions *Right, Left, Left, Left* and *Left, Left* will all lead to state $s_0$ with a probability $1 - p_w$.

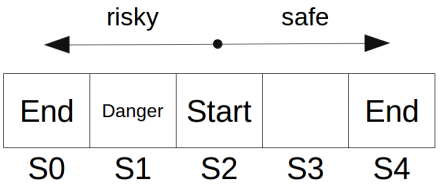


**Figure 2: The test scenario used to generate the C51 estimations. There is a probability $p$ of ending the episode with a negative reward in $s_1$, otherwise with probability $1 - p$ the agent can safely pass through.**

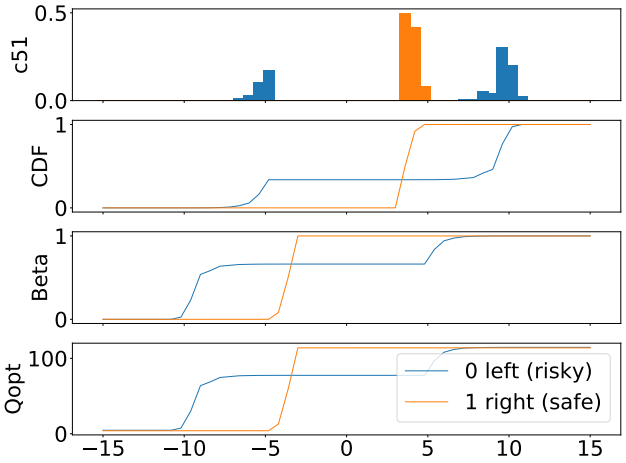


**Figure 3: State information at the beginning of an episode. From top to bottom: *(C51)* return probability distribution, *(CDF)* cumulative density function, the probability $\beta$ (*Beta*) of having a positive budget at the end of the episode as a function of the budget, and the optimistic value expectation $Q_{opt}$.**

Two standard outcomes in this world are easily seen: if the agent picks the route towards $s_4$ it will receive deterministically a reward of $r_{det}$. However, if it chooses a path towards $s_0$, with probability $p_w$, it will receive a reward of $r_{danger}$ and, with probability $1 - p_w$, it will receive a reward of $r_{max}$. By using C51 algorithm we can estimate not only the expected value of each action, but the complete distribution. Figure 3 (top) shows the information for the initial state $s_2$ after being trained using C51 with values $r_{danger} = -5, r_{det} = 4, r_{max} = 10$.

## 5.2 C51 training

To obtain the returns' distributional information we trained an agent for a total of 20,000 episodes using the C51 categorical algorithm shown in Alg. 1 as follows. We used 1,000 episodes to initialize a replay memory which was updated after the same number of steps. Then, we followed an $\epsilon$-greedy strategy for 9,000 episodes to pick up an action (line 2 of Alg. 1). Finally, we trained the model for the remaining 10,000 episodes. The result of this process for state $S2$ can be seen in the first panel of Fig. 3. An important remark is the fact that C51 makes no use of the distributional information for action selection, instead it just considers the expectation $Q(s_{t+1}, a) := \sum_i z_i p_i(s_{t+1}, a)$. For this reason, we used the distributional information along with the expectation to build a budget-aware algorithm; first by calculating an optimistic estimate and later by using value iteration to calculate the real estimate.

## 5.3 Learning budget aware policies

To illustrate the usefulness of knowing the reward distribution provided by C51 algorithm to learn budget-aware policies we will use the results of Fig. 3. The top subplot (*C51*) shows the output of using C51 algorithm to estimate the reward distribution for a given problem; where an agent has to pick between actions $a_0$ (in blue) and $a_1$ (in orange). C51 estimated that there are two possible outcomes for $a_0$, a positive outcome and a negative outcome. On the other hand $a_1$ has only one positive outcome.

Hence, $a_0$ can be considered *risky* strategy whereas $a_1$ is a *safe* one. The action the agent has to pick depends on the chosen optimality criterion. If we use the standard expected value criterion (see Eq. 1), which does not consider the budget, then the selection would be $a_0$, because it has the larger expected value, even though it could be considered a risky option. The second subplot in Fig. 3 shows the CDF plot for each action and the third subplot shows the $\beta$ probability shown in Eq. 4. The optimistic expected value described by Eq. 5 is shown in the fourth subplot. If the agent tries to maximize its probability of not running out of resources by the end of an episode, or if maximizes its optimistic estimate, it would follow a policy derived from either of these subplots. As an example, for $b = 0$ then $\arg\max_a \beta = \arg\max_a \hat{Q}_{opt} = right$, so the optimal action is move to the right.

An interesting result of following the budget-aware optimistic policy is shown in Fig. 4, where this policy is compared to a risky and safe policy for 15 episodes, starting with a budget of zero. We would expect that our budget-aware agent picked to be risky only when its budget was enough to overcome the worst possible outcome. And this behavior can be observed in the second subplot, which shows the budget at the end of each episode. During the first two episodes the budget-aware agent, labeled *budget c51*, chooses the safe action, hence following the behavior of the safe agent. After the second episode it determines it has enough resources to play risky and chooses left with a bad outcome. Then it decides to move right for the next two episodes, which can be seen because the slopes of the budget-aware agent and the safe agent are equal. When it has enough budget again it plays being risky and its accumulated budget increases rapidly. In contrast, the risky agent ran out of budget on the first episode and could not keep playing.


**Figure 4: Top: the rewards received by an agent following an optimistic budget-aware policy for 15 episodes, as estimated by Eq. 5. Bottom: the corresponding accumulated budget. The risky agent can run out of budget immediately, while the budget-aware agent modifies its policy according to its resources.**

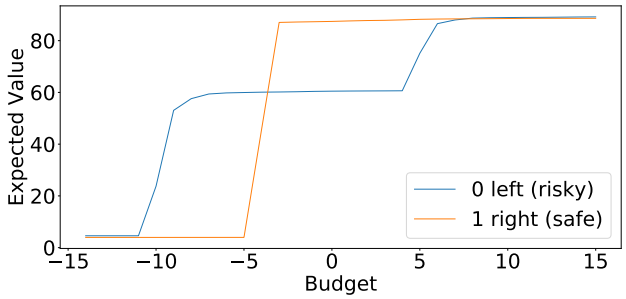


**Figure 5: Expected value of starting an episode with a certain budget ($s_2$) calculated by value iteration, Eq 6. For example, when the agent starts with a budget$> -4$ the value of taking a safe action is higher.**

## 5.4 Budget-aware value iteration

Fig. 5 shows the Q-values of having a budget in the range $[-15, 15]$ for the initial state $s_0 = S2$, calculated using value iteration as defined by Eq. 6. Selecting the action with the largest expected q-value, using a max or softmax strategy, for each budget will lead to our budget-aware policy, which we show in Fig. 6 for states $S1$ and $S3$ also. To show the relative differences between the expected values for each action, Fig. 6 uses a softmax strategy. It can be seen from the plot that regardless of the budget, if the agent is in state $s_1$, the optimal action is to move to the left. While in $S2$, the policy indicates that for a budget in the range [-10, -4] the optimal action is going to the left as well. This is because C51 estimated that the expected return of moving to the right is 4, so, even if the current budget is -4, according to Eq. 2, the best possible budget at the end of the episode will be 0. So the best possible action is moving to the left and aim to obtain the large risky reward of 10, so as to end with a positive balance. Something similar happens in state $S3$: if the agent "owes" four or more units of budget, the only way to obtain a positive balance is moving to the left.

A remark worth noting is that in our test scenario both the optimistic and the value iteration estimates lead to the same policy,

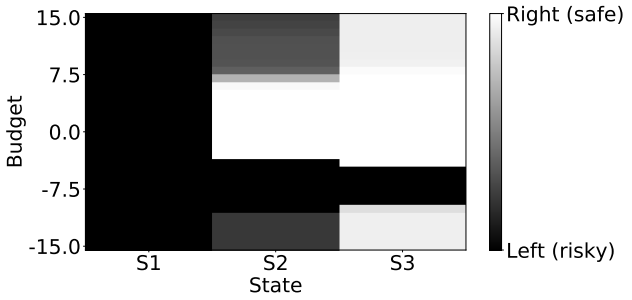


**Figure 6: Probability of selecting right (white/safe) or left (black/risky) after running the VI budget-aware (Eq. 6) on every state.**

shown in Fig. 6. One can tell this by observing that the budget intervals where the risky action expected values are larger, compared to the safe one, are identical (see bottom subplot of Fig. 3 and Fig. 5).

## 6 DISCUSSION

As far as we know this work is the first one to use a distributional approach to deal with the problem of budget and risk in RL, in contrast with the traditional RL approaches that just add the budget as part of the state representation. To do so we made some assumptions to set a foundation for further research. Perhaps the strongest assumption is the fact that our approach relies on the existence of prior knowledge about the return distribution, in our case this information was obtained from C51 algorithm's output. So, if the only way to acquire the knowledge to estimate the return distribution is through direct interaction with the environment, then a future research path is to develop an extension of this work that considers budget during the learning process. However, if historical information is available, e.g. a stock price time-series, then we can use that offline information to train a categorical model like C51 does, and then use our approach to take budget-aware decisions. Another implicit assumption arises from the fact that the agent attempts to completely avoid the risk of running out of budget by the end of an episode, therefore one could say that it has a *risk-averse* profile. However, what if the agent is willing to expose itself to a certain amount of risk? Then, the policy of a risk-seeking agent will be different. As an example, if the agent can tolerate some risk it might choose to move to the left when having a budget of 4 only if the probability of receiving the -5 return is smaller than its risk tolerance. The benefits of adding as an input the agent's risk profile would be better perceived in scenarios with more than two return outcomes. Finally, another possible follow-up work is to convert BRAD into an iterative process: use C51 to learn an *intra-episode* policy, and then improve it using value iteration, just as it was described, to create an *inter-episode* budget-aware policy. Then the value iteration output can be used to relearn a new *intra-episode* policy and so on.

## 7 CONCLUSIONS

Safe reinforcement learning has many real-world applications where constraints and restrictions cannot be ignored [12]. One way to model those is by means of a budget, which the agent is not allowed to deplete during its interactions with the environment. In this work we contribute with an approach to use distributional return information to take budget- and risk-aware decisions. We also showed that in a grid-world scenario, a simple optimistic approach is good enough to estimate a budget-aware policy. In other more complex scenarios with a richer variety of returns this may not be the case, and further experiments are needed. Finally, we pointed out some possible future research directions such as online budget learning and adding a parameter to account for risk tolerance.

## ACKNOWLEDGEMENTS

This research has received funding through the ERA-Net Smart Grids Plus project Grid-Friends, with support from the European Union's Horizon 2020 research and innovation programme.

## REFERENCES

[1] Nicholas Barberis and Wei Xiong. 2012. Realization utility. *Journal of Financial Economics* 104, 2 (2012), 251–271.
[2] Marc G Bellemare, Will Dabney, and Rémi Munos. 2017. A distributional perspective on reinforcement learning. *Proceedings of the 34th International Conference on Machine Learning* (2017).
[3] Richard Bellman. 1957. A Markovian decision process. *Journal of Mathematics and Mechanics* 6, 5 (1957), 679–684.
[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. (2016). arXiv:arXiv:1606.01540
[5] Javier García and Fernando Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* 16 (2015), 1437–1480.
[6] Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. 2008. Safe exploration for reinforcement learning.. In *ESANN*. 143–148.
[7] Matthias Heger. 1994. Consideration of risk in reinforcement learning. In *Machine Learning Proceedings 1994*. Elsevier, 105–111.
[8] Philippe Jorion (Ed.). 20006. *Value at Risk*. McGraw-Hill Professional Publishing, USA.
[9] Hussain Kazmi, Fahad Mehmood, Stefan Lodeweyckx, and Johan Driesen. 2018. Gigawatt-hour scale savings on a budget of zero: Deep reinforcement learning based optimal control of hot water systems. *Energy* 144 (2018), 159–168.
[10] Deguang Kong, Xiannian Fan, Konstantin Shmakov, and Jian Yang. 2018. A Combinational Optimization Approach for Advertising Budget Allocation. In *Companion of the The Web Conference 2018 on The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 53–54.
[11] Prashanth L A and Mohammad Ghavamzadeh. 2013. Actor-Critic Algorithms for Risk-Sensitive MDPs. *NIPS* (2013).
[12] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. 2017. AI Safety Gridworlds. *arXiv:1711.09883* (2017).
[13] Oliver Mihatsch and Ralph Neuneier. 2002. Risk-Sensitive Reinforcement Learning. *Machine Learning* (2002).
[14] Teodor Mihai Moldovan and Pieter Abbeel. 2012. Safe exploration in Markov decision processes. *International Conference on Machine Learning* (2012).
[15] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. *International Conference on Machine Learning* (2016).
[16] Jonathan Serrano-Cuevas, Ansel Y Rodríguez González, Miguel Palacios Alonso, Enrique MunĪČoz De Cote, and Luis Enrique Sucar. 2015. Distributed energy procurement and management in smart environments. In *Smart Cities Conference (ISC2), 2015 IEEE First International*. IEEE, 1–6.
[17] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.
[18] A Tamar, D Di Castro, and S Mannor. 2016. Learning the variance of the reward-to-go. *Journal of Machine Learning Research* (March 2016).
[19] Long Tran-Thanh, Archie C. Chapman, Enrique Munoz de Cote, Alex Rogers, and Nicholas R. Jennings. 2010. Epsilon-First Policies for Budget-Limited Multi-Armed Bandits. *AAAI* (2010).
[20] Christopher Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8 (1992), 279–292.
[21] Evert Wipplinger. 2007. Philippe Jorion: Value at Risk-The New Benchmark for Managing Financial Risk. *Financial Markets and Portfolio Management* 21, 3 (2007), 397.
[22] Jia Yuan Yu and Evdokia Nikolova. 2013. Sample Complexity of Risk-Averse Bandit-Arm Selection.. In *IJCAI*. 2576–2582.